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

```
import tensorflow.keras as keras
import tensorflow as tf
import pandas as pd
import numpy as np
import os
from tqdm import tqdm
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import LSTM, GRU
from tensorflow.keras.layers import BatchNormalization
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
#!pip install --upgrade tensorflow #run everytime when u come here
#!pip install keras==2.2.4
#!pip install grpcio==1.24.3
from tensorflow.keras.layers import Convolution1D, GlobalMaxPooling1D
from tensorflow.keras.callbacks import ModelCheckpoint
#from tensorflow.keras import backend as K
from prettytable import PrettyTable
x = PrettyTable()

x.field_names = ["model", "loss", "accuracy",'auc']

from tensorflow.keras.preprocessing import sequence, text
```

```
Using TensorFlow backend.
```

```
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.google

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

```
tf.version
print(tf.__version__)

```

```
2.1.0
```

```
data=pd.read_csv('/gdrive/My Drive/preprocessed_data.csv')
print(data.shape)
data.head()
```

```
(109248, 9)
```

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	project_is_approved	clean_categor
0	ca	mrs	grades_prek_2	53	1	math_scie
1	ut	ms	grades_3_5	4	1	specialne
2	ca	mrs	grades_prek_2	10	1	literacy_langu
3	ga	mrs	grades_prek_2	2	1	appliedlearn
4	wa	mrs	grades_3_5	2	1	literacy_langu

```
data.price.max()
data.price.min()
```

```
0.66
```

```
y=data['project_is_approved']
#data['price']=data.price.apply(lambda x:round(x))
data.drop(['project_is_approved'],axis=1,inplace=True)
```

```
x_train, x_test, y_train, y_test=train_test_split(data,y,test_size=0.18,stratify=y ,random_state=42)
x_train, x_cv, y_train, y_cv=train_test_split(x_train,y_train,test_size=0.15,stratify=y_train ,random_state=42)

```

```
def get_numericals(df,col,val_dict=1):
  if(val_dict==1):
    val_dict={}
    print('here')
    valuess=x_train[col].unique()
    #print(valuess)
    for i in range(len(valuess)):
        val_dict[valuess[i]]=i

    #print(val_dict)
    df[col]= df[col].map(val_dict)
    return(val_dict)
  else:
    valuess=df[col].unique()
    #print(valuess)
    for i in range(len(valuess)):
        if(valuess[i] not in mp.keys() ):
          mp[valuess[i]]=len(valuess)+1
    #print(val_dict)
    df[col]= df[col].map(mp)


  #
  #df[col]=df[col].map(val_dict)
```

```
mp=get_numericals(x_train,'clean_categories')
get_numericals(x_cv,'clean_categories',mp)
```

```
get_numericals(x_cv,'clean_categories',mp)
get_numericals(x_test,'clean_categories',mp)


mp=get_numericals(x_train,'clean_subcategories')
get_numericals(x_cv,'clean_subcategories',mp)
get_numericals(x_test,'clean_subcategories',mp)



mp=get_numericals(x_train,'school_state')
get_numericals(x_cv,'school_state',mp)
get_numericals(x_test,'school_state',mp)


mp=get_numericals(x_train,'teacher_prefix')
get_numericals(x_cv,'teacher_prefix',mp)
get_numericals(x_test,'teacher_prefix',mp)



mp=get_numericals(x_train,'project_grade_category')
get_numericals(x_cv,'project_grade_category',mp)
get_numericals(x_test,'project_grade_category',mp)


```

```
here
here
here
here
here
```

```
tokenizer = text.Tokenizer(num_words=200000)
tokenizer.fit_on_texts(x_train.essay)
train_sequences_tr = tokenizer.texts_to_sequences(x_train.essay)
train_sequences_cv = tokenizer.texts_to_sequences(x_cv.essay)
train_sequences_te = tokenizer.texts_to_sequences(x_test.essay)


```

```
maxi=-1
for i,rows in x_train.iterrows():

    tokens=rows.essay.split()
    if(len(tokens)>maxi):
        maxi=len(tokens)



print(maxi)

```

```
331
```

```
vocab_size=len(tokenizer.word_index)+1
max_rev_len=maxi
vocab_size


```

```
48876
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences


train_sequences_tr_pad = pad_sequences(train_sequences_tr, maxlen=max_rev_len)
train_sequences_cv_pad = pad_sequences(train_sequences_cv, maxlen=max_rev_len)
train_sequences_te_pad = pad_sequences(train_sequences_te, maxlen=max_rev_len)

```

```

```

```
import pickle
with open('/gdrive/My Drive/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  model

```

```
```

```
EMBEDDING_DIM=300
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
count=0
embedding_matrix1 = np.zeros((len(word_index) + 1, EMBEDDING_DIM))
for word, i in word_index.items():
    embedding_vector = glove_words.get(word)
    if embedding_vector is not None:
        count+=1
        # words not found in embedding index will be all-zeros.
        embedding_matrix1[i] = embedding_vector
print(count)


```

```
Found 48875 unique tokens.
43914
```

```
embedding_layer_text1 = Embedding(len(word_index) + 1,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix1],
                            input_length=max_rev_len,
                            trainable=False)

sequence_input = keras.Input(shape=(max_rev_len,))
emb_text=embedding_layer_text1(sequence_input)
merged=LSTM(96,return_sequences=True )(emb_text)
merged_text=LSTM(64,return_sequences=False)(merged)


```

```
from sklearn.preprocessing import MinMaxScaler
v=MinMaxScaler()
x_train['price']=v.fit_transform(x_train.price.values.reshape(-1,1))
x_cv['price']=v.transform(x_cv.price.values.reshape(-1,1))
x_test['price']=v.transform(x_test.price.values.reshape(-1,1))
```

```
v=MinMaxScaler()
x_train['teacher_number_of_previously_posted_projects']=v.fit_transform(x_train.teacher_number_of_previously_posted_projects.values.reshape(
x_cv['teacher_number_of_previously_posted_projects']=v.transform(x_cv.teacher_number_of_previously_posted_projects.values.reshape(-1,1))
x_test['teacher_number_of_previously_posted_projects']=v.transform(x_test.teacher_number_of_previously_posted_projects.values.reshape(-1,1))
```

```
#def auc(y_true, y_pred):
#    auc = tf.metrics.auc(y_true, y_pred)[1]
#    K.get_session().run(tf.local_variables_initializer())
#    return auc


grade=keras.Input(shape=(1,),name='grade')
grade_emb=Embedding( len(x_train.project_grade_category.unique())+1,2,input_length=1)(grade)
grade_emb=keras.layers.Flatten()(grade_emb)


school_state=keras.Input(shape=(1,),name='school_state')
school_state_emb=Embedding( len(x_train.school_state.unique())+1,2,input_length=1)(school_state)
school_state_emb=keras.layers.Flatten()(school_state_emb)


teacher_prefix=keras.Input(shape=(1,),name='teacher_prefix')
teacher_prefix_emb=Embedding( len(x_train.teacher_prefix.unique())+1,2,input_length=1)(teacher_prefix)
teacher_prefix_emb=keras.layers.Flatten()(teacher_prefix_emb)


cc=keras.Input(shape=(1,),name='cc')
cc_emb=Embedding( len(x_train.clean_categories.unique())+1,2,input_length=1)(cc)
cc_emb=keras.layers.Flatten()(cc_emb)



csc=keras.Input(shape=(1,),name='csc')
csc_emb=Embedding( len(x_train.clean_subcategories.unique())+1,2,input_length=1)(csc)
csc_emb=keras.layers.Flatten()(csc_emb)

price=keras.Input(shape=(1,),name='price')
```

```
pp=keras.Input(shape=(1,),name='pp')

##############################3



l=[school_state_emb,teacher_prefix_emb,grade_emb,cc_emb,csc_emb,merged_text,price,pp]
concatenated=keras.layers.Concatenate()(l)



a=Dense(128,activation='relu')(concatenated)
model=Dropout(.1)(a)

model=Dense(64,activation='relu')(model)
model=Dropout(.2)(model)

out=Dense(2,activation='softmax')(model)

#y_train= keras.utils.to_categorical(y_train)

model = keras.Model(inputs=[school_state,teacher_prefix,grade,cc,csc,sequence_input,price,pp],outputs=out)
model.summary()

```

```
Model: "model_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 331)]        0
__________________________________________________________________________________________________
school_state (InputLayer)       [(None, 1)]          0
__________________________________________________________________________________________________
teacher_prefix (InputLayer)     [(None, 1)]          0
__________________________________________________________________________________________________
grade (InputLayer)              [(None, 1)]          0
__________________________________________________________________________________________________
cc (InputLayer)                 [(None, 1)]          0
__________________________________________________________________________________________________
csc (InputLayer)                [(None, 1)]          0
__________________________________________________________________________________________________
embedding_17 (Embedding)        (None, 331, 300)     14662800    input_3[0][0]
__________________________________________________________________________________________________
embedding_19 (Embedding)        (None, 1, 2)         104         school_state[0][0]
__________________________________________________________________________________________________
embedding_20 (Embedding)        (None, 1, 2)         12          teacher_prefix[0][0]
__________________________________________________________________________________________________
embedding_18 (Embedding)        (None, 1, 2)         10          grade[0][0]
__________________________________________________________________________________________________
embedding_21 (Embedding)        (None, 1, 2)         104         cc[0][0]
__________________________________________________________________________________________________
embedding_22 (Embedding)        (None, 1, 2)         786         csc[0][0]
__________________________________________________________________________________________________
lstm_4 (LSTM)                   (None, 331, 96)      152448      embedding_17[0][0]
__________________________________________________________________________________________________
flatten_16 (Flatten)            (None, 2)            0           embedding_19[0][0]
__________________________________________________________________________________________________
flatten_17 (Flatten)            (None, 2)            0           embedding_20[0][0]
__________________________________________________________________________________________________
flatten_15 (Flatten)            (None, 2)            0           embedding_18[0][0]
__________________________________________________________________________________________________
flatten_18 (Flatten)            (None, 2)            0           embedding_21[0][0]
__________________________________________________________________________________________________
flatten_19 (Flatten)            (None, 2)            0           embedding_22[0][0]
__________________________________________________________________________________________________
lstm_5 (LSTM)                   (None, 64)           41216       lstm_4[0][0]
__________________________________________________________________________________________________
price (InputLayer)              [(None, 1)]          0
__________________________________________________________________________________________________
```

```
pp (InputLayer)                 [(None, 1)]          0
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 76)           0           flatten_16[0][0]
                                                                 flatten_17[0][0]
                                                                 flatten_15[0][0]
                                                                 flatten_18[0][0]
                                                                 flatten_19[0][0]
                                                                 lstm_5[0][0]
                                                                 price[0][0]
                                                                 pp[0][0]
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 128)          9856        concatenate_3[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 128)          0           dense_9[0][0]
__________________________________________________________________________________________________
dense_10 (Dense)                (None, 64)           8256        dropout_6[0][0]
__________________________________________________________________________________________________
dropout_7 (Dropout)             (None, 64)           0           dense_10[0][0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 2)            130         dropout_7[0][0]
==================================================================================================
Total params: 14,875,722
Trainable params: 212,922
Non-trainable params: 14,662,800
__________________________________________________________________________________________________
```

```

from tensorflow.keras.optimizers import Adam

#def auc( y_true, y_pred ) :
#    import tensorflow.compat.v1 as tf1
#    score = tf1.py_func( lambda y_true, y_pred : sklm.roc_auc_score( y_true, y_pred).astype('float32'),
#                         [y_true, y_pred],
#                         'float32',
#                         stateful=False,
#                         name='sklearnAUC' )
#    return score

model.compile(
    # Technical note: when using embedding layers, I highly recommend using one of the optimizers
    # found  in tf.train: https://www.tensorflow.org/api_guides/python/train#Optimizers
```

```
    # Passing in a string like 'adam' or 'SGD' will load one of keras's optimizers (found under
    # tf.keras.optimizers). They seem to be much slower on problems like this, because they
    # don't efficiently handle sparse gradient updates.
    #tf.train.AdamOptimizer(0.0001),
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

```
#x_cvv=pd.DataFrame([x_cv.school_state,x_cv.teacher_number_of_previously_posted_projects,x_cv.teacher_prefix,x_cv.project_grade_category,x_c
#x_trr=pd.DataFrame([x_train.school_state,x_train.teacher_number_of_previously_posted_projects,x_train.teacher_prefix,x_train.project_grade_
```

```
from sklearn.utils import class_weight
class_weights = class_weight.compute_class_weight('balanced',
                                                  np.unique(y_train),
                                                  y_train)

```

```

import sklearn.metrics as sklm

class Metrics(tf.keras.callbacks.Callback):
    def __init__(self,x,y,cvx,cvy,cl,tl):
        self.train_data = x
        self.y = y
        self.validation_data = cvx
        self.ycv = cvy
        self.tr_auc = tl
        self.cv_auc = cl









    def on_epoch_end(self, epoch, logs={}):

```

```
        score = np.asarray(self.model.predict(self.train_data))
        targ = np.argmax(self.y,axis=1)
        self.tr_auc.append(sklm.roc_auc_score(targ, score[:,1]))
        scorecv = np.asarray(self.model.predict(self.validation_data))
        targcv = np.argmax(self.ycv,axis=1)
        self.cv_auc.append(sklm.roc_auc_score(targcv, scorecv[:,1]))
        #print('here')

        #predict = np.round(np.asarray(self.model.predict([self.validation_data[0],self.validation_data[1],self.validation_data[2],self.vali


        #print(targ.shape)
        #print(score.shape)
        #print('#'*30)
        #print(score)
        #print(targ)
        #print(sklm.roc_auc_score(targ, score[:,1]))

        #self.auc.append(sklm.roc_auc_score(targ, score[:,1]))
        print(" — Train_Auc is %f and  val_Auc: %f " %(sklm.roc_auc_score(targ, score[:,1]),sklm.roc_auc_score(targcv, scorecv[:,1])))

        return

    def get_data(self):
      print(self.cv_auc)
      print(self.tr_auc)

      return(self.tr_auc,self.cv_auc)

```

```
y_train= keras.utils.to_categorical(y_train)
y_cv= keras.utils.to_categorical(y_cv)
y_test= keras.utils.to_categorical(y_test)
```

```
#filepath="/gdrive/My Drive/LSTM_assignment/best_model.hdf5"
#checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='max')
```

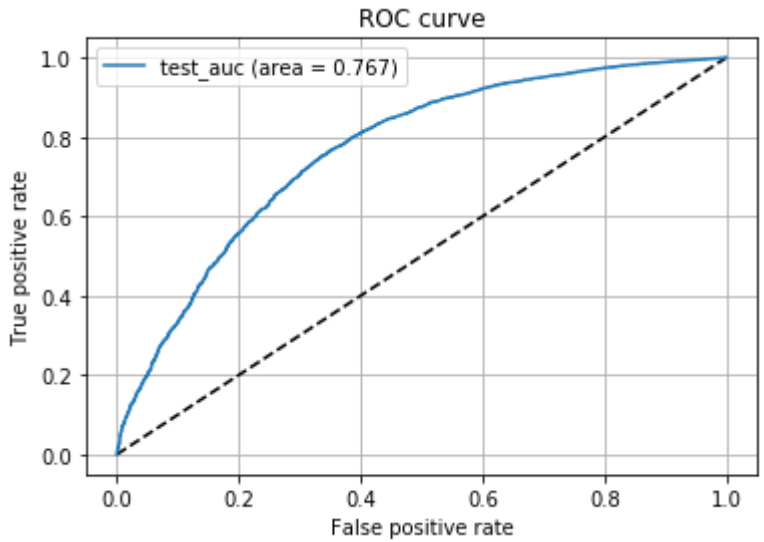
```
cl=[]
tl=[]
```

```
ti=[]
metric = Metrics([x_train.school_state,x_train.teacher_prefix,x_train.project_grade_category,x_train.clean_categories,x_train.clean_subcateg



history = model.fit(
    x=[x_train.school_state,x_train.teacher_prefix,x_train.project_grade_category,x_train.clean_categories,x_train.clean_subcategories,train
    y=y_train,
    batch_size=128,
    epochs=5,
    verbose=2,
    class_weight=class_weights,
    callbacks=[metric],
    validation_data=([x_cv.school_state,x_cv.teacher_prefix,x_cv.project_grade_category,x_cv.clean_categories,x_cv.clean_subcategories,train
)


```

```
Train on 76145 samples, validate on 13438 samples
Epoch 1/5
 — Train_Auc is 0.742032 and  val_Auc: 0.724302
76145/76145 - 83s - loss: 0.4091 - accuracy: 0.8473 - val_loss: 0.3864 - val_accuracy: 0.8505
Epoch 2/5
 — Train_Auc is 0.774062 and  val_Auc: 0.744228
76145/76145 - 79s - loss: 0.3720 - accuracy: 0.8519 - val_loss: 0.3752 - val_accuracy: 0.8530
Epoch 3/5
 — Train_Auc is 0.789232 and  val_Auc: 0.751723
76145/76145 - 79s - loss: 0.3605 - accuracy: 0.8576 - val_loss: 0.3692 - val_accuracy: 0.8541
Epoch 4/5
 — Train_Auc is 0.805414 and  val_Auc: 0.755581
76145/76145 - 79s - loss: 0.3509 - accuracy: 0.8623 - val_loss: 0.3686 - val_accuracy: 0.8509
Epoch 5/5
 — Train_Auc is 0.820299 and  val_Auc: 0.755297
76145/76145 - 80s - loss: 0.3411 - accuracy: 0.8664 - val_loss: 0.3761 - val_accuracy: 0.8466
```

```
from sklearn.metrics import roc_curve

import matplotlib.pyplot as plt
y_pred_keras =model.predict([x_test.school_state.to_numpy(),x_test.teacher_prefix.to_numpy(),x_test.project_grade_category.to_numpy(),x_test
fpr_keras, tpr_keras, thresholds_keras = roc_curve(np.argmax(y_test,axis=1), y_pred_keras)
from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)
```

```
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='test_auc (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.grid()
plt.show()
#.7111

```




```
score=model.evaluate(x=[x_test.school_state.to_numpy(),x_test.teacher_prefix.to_numpy(),x_test.project_grade_category.to_numpy(),x_test.clea
x.add_row(['model1',score[0],score[1] ,auc_keras])
```

```
19665/19665 [==============================] - 10s 527us/sample - loss: 0.3668 - accuracy: 0.8513
```

```
metrics_tr,metrics=metric.get_data()
```

```
[0.724302324854811, 0.7442279834545028, 0.7517226058662523, 0.7555813042000887, 0.755297358059789]
[0.7420323560743027, 0.774062158903527, 0.7892317054592499, 0.8054144321376007, 0.8202985276493437]
```

```
import datetime
```

```
import tensorflow as tf


#train_log_dir = '/gdrive/My Drive/Graph/model2_idf_7_9'
train_log_dir = '/gdrive/My Drive/Graph/model1/train'
cv_log_dir = '/gdrive/My Drive/Graph/model1/cv'
test_log_dir = '/gdrive/My Drive/Graph/model1/test'

train_summary_writer = tf.summary.create_file_writer(train_log_dir)
cv_summary_writer = tf.summary.create_file_writer(cv_log_dir)
test_summary_writer = tf.summary.create_file_writer(test_log_dir)

mye=len(history.history['accuracy'])
for epoch in range(len(history.history['accuracy'])):
  with train_summary_writer.as_default():
        tf.summary.scalar('loss', history.history['loss'][epoch],step=epoch)
        tf.summary.scalar('accuracy', history.history['accuracy'][epoch],step=epoch )
        tf.summary.scalar('AUC', metrics_tr[epoch],step=epoch)
  with cv_summary_writer.as_default():
        tf.summary.scalar('loss', history.history['val_loss'][epoch],step=epoch)
        tf.summary.scalar('accuracy', history.history['val_accuracy'][epoch],step=epoch )
        tf.summary.scalar('AUC', metrics[epoch],step=epoch)
  if(epoch==mye-1):
      with test_summary_writer.as_default():
          tf.summary.scalar('loss', score[0],step=mye)
          tf.summary.scalar('accuracy', score[1],step=mye )
          tf.summary.scalar('AUC', auc_keras,step=mye )

          print('here')

train_summary_writer.close()
```

```
here
```

```
%load_ext tensorboard
%tensorboard --logdir '/gdrive/My Drive/Graph/model1'
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Reusing TensorBoard on port 6006 (pid 1249), started 0:02:07 ago. (Use '!kill 1249' to kill it.)
```

TensorBoard SCALARS INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.03

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

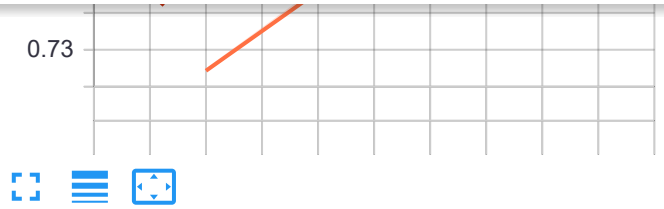
cv

test

train

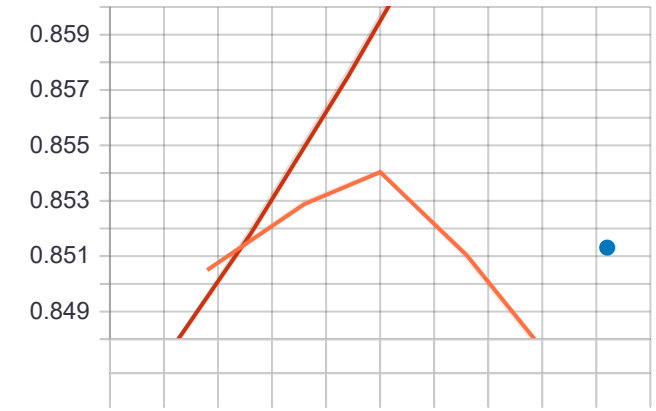
TOGGLE ALL RUNS

/gdrive/My Drive/Graph/model1



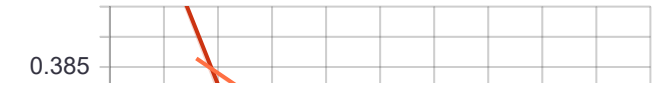
accuracy

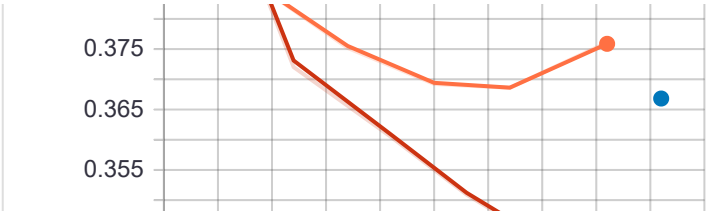
accuracy
tag: accuracy



loss

loss
tag: loss





Model 2: based out of tf-idf

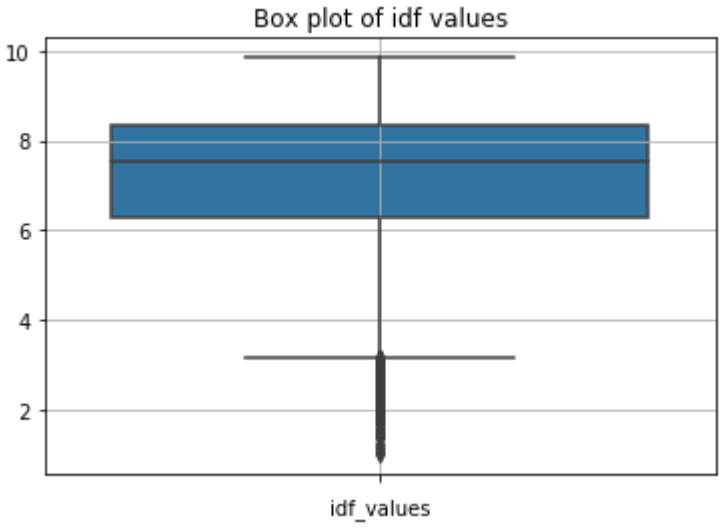
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tf_essay = TfidfVectorizer(max_features=10000,min_df=3)

text_tfidf_train = vectorizer_tf_essay.fit_transform(x_train['essay'])
text_tfidf_cv = vectorizer_tf_essay.transform(x_cv['essay'])
text_tfidf_test = vectorizer_tf_essay.transform(x_test['essay'])

```

```
vectorizer_tf_essay.idf_.shape
```

```
(10000,)
```

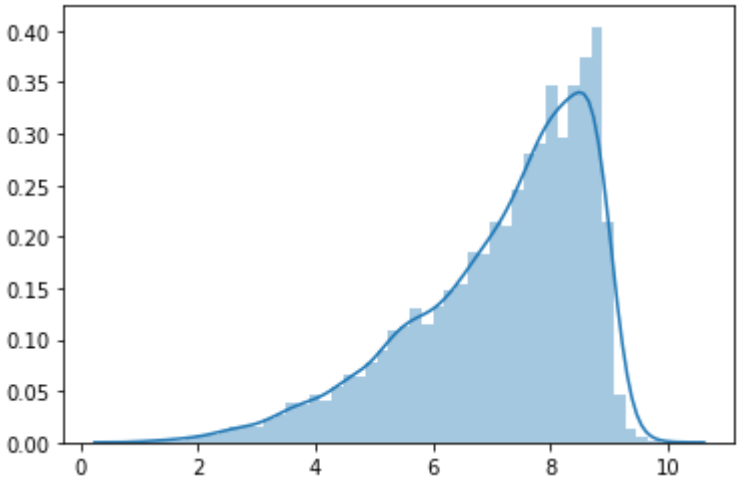
```
word2tfidf = dict(zip(vectorizer_tf_essay.get_feature_names(), vectorizer_tf_essay.idf_))

#for word, score in word2tfidf.items():
#    print(word, score)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
#as per the plot lets take values between 7 and 9
sns.boxplot(x=vectorizer_tf_essay.idf_,orient='v')
plt.grid()
#plt.xlim((1,30))
plt.xlabel('idf_values')
plt.title('Box plot of idf values')
plt.show()
```




```
sns.distplot(vectorizer_tf_essay.idf_)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6fe34ee48>
```




```
max(vectorizer_tf_essay.idf_)
```

```
9.842512556239848
```

```
list_of_imp_words=[]
```

```
for key,val in word2tfidf.items():
  if(word2tfidf[key]>4 and word2tfidf[key]<=10 ):
    #print(key)
    list_of_imp_words.append(key)
```

```
#list_of_imp_words=[]
#for key,val in word2tfidf.items():
#  if(word2tfidf[key]>4 and word2tfidf[key]<8 ):
#    list_of_imp_words.append(key)
```

```
len(list_of_imp_words)
```

```
9547
```

```

from tensorflow.keras.preprocessing.sequence import pad_sequences
tokenizer = text.Tokenizer()
#tokenizer.word_index=vectorizer_tf_essay.vocabulary_
#tokenizer.word_index=dd
tokenizer.fit_on_texts(list_of_imp_words)

train_sequences_tr = tokenizer.texts_to_sequences(x_train.essay)
train_sequences_cv = tokenizer.texts_to_sequences(x_cv.essay)
train_sequences_te = tokenizer.texts_to_sequences(x_test.essay)


train_sequences_tr_pad = pad_sequences(train_sequences_tr, maxlen=max_rev_len)
train_sequences_cv_pad = pad_sequences(train_sequences_cv, maxlen=max_rev_len)
train_sequences_te_pad = pad_sequences(train_sequences_te, maxlen=max_rev_len)

```

```
EMBEDDING_DIM=300
word_index1 = tokenizer.word_index
#word_index =vectorizer_tf_essay.vocabulary_
#word_index =dd

print('Found %s unique tokens.' % len(word_index1))

```

```
embedding_matrix = np.zeros((len(word_index1) + 1, EMBEDDING_DIM))
count=0
for word, i in word_index1.items():
  count+=1
  if(word in list_of_imp_words):
    embedding_vector = glove_words.get(word)
    if embedding_vector is not None :
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
```

```
Found 9547 unique tokens.
```

```
embedding_layer_text = Embedding(len(word_index1) + 1,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix],
                            input_length=331,
                            trainable=False)

sequence_input = keras.Input(shape=(max_rev_len,))
emb_text=embedding_layer_text(sequence_input)
merged=LSTM(96,return_sequences=True )(emb_text)
merged_text=LSTM(64,return_sequences=False)(merged)
```

```
#def auc(y_true, y_pred):
#    auc = tf.metrics.auc(y_true, y_pred)[1]
#    K.get_session().run(tf.local_variables_initializer())
#    return auc


grade=keras.Input(shape=(1,),name='grade')
grade_emb=Embedding( len(x_train.project_grade_category.unique())+1,2,input_length=1)(grade)
grade_emb=keras.layers.Flatten()(grade_emb)
```

```
school_state=keras.Input(shape=(1,),name='school_state')
school_state_emb=Embedding( len(x_train.school_state.unique())+1,2,input_length=1)(school_state)
school_state_emb=keras.layers.Flatten()(school_state_emb)


teacher_prefix=keras.Input(shape=(1,),name='teacher_prefix')
teacher_prefix_emb=Embedding( len(x_train.teacher_prefix.unique())+1,2,input_length=1)(teacher_prefix)
teacher_prefix_emb=keras.layers.Flatten()(teacher_prefix_emb)


cc=keras.Input(shape=(1,),name='cc')
cc_emb=Embedding( len(x_train.clean_categories.unique())+1,2,input_length=1)(cc)
cc_emb=keras.layers.Flatten()(cc_emb)


csc=keras.Input(shape=(1,),name='csc')
csc_emb=Embedding( len(x_train.clean_subcategories.unique())+1,5,input_length=1)(csc)
csc_emb=keras.layers.Flatten()(csc_emb)

price=keras.Input(shape=(1,),name='price')
pp=keras.Input(shape=(1,),name='pp')

##############################3


l=[school_state_emb,teacher_prefix_emb,grade_emb,cc_emb,csc_emb,merged_text,price,pp]
concatenated=keras.layers.Concatenate()(l)



a=Dense(512,activation='relu')(concatenated)
model=Dropout(.2)(a)
model=Dense(256,activation='relu')(model)
model=Dropout(.2)(model)
model=Dense(128,activation='relu')(model)
out=Dense(2,activation='softmax')(model)

#y_train= keras.utils.to_categorical(y_train)

```

```
model = keras.Model(inputs=[school_state,teacher_prefix,grade,cc,csc,sequence_input,price,pp],outputs=out)
model.summary()

```

```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            [(None, 331)]        0
__________________________________________________________________________________________________
school_state (InputLayer)       [(None, 1)]          0
__________________________________________________________________________________________________
teacher_prefix (InputLayer)     [(None, 1)]          0
__________________________________________________________________________________________________
grade (InputLayer)              [(None, 1)]          0
__________________________________________________________________________________________________
cc (InputLayer)                 [(None, 1)]          0
__________________________________________________________________________________________________
csc (InputLayer)                [(None, 1)]          0
__________________________________________________________________________________________________
embedding_23 (Embedding)        (None, 331, 300)     2864400     input_4[0][0]
__________________________________________________________________________________________________
embedding_25 (Embedding)        (None, 1, 2)         104         school_state[0][0]
__________________________________________________________________________________________________
embedding_26 (Embedding)        (None, 1, 2)         12          teacher_prefix[0][0]
__________________________________________________________________________________________________
embedding_24 (Embedding)        (None, 1, 2)         10          grade[0][0]
__________________________________________________________________________________________________
embedding_27 (Embedding)        (None, 1, 2)         104         cc[0][0]
__________________________________________________________________________________________________
embedding_28 (Embedding)        (None, 1, 5)         1965        csc[0][0]
__________________________________________________________________________________________________
lstm_6 (LSTM)                   (None, 331, 96)      152448      embedding_23[0][0]
__________________________________________________________________________________________________
flatten_21 (Flatten)            (None, 2)            0           embedding_25[0][0]
__________________________________________________________________________________________________
flatten_22 (Flatten)            (None, 2)            0           embedding_26[0][0]
__________________________________________________________________________________________________
flatten_20 (Flatten)            (None, 2)            0           embedding_24[0][0]
__________________________________________________________________________________________________
flatten_23 (Flatten)            (None, 2)            0           embedding_27[0][0]
__________________________________________________________________________________________________
flatten_24 (Flatten)            (None, 5)            0           embedding_28[0][0]
__________________________________________________________________________________________________
lstm_7 (LSTM)                   (None, 64)           41216       lstm_6[0][0]
__________________________________________________________________________________________________
price (InputLayer)              [(None, 1)]          0
__________________________________________________________________________________________________
```

```
pp (InputLayer)                 [(None, 1)]          0
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 79)           0           flatten_21[0][0]
                                                                 flatten_22[0][0]
                                                                 flatten_20[0][0]
                                                                 flatten_23[0][0]
                                                                 flatten_24[0][0]
                                                                 lstm_7[0][0]
                                                                 price[0][0]
                                                                 pp[0][0]
__________________________________________________________________________________________________
dense_12 (Dense)                (None, 512)          40960       concatenate_4[0][0]
__________________________________________________________________________________________________
dropout_8 (Dropout)             (None, 512)          0           dense_12[0][0]
__________________________________________________________________________________________________
dense_13 (Dense)                (None, 256)          131328      dropout_8[0][0]
__________________________________________________________________________________________________
dropout_9 (Dropout)             (None, 256)          0           dense_13[0][0]
__________________________________________________________________________________________________
dense_14 (Dense)                (None, 128)          32896       dropout_9[0][0]
__________________________________________________________________________________________________
dense_15 (Dense)                (None, 2)            258         dense_14[0][0]
==================================================================================================
Total params: 3,265,701
Trainable params: 401,301
Non-trainable params: 2,864,400
__________________________________________________________________________________________________
```

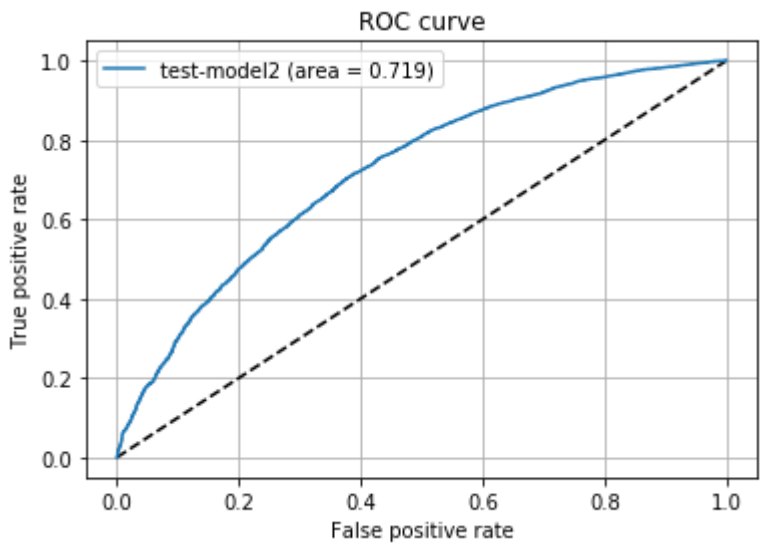
```
from keras.optimizers import Adam
model.compile(
    # Technical note: when using embedding layers, I highly recommend using one of the optimizers
    # found  in tf.train: https://www.tensorflow.org/api_guides/python/train#Optimizers
    # Passing in a string like 'adam' or 'SGD' will load one of keras's optimizers (found under
    # tf.keras.optimizers). They seem to be much slower on problems like this, because they
    # don't efficiently handle sparse gradient updates.
    #tf.train.AdamOptimizer(0.0001),
    optimizer='Adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

```
_category,x_cv.clean_categories,x_cv.clean_subcategories,train_sequences_cv_pad,x_cv.price,x_cv.teacher_number_of_previously_posted_projects
```

```
Train on 76145 samples, validate on 13438 samples
Epoch 1/5
 — Train_Auc is 0.734774 and  val_Auc: 0.712974
76145/76145 - 81s - loss: 0.3978 - accuracy: 0.8484 - val_loss: 0.3890 - val_accuracy: 0.8486
Epoch 2/5
 — Train_Auc is 0.758910 and  val_Auc: 0.716914
76145/76145 - 78s - loss: 0.3800 - accuracy: 0.8498 - val_loss: 0.3863 - val_accuracy: 0.8461
Epoch 3/5
 — Train_Auc is 0.782316 and  val_Auc: 0.724422
76145/76145 - 79s - loss: 0.3709 - accuracy: 0.8535 - val_loss: 0.3902 - val_accuracy: 0.8351
Epoch 4/5
 — Train_Auc is 0.813238 and  val_Auc: 0.722348
76145/76145 - 79s - loss: 0.3564 - accuracy: 0.8588 - val_loss: 0.3837 - val_accuracy: 0.8493
Epoch 5/5
 — Train_Auc is 0.851010 and  val_Auc: 0.710603
76145/76145 - 80s - loss: 0.3398 - accuracy: 0.8649 - val_loss: 0.3944 - val_accuracy: 0.8419
```

```
from sklearn.metrics import roc_curve

import matplotlib.pyplot as plt
y_pred_keras =model.predict([x_test.school_state.to_numpy(),x_test.teacher_prefix.to_numpy(),x_test.project_grade_category.to_numpy(),x_test
fpr_keras, tpr_keras, thresholds_keras = roc_curve(np.argmax(y_test,axis=1), y_pred_keras)
from sklearn.metrics import auc
```

```
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='test-model2 (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.grid()
plt.show()
#.7111

```




```
score=model.evaluate(x=[x_test.school_state.to_numpy(),x_test.teacher_prefix.to_numpy(),x_test.project_grade_category.to_numpy(),x_test.clea
x.add_row(['model2',score[0],score[1],auc_keras])
```

```
19665/19665 [==============================] - 10s 511us/sample - loss: 0.3895 - accuracy: 0.8449
```

```
metrics_tr,metrics=metrics.get_data()
```

```
[0.7129735892166831, 0.7169137566927622, 0.7244224492843278, 0.7223476472095256, 0.7106027962381555]
[0.7347735388158813, 0.75890989453331, 0.782315740477907, 0.8132384142467418, 0.8510098955073249]
```

```python
import datetime
import tensorflow as tf



current_time = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
#train_log_dir = '/gdrive/My Drive/Graph/model2_idf_7_9'
train_log_dir = '/gdrive/My Drive/Graph/model2/train/model2_idf_4_10'
cv_log_dir = '/gdrive/My Drive/Graph/model2/cv/model2_idf_4_10'
test_log_dir = '/gdrive/My Drive/Graph/model2/test/model2_idf_4_10'

train_summary_writer = tf.summary.create_file_writer(train_log_dir)
cv_summary_writer = tf.summary.create_file_writer(cv_log_dir)
test_summary_writer = tf.summary.create_file_writer(test_log_dir)

mye=len(history.history['accuracy'])
for epoch in range(len(history.history['accuracy'])):
  with train_summary_writer.as_default():
        tf.summary.scalar('loss', history.history['loss'][epoch],step=epoch)
        tf.summary.scalar('accuracy', history.history['accuracy'][epoch],step=epoch )
        tf.summary.scalar('AUC', metrics_tr[epoch],step=epoch)
  with cv_summary_writer.as_default():
        tf.summary.scalar('loss', history.history['val_loss'][epoch],step=epoch)
        tf.summary.scalar('accuracy', history.history['val_accuracy'][epoch],step=epoch )
        tf.summary.scalar('AUC', metrics[epoch],step=epoch)
  if(epoch==mye-1):
      with test_summary_writer.as_default():
          tf.summary.scalar('loss', score[0],step=mye)
          tf.summary.scalar('accuracy', score[1],step=mye )
          tf.summary.scalar('AUC', auc_keras,step=mye )

          print('here')

train_summary_writer.close()
```

```
here
```

```python
%load_ext tensorboard
%tensorboard --logdir '/gdrive/My Drive/Graph/model2'
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

TensorBoard SCALARS INACTIVE

model 3

```
data=pd.read_csv('/gdrive/My Drive/preprocessed_data.csv')
y=data['project_is_approved']
#data['price']=data.price.apply(lambda x:round(x))
data.drop(['project_is_approved'],axis=1,inplace=True)
x_train, x_test, y_train, y_test=train_test_split(data,y,test_size=0.2,stratify=y ,random_state=42)
x_train, x_cv, y_train, y_cv=train_test_split(x_train,y_train,test_size=0.16,stratify=y_train ,random_state=42)

x_train.head()
```

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	clean_categories	clean_subcat
71197	nc	mrs	grades_9_12	1	history_civics music_arts	history_ge v
75022	nj	mrs	grades_3_5	34	literacy_language	literature
96510	il	ms	grades_3_5	0	literacy_language	es
84360	ny	ms	grades_prek_2	17	literacy_language specialneeds	literacy spec
16611	nc	ms	grades_9_12	1	history_civics	history_ge

```
from sklearn.preprocessing import OneHotEncoder
# school variable
encoder=OneHotEncoder(handle_unknown='ignore')
x_tr_ss=encoder.fit_transform(x_train.iloc[:,0].values.reshape(-1,1))
x_cv_ss=encoder.transform(x_cv.iloc[:,0].values.reshape(-1,1))
x_te_ss=encoder.transform(x_test.iloc[:,0].values.reshape(-1,1))

# teacher prefix variable
encoder=OneHotEncoder(handle_unknown='ignore')
x_tr_tp=encoder.fit_transform(x_train.iloc[:,1].values.reshape(-1,1))
x_cv_tp=encoder.transform(x_cv.iloc[:,1].values.reshape(-1,1))
x_te_tp=encoder.transform(x_test.iloc[:,1].values.reshape(-1,1))
```

```
#grade variable
encoder=OneHotEncoder(handle_unknown='ignore')
x_tr_grade=encoder.fit_transform(x_train.iloc[:,2].values.reshape(-1,1))
x_cv_grade=encoder.transform(x_cv.iloc[:,2].values.reshape(-1,1))
x_te_grade=encoder.transform(x_test.iloc[:,2].values.reshape(-1,1))


# cc variable
encoder=OneHotEncoder(handle_unknown='ignore')
x_tr_cc=encoder.fit_transform(x_train.iloc[:,4].values.reshape(-1,1))
x_cv_cc=encoder.transform(x_cv.iloc[:,4].values.reshape(-1,1))
x_te_cc=encoder.transform(x_test.iloc[:,4].values.reshape(-1,1))

# csc variable
encoder=OneHotEncoder(handle_unknown='ignore')
x_tr_csc=encoder.fit_transform(x_train.iloc[:,5].values.reshape(-1,1))
x_cv_csc=encoder.transform(x_cv.iloc[:,5].values.reshape(-1,1))
x_te_csc=encoder.transform(x_test.iloc[:,5].values.reshape(-1,1))
```

```
# dont run above code
from sklearn.preprocessing import MinMaxScaler
v=MinMaxScaler()
x_train['price']=v.fit_transform(x_train.price.values.reshape(-1,1))
x_cv['price']=v.transform(x_cv.price.values.reshape(-1,1))
x_test['price']=v.transform(x_test.price.values.reshape(-1,1))


v=MinMaxScaler()
x_train['teacher_number_of_previously_posted_projects']=v.fit_transform(x_train.teacher_number_of_previously_posted_projects.values.reshape(
x_cv['teacher_number_of_previously_posted_projects']=v.transform(x_cv.teacher_number_of_previously_posted_projects.values.reshape(-1,1))
x_test['teacher_number_of_previously_posted_projects']=v.transform(x_test.teacher_number_of_previously_posted_projects.values.reshape(-1,1))
```

```
x_tr_ss.shape
```

```
x_tr_ss.shape
pd.DataFrame(x_train['price']).shape

```

```
(73414, 1)
```

```
from scipy.sparse import hstack
x_trr=hstack((x_tr_ss,x_tr_tp,x_tr_grade,x_tr_cc,x_tr_csc,pd.DataFrame(x_train['price']),pd.DataFrame(x_train.teacher_number_of_previously_p
x_cvv=hstack((x_cv_ss,x_cv_tp,x_cv_grade,x_cv_cc,x_cv_csc,pd.DataFrame(x_cv.price),pd.DataFrame(x_cv.teacher_number_of_previously_posted_pro
x_tee=hstack((x_te_ss,x_te_tp,x_te_grade,x_te_cc,x_te_csc,pd.DataFrame(x_test.price),pd.DataFrame(x_test.teacher_number_of_previously_posted
```

```
x_trr = np.expand_dims(x_trr.toarray(), axis=2)
x_cvv = np.expand_dims(x_cvv.toarray(), axis=2)
x_tee = np.expand_dims(x_tee.toarray(), axis=2)
```

```

kernel_size=(3)
grade=keras.Input(shape=(507,1),name='other_text')

cd=keras.layers.Conv1D(128,kernel_size,padding='valid',activation='relu',strides=1)(grade)
cd=keras.layers.Conv1D(64,kernel_size,padding='valid',activation='relu',strides=1)(cd)
cd=keras.layers.Flatten()(cd)
```

```
tokenizer = text.Tokenizer(num_words=200000)
tokenizer.fit_on_texts(x_train.essay)
train_sequences_tr = tokenizer.texts_to_sequences(x_train.essay)
train_sequences_cv = tokenizer.texts_to_sequences(x_cv.essay)
train_sequences_te = tokenizer.texts_to_sequences(x_test.essay)
from tensorflow.keras.preprocessing.sequence import pad_sequences


train_sequences_tr_pad = pad_sequences(train_sequences_tr, maxlen=max_rev_len)
train_sequences_cv_pad = pad_sequences(train_sequences_cv, maxlen=max_rev_len)
train_sequences_te_pad = pad_sequences(train_sequences_te, maxlen=max_rev_len)


```

```
embedding_layer_text = Embedding(len(word_index) + 1,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix1],
                            input_length=max_rev_len,
                            trainable=False)

sequence_input = keras.Input(shape=(max_rev_len))
emb_text=embedding_layer_text(sequence_input)
merged=LSTM(64,return_sequences=True )(emb_text)
merged_text=LSTM(32,return_sequences=False)(merged)
merged_text=keras.layers.Flatten()(merged_text)


```

```
#def auc(y_true, y_pred):
#    auc = tf.metrics.auc(y_true, y_pred)[1]
#    K.get_session().run(tf.local_variables_initializer())
#    return auc




##############################3


l=[cd,merged_text]
concatenated=keras.layers.Concatenate()(l)



a=Dense(512,activation='relu')(concatenated)
model=Dropout(.4)(a)
model=Dense(256,activation='relu')(model)
model=Dropout(.4)(model)
model=Dense(128,activation='relu')(model)
model=Dropout(.4)(model)
model=Dense(64,activation='relu')(model)
model=Dropout(.4)(model)
```

```
out=Dense(2,activation='softmax')(model)

#y_train= keras.utils.to_categorical(y_train)

model = keras.Model(inputs=[grade,sequence_input],outputs=out)
model.summary()

```

```
Model: "model_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_5 (InputLayer)            [(None, 331)]        0
__________________________________________________________________________________________________
other_text (InputLayer)         [(None, 507, 1)]     0
__________________________________________________________________________________________________
embedding_29 (Embedding)        (None, 331, 300)     14662800    input_5[0][0]
__________________________________________________________________________________________________
conv1d (Conv1D)                 (None, 505, 128)     512         other_text[0][0]
__________________________________________________________________________________________________
lstm_8 (LSTM)                   (None, 331, 64)      93440       embedding_29[0][0]
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 503, 64)      24640       conv1d[0][0]
__________________________________________________________________________________________________
lstm_9 (LSTM)                   (None, 32)           12416       lstm_8[0][0]
__________________________________________________________________________________________________
flatten_25 (Flatten)            (None, 32192)        0           conv1d_1[0][0]
__________________________________________________________________________________________________
flatten_26 (Flatten)            (None, 32)           0           lstm_9[0][0]
__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 32224)        0           flatten_25[0][0]
                                                                 flatten_26[0][0]
__________________________________________________________________________________________________
dense_16 (Dense)                (None, 512)          16499200    concatenate_5[0][0]
__________________________________________________________________________________________________
dropout_10 (Dropout)            (None, 512)          0           dense_16[0][0]
__________________________________________________________________________________________________
dense_17 (Dense)                (None, 256)          131328      dropout_10[0][0]
__________________________________________________________________________________________________
dropout_11 (Dropout)            (None, 256)          0           dense_17[0][0]
__________________________________________________________________________________________________
dense_18 (Dense)                (None, 128)          32896       dropout_11[0][0]
__________________________________________________________________________________________________
dropout_12 (Dropout)            (None, 128)          0           dense_18[0][0]
__________________________________________________________________________________________________
dense_19 (Dense)                (None, 64)           8256        dropout_12[0][0]
__________________________________________________________________________________________________
dropout_13 (Dropout)            (None, 64)           0           dense_19[0][0]
__________________________________________________________________________________________________
dense_20 (Dense)                (None, 2)            130         dropout_13[0][0]
==================================================================================================
Total params: 31,465,618
```

```
Trainable params: 16,802,818
Non-trainable params: 14,662,800
_________________________________________________________________________________________________
```

```
from tensorflow.keras.optimizers import Adam
model.compile(
    # Technical note: when using embedding layers, I highly recommend using one of the optimizers
    # found  in tf.train: https://www.tensorflow.org/api_guides/python/train#Optimizers
    # Passing in a string like 'adam' or 'SGD' will load one of keras's optimizers (found under
    # tf.keras.optimizers). They seem to be much slower on problems like this, because they
    # don't efficiently handle sparse gradient updates.
    #tf.train.AdamOptimizer(0.0001),
    optimizer='Adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

```
from sklearn.utils import class_weight
class_weights = class_weight.compute_class_weight('balanced',
                                                  np.unique(y_train),
                                                  y_train)
y_train= keras.utils.to_categorical(y_train)
y_cv= keras.utils.to_categorical(y_cv)
y_test= keras.utils.to_categorical(y_test)
```

```

```

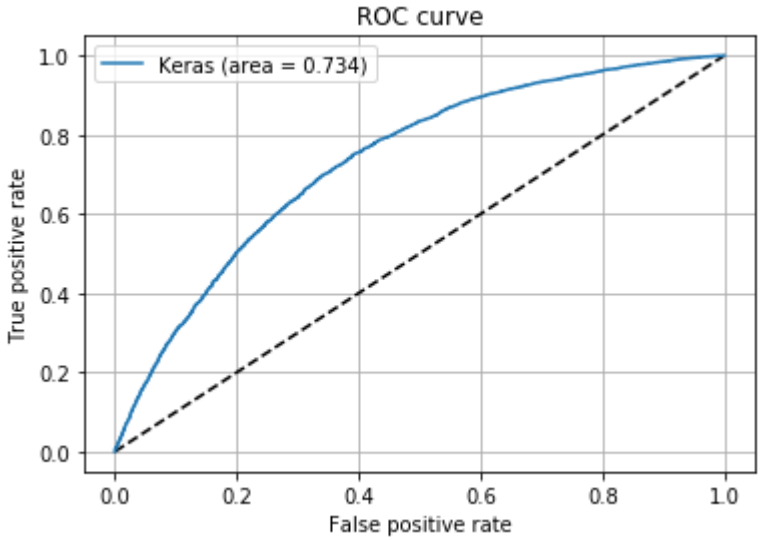
```
cl=[]
tl=[]

metrics = Metrics([x_trr,train_sequences_tr_pad],y_train,[x_cvv,train_sequences_cv_pad],y_cv,cl,tl)
history = model.fit(
    x=[x_trr,train_sequences_tr_pad ]  ,  y=y_train,
    batch_size=128,
    epochs=5,
    verbose=2,
    callbacks=[metrics],
```

```
    validation_data=([x_cvv,train_sequences_cv_pad],y_cv)
);

```

```
Train on 73414 samples, validate on 13984 samples
Epoch 1/5
 — Train_Auc is 0.682227 and  val_Auc: 0.667464
73414/73414 - 76s - loss: 0.4295 - accuracy: 0.8481 - val_loss: 0.4024 - val_accuracy: 0.8486
Epoch 2/5
 — Train_Auc is 0.728741 and  val_Auc: 0.698811
73414/73414 - 71s - loss: 0.4033 - accuracy: 0.8484 - val_loss: 0.3916 - val_accuracy: 0.8486
Epoch 3/5
 — Train_Auc is 0.760131 and  val_Auc: 0.711587
73414/73414 - 71s - loss: 0.3884 - accuracy: 0.8484 - val_loss: 0.3972 - val_accuracy: 0.8486
Epoch 4/5
 — Train_Auc is 0.787450 and  val_Auc: 0.717379
73414/73414 - 71s - loss: 0.3767 - accuracy: 0.8486 - val_loss: 0.3842 - val_accuracy: 0.8486
Epoch 5/5
 — Train_Auc is 0.813167 and  val_Auc: 0.714665
73414/73414 - 71s - loss: 0.3628 - accuracy: 0.8509 - val_loss: 0.3864 - val_accuracy: 0.8497
```

```

```

```
from sklearn.metrics import roc_curve

import matplotlib.pyplot as plt
y_pred_keras =model.predict([x_tee,train_sequences_te_pad])[:,1]
fpr_keras, tpr_keras, thresholds_keras = roc_curve(np.argmax(y_test,axis=1), y_pred_keras)
from sklearn.metrics import auc
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.grid()
plt.show()
#.7111

```




```
score=model.evaluate([x_tee,train_sequences_te_pad],y_test)
x.add_row(['model3',score[0],score[1] ,auc_keras])
```

```
21850/21850 [==============================] - 10s 470us/sample - loss: 0.3768 - accuracy: 0.8516
```

```
metrics_tr,metrics=metrics.get_data()
```

```
[0.6674636765960502, 0.6988112300720484, 0.7115868009471532, 0.7173785554818144, 0.7146645674012782]
[0.6822266477207454, 0.7287407392527472, 0.7601312093071658, 0.7874499890982783, 0.8131667245751216]
```

```
metrics
```

```
[0.6674636765960502,
 0.6988112300720484,
 0.7115868009471532,
 0.7173785554818144,
 0.7146645674012782]
```

```
metrics_tr
```

```
[0.6822266477207454,
 0.7287407392527472,
 0.7601312093071658,
 0.7874499890982783,
 0.8131667245751216]
```

```
import datetime
import tensorflow as tf



current_time = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
#train_log_dir = '/gdrive/My Drive/Graph/model2_idf_7_9'
train_log_dir = '/gdrive/My Drive/Graph/model3/train'
cv_log_dir = '/gdrive/My Drive/Graph/model3/cv'
test_log_dir = '/gdrive/My Drive/Graph/model3/test'

train_summary_writer = tf.summary.create_file_writer(train_log_dir)
cv_summary_writer = tf.summary.create_file_writer(cv_log_dir)
test_summary_writer = tf.summary.create_file_writer(test_log_dir)

mye=len(history.history['accuracy'])
for epoch in range(len(history.history['accuracy'])):
  with train_summary_writer.as_default():
        tf.summary.scalar('loss', history.history['loss'][epoch],step=epoch)
        tf.summary.scalar('accuracy', history.history['accuracy'][epoch],step=epoch )
        tf.summary.scalar('AUC', metrics_tr[epoch],step=epoch)
  with cv_summary_writer.as_default():
        tf.summary.scalar('loss', history.history['val_loss'][epoch],step=epoch)
        tf.summary.scalar('accuracy', history.history['val_accuracy'][epoch],step=epoch )
        tf.summary.scalar('AUC', metrics[epoch],step=epoch)
  if(epoch==mye-1):
      with test_summary_writer.as_default():
          tf.summary.scalar('loss', score[0],step=mye)
          tf.summary.scalar('accuracy', score[1],step=mye )
          tf.summary.scalar('AUC', auc_keras,step=mye )

          print('here')
```

```
train_summary_writer.close()
```

```
here
```

```
%load_ext tensorboard
%tensorboard --logdir '/gdrive/My Drive/Graph/model3'
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

TensorBoard SCALARS INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

cv

test

train

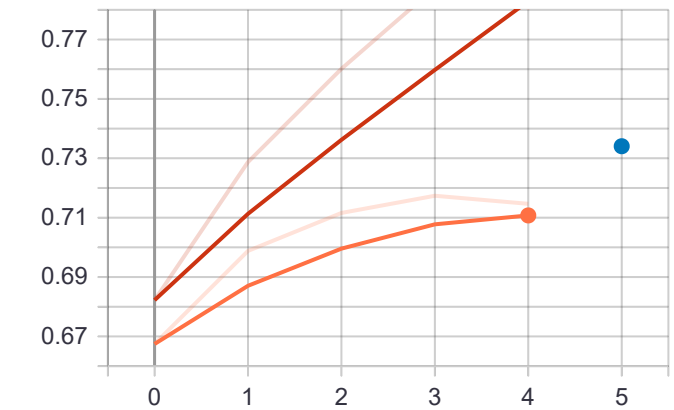
TOGGLE ALL RUNS

/gdrive/My Drive/Graph/model3

Filter tags (regular expressions supported)

AUC

AUC
tag: AUC



accuracy

loss

```
print(x)
```

```
+--------+---------------------+------------+--------------------+
| model  |         loss        |  accuracy  |        auc         |
+--------+---------------------+------------+--------------------+
| model1 | 0.36682035597929236 | 0.8513094  | 0.7670107747258887 |
| model2 | 0.38954551600937076 | 0.84485126 | 0.7193023503937688 |
| model3 | 0.37682461281935736 | 0.8516247  | 0.7340591727633737 |
+--------+---------------------+------------+--------------------+
```