#LSTM - Assignment on Donor Choose Datataset

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from tensorflow.keras.layers import (LSTM, Input, Embedding,Dense,
Flatten, Concatenate,Dropout)
import datetime
from tensorflow.keras.callbacks import Callback, EarlyStopping,
ModelCheckpoint,LearningRateScheduler
from tensorflow.keras import optimizers
import pickle
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow as tf
from tensorflow.keras.regularizers import l2
from keras.utils import np_utils
from tensorflow.keras.layers import Conv1D

from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
data=pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/preprocessed_data.csv")
data.head()
```

```
  school_state teacher_prefix project_grade_category  \
0           ca            mrs          grades_prek_2
1           ut             ms             grades_3_5
2           ca            mrs          grades_prek_2
3           ga            mrs          grades_prek_2
4           wa            mrs             grades_3_5

   teacher_number_of_previously_posted_projects
project_is_approved  \
0                                            53                    1

1                                             4                    1

2                                            10                    1

3                                             2                    1

4                                             2                    1
```

```
    clean_categories                clean_subcategories  \
0       math_science  appliedsciences health_lifescience
1       specialneeds                         specialneeds
2  literacy_language                             literacy
3    appliedlearning                   earlydevelopment
4  literacy_language                             literacy

                                               essay   price
0  i fortunate enough use fairy tale stem kits cl...  725.05
1  imagine 8 9 years old you third grade classroo...  213.03
2  having class 24 students comes diverse learner...  329.00
3  i recently read article giving students choice...  481.04
4  my students crave challenge eat obstacles brea...   17.74
```

```
data.shape
```

```
(109248, 9)
```

```
data.dropna(inplace=True)
```

```
y=data["project_is_approved"]
X=data.drop("project_is_approved",axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,
test_size=0.33, stratify=y_train)
```

```
print(X_train.shape,X_test.shape)
print(y_train.shape,y_test.shape)
print(X_cv.shape,y_cv.shape)
```

```
(49041, 8) (36052, 8)
(49041,) (36052,)
(24155, 8) (24155,)
```

#1. Model_1

Essay Feature Vectorization

```
# split paragraphs and sentences into smaller units using Tokenizer
t = Tokenizer()
t.fit_on_texts(X_train['essay'].values)
vocab_size = len(t.word_index) + 1

X_tr_pre_seq_essay = t.texts_to_sequences(X_train['essay'].values)
X_ts_pre_seq_essay = t.texts_to_sequences(X_test['essay'].values)
X_cv_pre_seq_essay = t.texts_to_sequences(X_cv['essay'].values)

# Ensuring that all sequences in a list have the same length using
```

```
pad_sequences
max_length = 800
X_tr_pad_essay = pad_sequences(X_tr_pre_seq_essay,maxlen=max_length)
X_ts_pad_essay = pad_sequences(X_ts_pre_seq_essay,maxlen=max_length)
X_cv_pad_essay = pad_sequences(X_cv_pre_seq_essay,maxlen=max_length)

X_tr_pad_essay.shape
X_ts_pad_essay.shape
X_cv_pad_essay.shape
```

```
(24155, 800)
```

Embedding Essay

```
with open("/content/drive/MyDrive/Colab Notebooks/glove_vectors",
"rb") as f:
  model = pickle.load(f)
  glove_words = set(model.keys())

vocab_size = len(t.word_index) + 1
print(len(glove_words))
```

```
51510
```

```
tokenizer=t
glove_v = open("/content/drive/MyDrive/Colab
Notebooks/glove_vectors","rb")
total_g_w = pickle.load(glove_v)

tf.keras.backend.clear_session()
import numpy as np

embedding_matrix = np.zeros((vocab_size, 300))
for word, i in t.word_index.items():
      embedding_vector = total_g_w.get(word)
      if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector

essay_input = Input(shape=(800,), name="essay")
featurized_essay = Embedding(vocab_size, 300,
weights=[embedding_matrix], input_length=4, trainable=False)
(essay_input)
featurized_essay = LSTM(128, kernel_initializer='he_normal',dropout =
0.4,return_sequences = True)(featurized_essay)
featurized_essay = Flatten()(featurized_essay)

embedding_matrix.shape
```

```
(41370, 300)
```

All Categorical Feature

School State Feature

```
# split paragraphs and sentences into smaller units using Tokenizer
t2 = Tokenizer()
t2.fit_on_texts(X_train['school_state'].values)
vocab_size = len(t2.word_index) + 1

X_train_school_state_pre_seq =
t2.texts_to_sequences(X_train['school_state'])
X_test_school_state_pre_seq =
t2.texts_to_sequences(X_test['school_state'])
X_cv_school_state_pre_seq =
t2.texts_to_sequences(X_cv['school_state'])

# Ensuring that all sequences in a list have the same length using
pad_sequences
max_length = 1
X_train_school_state_pad_seq =
pad_sequences(X_train_school_state_pre_seq,maxlen=max_length)
X_test_school_state_pad_seq =
pad_sequences(X_test_school_state_pre_seq,maxlen=max_length)
X_cv_school_state_pad_seq =
pad_sequences(X_cv_school_state_pre_seq,maxlen=max_length)

#print(vocab_size)
Unique_school_state=X_train['school_state'].nunique()
embedding_size = int(np.ceil((Unique_school_state) / 2))
school_state_input = Input(shape=(1,), name="school_state")
school_state_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size,trainable = True)(school_state_input)
#school_state_feature = Embedding(input_dim = vocab_size+1, output_dim
= 2, input_length = 1)(school_state_input)
school_state_feature = Flatten()(school_state_feature)

X_train_school_state_pad_seq.shape
```

```
(49041, 1)
```

Project Grade Category

```
t3 = Tokenizer()
t3.fit_on_texts(X_train['project_grade_category'].values)
vocab_size = len(t3.word_index) + 1

#print(vocab_size)

X_train_project_grade_category_pre_seq =
t3.texts_to_sequences(X_train['project_grade_category'])
X_test_project_grade_category_pre_seq =
t3.texts_to_sequences(X_test['project_grade_category'])
```

```
X_cv_project_grade_category_pre_seq =
t3.texts_to_sequences(X_cv['project_grade_category'])

# Ensuring that all sequences in a list have the same length using
pad_sequences
max_length = 5
X_train_project_grade_category_pad_seq =
pad_sequences(X_train_project_grade_category_pre_seq,maxlen=max_length
)
X_test_project_grade_category_pad_seq =
pad_sequences(X_test_project_grade_category_pre_seq,maxlen=max_length)
X_cv_project_grade_category_pad_seq =
pad_sequences(X_cv_project_grade_category_pre_seq,maxlen=max_length)

Unique_school_state=X_train['project_grade_category'].nunique()
embedding_size = int(np.ceil((Unique_school_state) / 2))

project_grade_category_input = Input(shape=(5,),
name="project_grade_category")
project_grade_category_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size,trainable = True)
(project_grade_category_input)
project_grade_category_feature = Flatten()
(project_grade_category_feature)

X_train_project_grade_category_pad_seq.shape
```

```
(49041, 5)
```

Clean Category Feature

```
t4 = Tokenizer()
t4.fit_on_texts(X_train['clean_categories'].values)
vocab_size = len(t4.word_index) + 1

#print(vocab_size)

X_train_clean_categories_pre_seq =
t4.texts_to_sequences(X_train['clean_categories'])
X_test_clean_categories_pre_seq =
t4.texts_to_sequences(X_test['clean_categories'])
X_cv_clean_categories_pre_seq =
t4.texts_to_sequences(X_cv['clean_categories'])

# Ensuring that all sequences in a list have the same length using
pad_sequences
max_length = 5
X_train_clean_categories_pad_seq =
pad_sequences(X_train_clean_categories_pre_seq,maxlen=max_length)
X_test_clean_categories_pad_seq =
pad_sequences(X_test_clean_categories_pre_seq,maxlen=max_length)
```

```
X_cv_clean_categories_pad_seq =
pad_sequences(X_cv_clean_categories_pre_seq,maxlen=max_length)

Unique_school_state=X_train['clean_categories'].nunique()
embedding_size = int(np.ceil((Unique_school_state) / 2))

project_subject_categories_input = Input(shape=(5,),
name="clean_categories")
project_subject_categories_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size,trainable = True)
(project_subject_categories_input)
project_subject_categories_feature = Flatten()
(project_subject_categories_feature)

X_train_clean_categories_pad_seq.shape
```

```
(49041, 5)
```

Clean Sub Category Feature

```
t5 = Tokenizer()
t5.fit_on_texts(X_train['clean_subcategories'].values)
vocab_size = len(t5.word_index) + 1

#print(vocab_size)

X_train_clean_subcategories_pre_seq =
t5.texts_to_sequences(X_train['clean_subcategories'])
X_test_clean_subcategories_pre_seq =
t5.texts_to_sequences(X_test['clean_subcategories'])
X_cv_clean_subcategories_pre_seq =
t5.texts_to_sequences(X_cv['clean_subcategories'])

# Ensuring that all sequences in a list have the same length using
pad_sequences
max_length = 5
X_train_clean_subcategories_pad_seq =
pad_sequences(X_train_clean_subcategories_pre_seq,maxlen=max_length)
X_test_clean_subcategories_pad_seq =
pad_sequences(X_test_clean_subcategories_pre_seq,maxlen=max_length)
X_cv_clean_subcategories_pad_seq =
pad_sequences(X_cv_clean_subcategories_pre_seq,maxlen=max_length)

Unique_school_state=X_train['clean_subcategories'].nunique()
embedding_size = int(np.ceil((Unique_school_state) / 2))

project_subject_subcategories_input = Input(shape=(5,),
name="clean_subcategories")
project_subject_subcategories_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size,trainable = True)
(project_subject_subcategories_input)
```

```
project_subject_subcategories_feature = Flatten()
(project_subject_subcategories_feature)

X_train_clean_subcategories_pad_seq.shape
```

```
(49041, 5)
```

Teacher Prefix Feature

```
t6 = Tokenizer()
t6.fit_on_texts(X_train['teacher_prefix'].values)

vocab_size = len(t6.word_index) + 1

#print(vocab_size)

X_train_teacher_prefix_pre_seq =
t6.texts_to_sequences(X_train['teacher_prefix'])
X_test_teacher_prefix_pre_seq =
t6.texts_to_sequences(X_test['teacher_prefix'])
X_cv_teacher_prefix_pre_seq =
t6.texts_to_sequences(X_cv['teacher_prefix'])

# Ensuring that all sequences in a list have the same length using
pad_sequences
max_length = 1
X_train_teacher_prefix_pad_seq =
pad_sequences(X_train_teacher_prefix_pre_seq,maxlen=max_length)
X_test_teacher_prefix_pad_seq =
pad_sequences(X_test_teacher_prefix_pre_seq,maxlen=max_length)
X_cv_teacher_prefix_pad_seq =
pad_sequences(X_cv_teacher_prefix_pre_seq,maxlen=max_length)

Unique_school_state=X_train['teacher_prefix'].nunique()
embedding_size = int(np.ceil((Unique_school_state) / 2))

teacher_prefix_input = Input(shape=(1,), name="teacher_prefix")
teacher_prefix_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size,trainable = True)(teacher_prefix_input)
teacher_prefix_feature = Flatten()(teacher_prefix_feature)

X_train_teacher_prefix_pad_seq.shape
```

```
(49041, 1)
```

Numerical Feature : Teacher_number_of_previously_posted_projects and Price

```
from sklearn.preprocessing import StandardScaler

teacher_n_scalar = StandardScaler()
teacher_n_scalar.fit(X_train['teacher_number_of_previously_posted_proj
ects'].values.reshape(-1,1))
```

```
X_train_teacher_number_of_previously_posted_project =
teacher_n_scalar.transform(X_train['teacher_number_of_previously_poste
d_projects'].values.reshape(-1, 1))
X_test_teacher_number_of_previously_posted_project =
teacher_n_scalar.transform(X_test['teacher_number_of_previously_posted
_projects'].values.reshape(-1, 1))
X_cv_teacher_number_of_previously_posted_project =
teacher_n_scalar.transform(X_cv['teacher_number_of_previously_posted_p
rojects'].values.reshape(-1, 1))


price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1))
X_train_price_scalar =
price_scalar.transform(X_train['price'].values.reshape(-1,1))
X_test_price_scalar =
price_scalar.transform(X_test['price'].values.reshape(-1,1))
X_cv_price_scalar =
price_scalar.transform(X_cv['price'].values.reshape(-1,1))

# to cancatenate numeric feature reshaping array
X_train_price_scalar = X_train_price_scalar.reshape(-1,1)
X_test_price_scalar = X_test_price_scalar.reshape(-1,1)
X_cv_price_scalar = X_cv_price_scalar.reshape(-1,1)

X_train_num_teacher_number_of_previously_posted_projects_and_price =
np.hstack((X_train_teacher_number_of_previously_posted_project,X_train
_price_scalar))
X_test_num_teacher_number_of_previously_posted_projects_and_price =
np.hstack((X_test_teacher_number_of_previously_posted_project,X_test_p
rice_scalar))
X_cv_num_teacher_number_of_previously_posted_projects_and_price =
np.hstack((X_cv_teacher_number_of_previously_posted_project,X_cv_price
_scalar))

numerical_input = Input(shape=(2,))
numeric_dense = Dense(128, activation='relu' ,
kernel_initializer='he_normal',kernel_regularizer=l2(0.001))
(numerical_input )

from keras.layers import concatenate

con_data_1 = concatenate([featurized_essay,school_state_feature,

teacher_prefix_feature,project_grade_category_feature,

project_subject_categories_feature,project_subject_subcategories_featu
re,
                          numeric_dense])
```

##MODEL 1

```
from keras.models import Model
from keras import regularizers, initializers


# Layer 1
model1 = Dense(256, activation = 'relu',
kernel_initializer="glorot_normal",
kernel_regularizer = regularizers.l2(0.01))(con_data_1)
model1 = Dropout(0.4)(model1)
# Layer 2
model1= Dense(128, activation =
'relu',kernel_initializer="glorot_normal",
kernel_regularizer = regularizers.l2(0.01))(model1)
model1= Dropout(0.4)(model1)
# Layer 3
model1 = Dense(64, activation = 'relu',kernel_initializer="he_normal",
kernel_regularizer = regularizers.l2(0.01))(model1)
model1 = Dropout(0.4)(model1)
# Output layer
output = Dense(2, activation = 'softmax', name= 'Model_1_output')
(model1)

#Model_1
Model_1 = Model(inputs = [essay_input,teacher_prefix_input,

school_state_input,project_grade_category_input,
                           project_subject_categories_input,
                           project_subject_subcategories_input,
                           numerical_input], outputs = [output])
print(Model_1.summary())
```

```
Model: "model"
__________________________________________________________________________
_____________________________
 Layer (type)                   Output Shape         Param #
Connected to
==========================================================================
============================
 essay (InputLayer)             [(None, 800)]        0           []



 embedding (Embedding)          (None, 800, 300)     12411000
['essay[0][0]']


 school_state (InputLayer)      [(None, 1)]          0           []
```

```
 teacher_prefix (InputLayer)    [(None, 1)]          0           []

 project_grade_category (InputL  [(None, 5)]         0           []
 ayer)

 clean_categories (InputLayer)  [(None, 5)]          0           []

 clean_subcategories (InputLaye  [(None, 5)]         0           []
 r)

 lstm (LSTM)                    (None, 800, 128)     219648
['embedding[0][0]']

 embedding_1 (Embedding)        (None, 1, 26)        1378
['school_state[0][0]']

 embedding_5 (Embedding)        (None, 1, 3)         21
['teacher_prefix[0][0]']

 embedding_2 (Embedding)        (None, 5, 2)         22
['project_grade_category[0][0]']

 embedding_3 (Embedding)        (None, 5, 24)        408
['clean_categories[0][0]']

 embedding_4 (Embedding)        (None, 5, 187)       7293
['clean_subcategories[0][0]']

 input_1 (InputLayer)           [(None, 2)]          0           []
```

```
 flatten (Flatten)              (None, 102400)       0
['lstm[0][0]']


 flatten_1 (Flatten)            (None, 26)           0
['embedding_1[0][0]']


 flatten_5 (Flatten)            (None, 3)            0
['embedding_5[0][0]']


 flatten_2 (Flatten)            (None, 10)           0
['embedding_2[0][0]']


 flatten_3 (Flatten)            (None, 120)          0
['embedding_3[0][0]']


 flatten_4 (Flatten)            (None, 935)          0
['embedding_4[0][0]']


 dense (Dense)                  (None, 128)          384
['input_1[0][0]']


 concatenate (Concatenate)      (None, 103622)       0
['flatten[0][0]',

'flatten_1[0][0]',

'flatten_5[0][0]',

'flatten_2[0][0]',

'flatten_3[0][0]',

'flatten_4[0][0]',

'dense[0][0]']


 dense_1 (Dense)                (None, 256)          26527488
['concatenate[0][0]']
```

```
 dropout (Dropout)              (None, 256)          0
['dense_1[0][0]']


 dense_2 (Dense)                (None, 128)          32896
['dropout[0][0]']


 dropout_1 (Dropout)            (None, 128)          0
['dense_2[0][0]']


 dense_3 (Dense)                (None, 64)           8256
['dropout_1[0][0]']


 dropout_2 (Dropout)            (None, 64)           0
['dense_3[0][0]']


 Model_1_output (Dense)         (None, 2)            130
['dropout_2[0][0]']


==================================================================
============================
Total params: 39,208,924
Trainable params: 26,797,924
Non-trainable params: 12,411,000
__________________________________________________________________
_____________________________
None

#Train data
train_data1 =
[X_tr_pad_essay,X_train_teacher_prefix_pad_seq,X_train_school_state_pa
d_seq,X_train_project_grade_category_pad_seq,X_train_clean_subcategori
es_pad_seq,X_train_clean_categories_pad_seq,X_train_num_teacher_number
_of_previously_posted_projects_and_price]
# Test data
test_data1 =
[X_ts_pad_essay,X_test_teacher_prefix_pad_seq,X_test_school_state_pad_
seq,X_test_project_grade_category_pad_seq,X_test_clean_subcategories_p
ad_seq,X_test_clean_categories_pad_seq,
X_test_num_teacher_number_of_previously_posted_projects_and_price]
# CV data
cv_data1 =
[X_cv_pad_essay,X_cv_teacher_prefix_pad_seq,X_cv_school_state_pad_seq,
X_cv_project_grade_category_pad_seq,X_cv_clean_subcategories_pad_seq,X
```

```
_cv_clean_categories_pad_seq,
X_cv_num_teacher_number_of_previously_posted_projects_and_price]

from sklearn.metrics import roc_auc_score

def aoc_roc(y_true, y_pred):
  return tf.py_function(roc_auc_score, (y_true, y_pred), tf.double)

y_tr_data_1 = np_utils.to_categorical(y_train, 2)
y_te_data_1 = np_utils.to_categorical(y_test, 2)
y_cv_data_1 = np_utils.to_categorical(y_cv, 2)
```

Compiling and fiting model_1

```
from tensorflow.keras.optimizers import RMSprop

Model_1.compile(optimizer = "rmsprop", loss =
'categorical_crossentropy', metrics = [aoc_roc])

%reload_ext tensorboard

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M
%S")
tensorboard=
tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1)

filepath="/content/drive/MyDrive/Colab
Notebooks/weights_copy_1.best.hdf5"

from keras.callbacks import TensorBoard

earlystop_1 = EarlyStopping(monitor='val_loss', patience=2, verbose=1)
checkpoint = ModelCheckpoint(filepath, monitor='val_auroc', verbose=1,
save_best_only=True, mode='max')
callbk_list = [checkpoint,earlystop_1,tensorboard]

history_1 = Model_1.fit(train_data1,y_tr_data_1,batch_size=512,
epochs=15,validation_data=(cv_data1,y_cv_data_1), callbacks =
callbk_list)

Epoch 1/15
96/96 [==============================] - ETA: 0s - loss: 2.9443 -
aoc_roc: 0.5802

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 47s 402ms/step - loss:
2.9443 - aoc_roc: 0.5802 - val_loss: 1.2906 - val_aoc_roc: 0.7033
Epoch 2/15
96/96 [==============================] - ETA: 0s - loss: 0.9321 -
aoc_roc: 0.6752
```

```
WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 392ms/step - loss:
0.9321 - aoc_roc: 0.6752 - val_loss: 0.7059 - val_aoc_roc: 0.7192
Epoch 3/15
96/96 [==============================] - ETA: 0s - loss: 0.6506 -
aoc_roc: 0.6944

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 39s 410ms/step - loss:
0.6506 - aoc_roc: 0.6944 - val_loss: 0.6064 - val_aoc_roc: 0.7284
Epoch 4/15
96/96 [==============================] - ETA: 0s - loss: 0.5844 -
aoc_roc: 0.7022

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 37s 388ms/step - loss:
0.5844 - aoc_roc: 0.7022 - val_loss: 0.5552 - val_aoc_roc: 0.7330
Epoch 5/15
96/96 [==============================] - ETA: 0s - loss: 0.5478 -
aoc_roc: 0.7086

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 399ms/step - loss:
0.5478 - aoc_roc: 0.7086 - val_loss: 0.5205 - val_aoc_roc: 0.7383
Epoch 6/15
96/96 [==============================] - ETA: 0s - loss: 0.5187 -
aoc_roc: 0.7225

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 395ms/step - loss:
0.5187 - aoc_roc: 0.7225 - val_loss: 0.4949 - val_aoc_roc: 0.7427
Epoch 7/15
96/96 [==============================] - ETA: 0s - loss: 0.4976 -
aoc_roc: 0.7228

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 393ms/step - loss:
0.4976 - aoc_roc: 0.7228 - val_loss: 0.4803 - val_aoc_roc: 0.7427
Epoch 8/15
96/96 [==============================] - ETA: 0s - loss: 0.4789 -
aoc_roc: 0.7280
```

```
WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 396ms/step - loss:
0.4789 - aoc_roc: 0.7280 - val_loss: 0.4607 - val_aoc_roc: 0.7501
Epoch 9/15
96/96 [==============================] - ETA: 0s - loss: 0.4627 -
aoc_roc: 0.7272

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 39s 408ms/step - loss:
0.4627 - aoc_roc: 0.7272 - val_loss: 0.4439 - val_aoc_roc: 0.7551
Epoch 10/15
96/96 [==============================] - ETA: 0s - loss: 0.4493 -
aoc_roc: 0.7311

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 402ms/step - loss:
0.4493 - aoc_roc: 0.7311 - val_loss: 0.4383 - val_aoc_roc: 0.7556
Epoch 11/15
96/96 [==============================] - ETA: 0s - loss: 0.4385 -
aoc_roc: 0.7319

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 398ms/step - loss:
0.4385 - aoc_roc: 0.7319 - val_loss: 0.4218 - val_aoc_roc: 0.7551
Epoch 12/15
96/96 [==============================] - ETA: 0s - loss: 0.4271 -
aoc_roc: 0.7369

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 395ms/step - loss:
0.4271 - aoc_roc: 0.7369 - val_loss: 0.4351 - val_aoc_roc: 0.7539
Epoch 13/15
96/96 [==============================] - ETA: 0s - loss: 0.4218 -
aoc_roc: 0.7373

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 38s 398ms/step - loss:
0.4218 - aoc_roc: 0.7373 - val_loss: 0.4098 - val_aoc_roc: 0.7559
Epoch 14/15
96/96 [==============================] - ETA: 0s - loss: 0.4159 -
aoc_roc: 0.7405
```

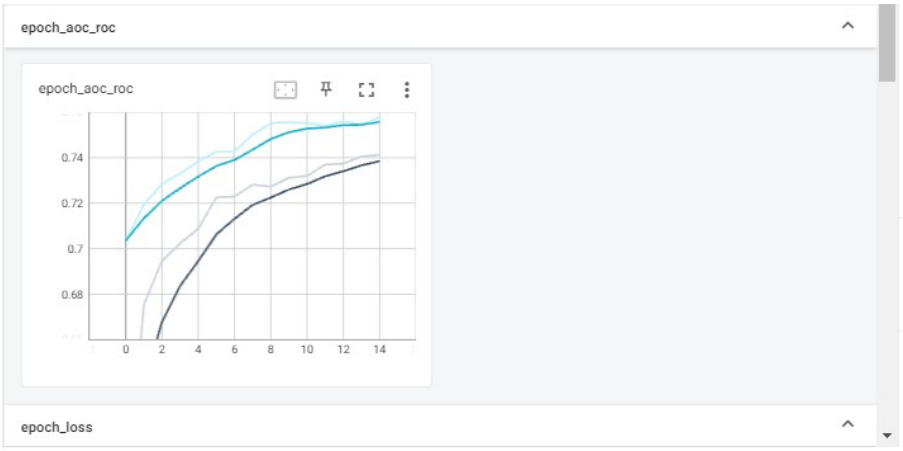
```
WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 40s 418ms/step - loss:
0.4159 - aoc_roc: 0.7405 - val_loss: 0.4078 - val_aoc_roc: 0.7546
Epoch 15/15
96/96 [==============================] - ETA: 0s - loss: 0.4092 -
aoc_roc: 0.7411

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 39s 404ms/step - loss:
0.4092 - aoc_roc: 0.7411 - val_loss: 0.4051 - val_aoc_roc: 0.7576
```

Model_1 - Tensorboard

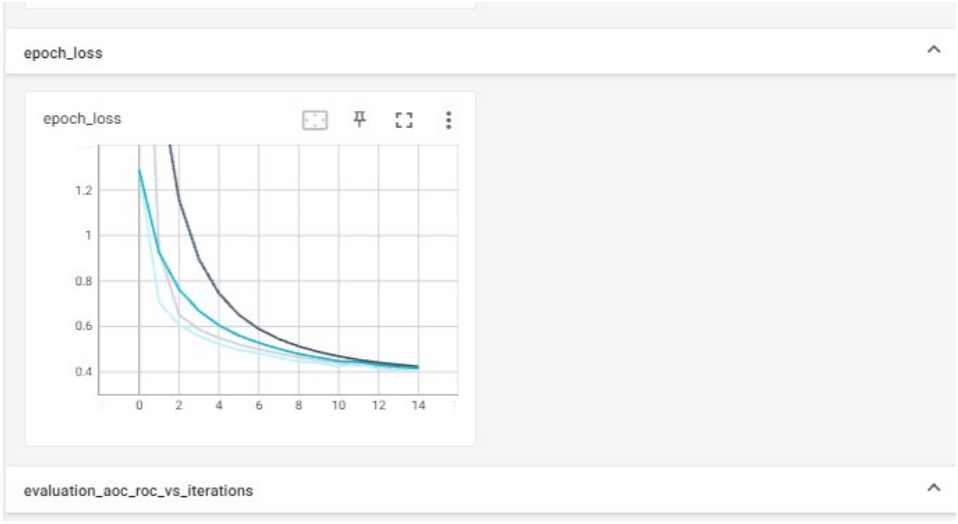
```
%reload_ext tensorboard
%tensorboard --logdir $log_dir
```

```
<IPython.core.display.Javascript object>
```

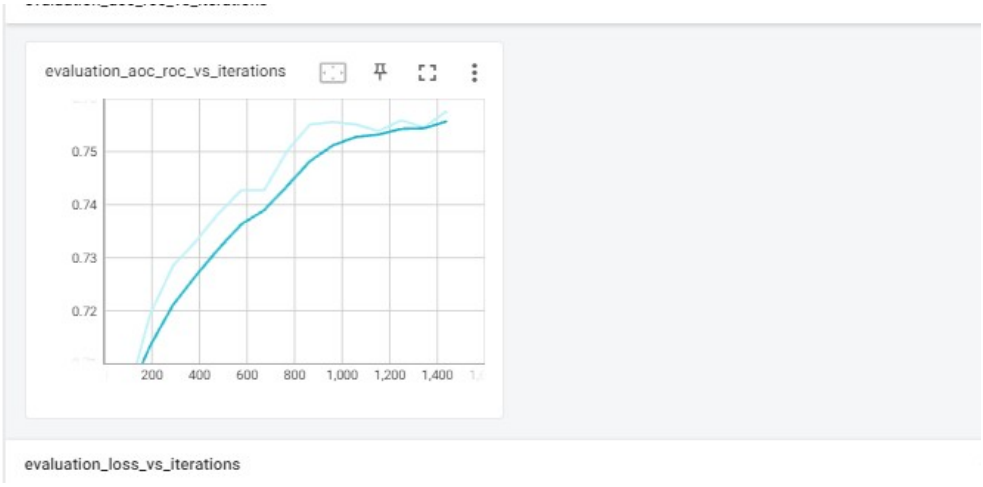
```
import IPython.display as display
from PIL import Image
display.display(Image.open('/content/aa.png'))
```



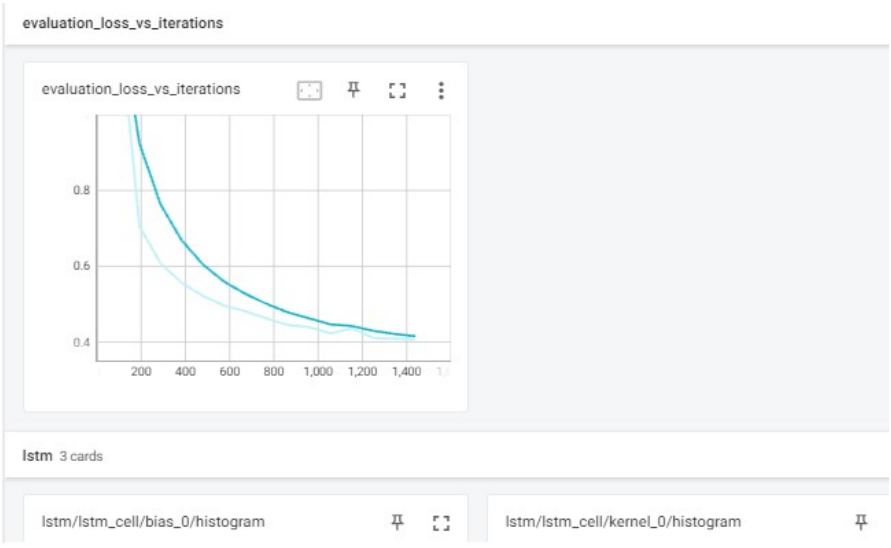

```
display.display(Image.open('/content/aa1.png'))
```



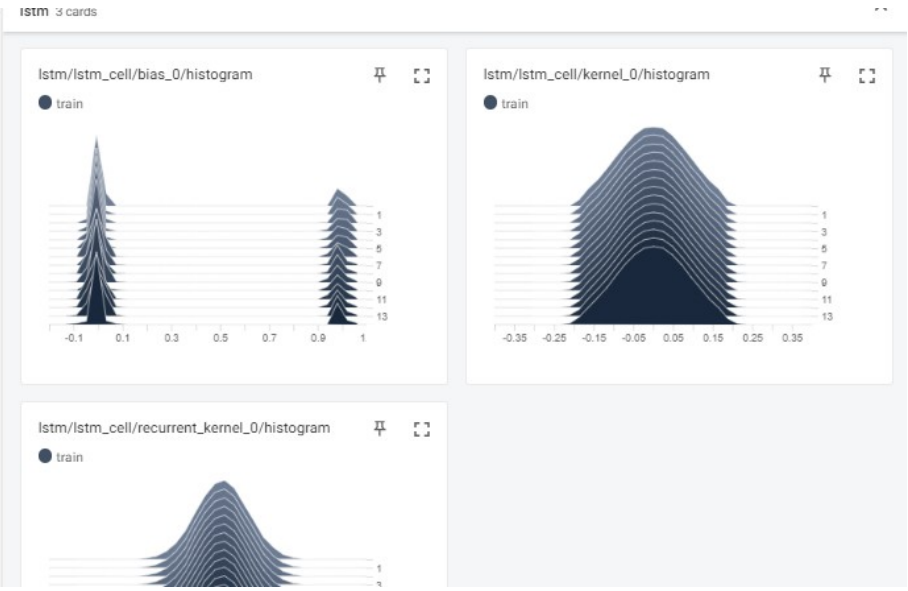

```
display.display(Image.open('/content/aa2.png'))
```



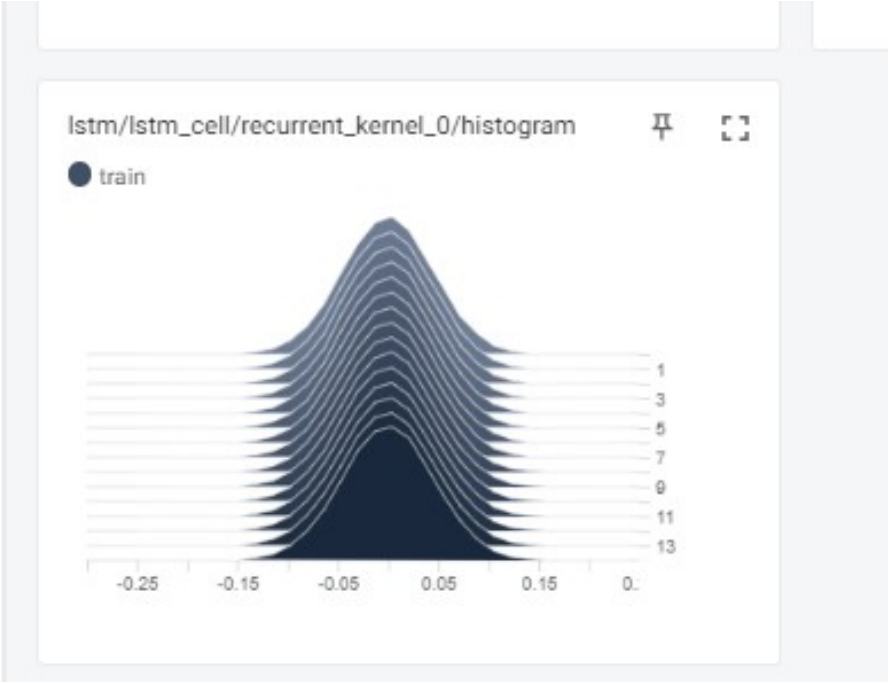

```
display.display(Image.open('/content/aa3.png'))
```



```
display.display(Image.open('/content/aa4.png'))
```




```
display.display(Image.open('/content/aa5.png'))
```

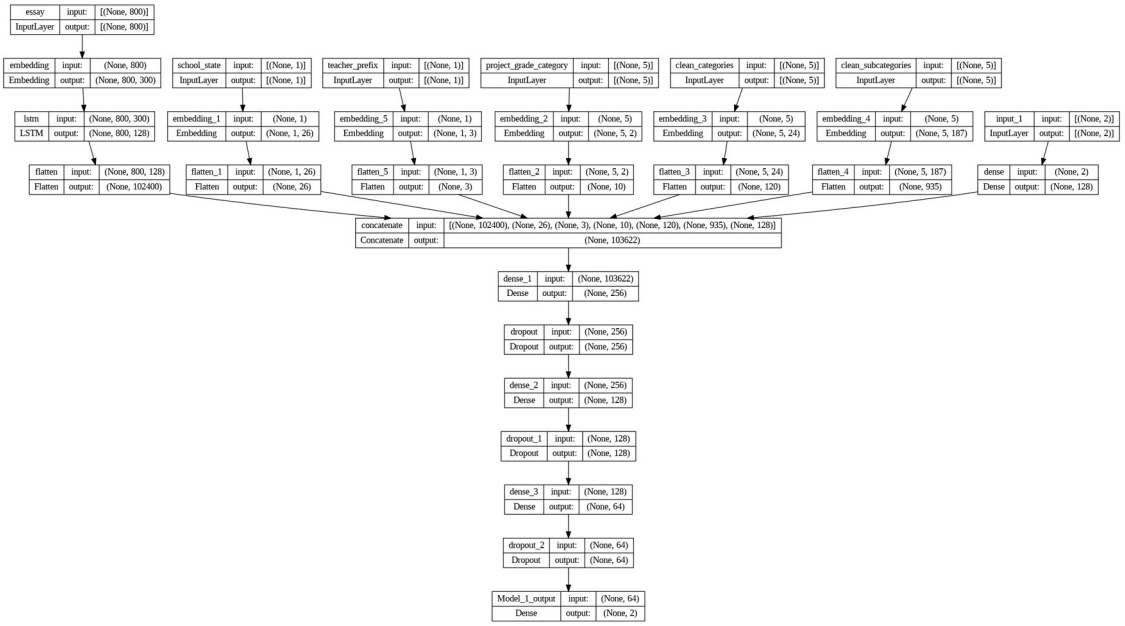



```
scores = Model_1.evaluate(test_data1, y_te_data_1,
verbose=0,batch_size=512)

print("%s: %.2f%%" % (Model_1.metrics_names[1], scores[1]*100))
```

```
aoc_roc: 74.35%
```

```
from keras.utils.vis_utils import plot_model
plot_model(Model_1, show_shapes=True, show_layer_names=True,
```

```
to_file='model_1.png')
from IPython.display import Image
Image(retina=True, filename='model_1.png')
```



#2. Model_2

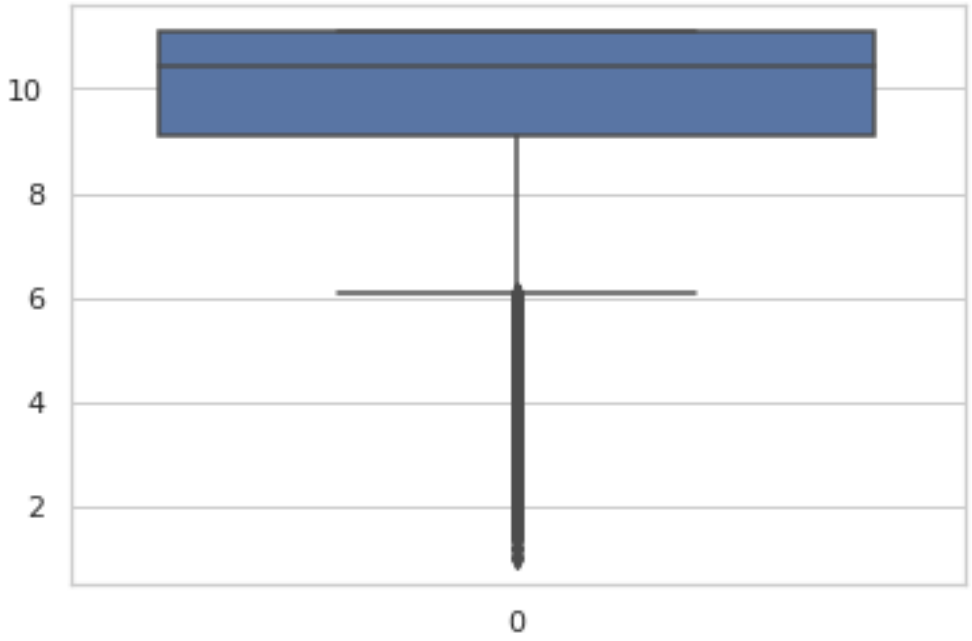
```
!nvidia-smi
```

```
Fri Feb 24 20:03:33 2023
+-----------------------------------------------------------------------------
--------+
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version:
12.0     |
|-------------------------------+----------------------
+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile
Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util
Compute M. |
|                               |                      |
MIG M. |
|
===============================+======================+==============
=======|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |
0 |
| N/A   71C    P0    28W /  70W |   4773MiB / 15360MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------
+----------------------+
```

```
+-----------------------------------------------------------------------------
--------+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name                  GPU
Memory |
|        ID   ID
Usage      |
|
=============================================================================
=======|
+-----------------------------------------------------------------------------
--------+
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
tfidf = vectorizer.fit_transform(X_train["essay"])

dataset = {'feature_name': vectorizer.get_feature_names() ,
'idf_value': vectorizer.idf_}

tfidf_df = pd.DataFrame(data=dataset)

import seaborn as sns
sns.set(style="whitegrid")
sns.boxplot(data=tfidf_df['idf_value'], color='b' )
```

```
<AxesSubplot:>
```



```
tfidf_df
```

```
      feature_name  idf_value
0               00   7.073045
1              000   5.860261
2             000s  11.107285
3              001  11.107285
4              002  11.107285
...            ...        ...
41328        zusak  10.701820
41329        zwink  11.107285
41330       zydeco  11.107285
41331      zynergy  11.107285
41332           zz  11.107285

[41333 rows x 2 columns]
```

```
for i in range(0,101,5):
  print("Percentile of {} is
{}".format(i,np.percentile(tfidf_df['idf_value'] , i)))
```

```
Percentile of 0 is 1.0082308361726968
Percentile of 5 is 5.92831456333733
Percentile of 10 is 7.165703364583109
Percentile of 15 is 7.950284751102687
Percentile of 20 is 8.581556527944546
Percentile of 25 is 9.092382151710535
Percentile of 30 is 9.4978472598187
Percentile of 35 is 9.854522203757432
Percentile of 40 is 10.190994440378645
Percentile of 45 is 10.414137991692854
Percentile of 50 is 10.414137991692854
Percentile of 55 is 10.701820064144636
Percentile of 60 is 10.701820064144636
Percentile of 65 is 11.1072851722528
Percentile of 70 is 11.1072851722528
Percentile of 75 is 11.1072851722528
Percentile of 80 is 11.1072851722528
Percentile of 85 is 11.1072851722528
Percentile of 90 is 11.1072851722528
Percentile of 95 is 11.1072851722528
Percentile of 100 is 11.1072851722528
```

```
print("The idf score having 20th Percentile :",
np.percentile(tfidf_df['idf_value'],[20]))
print("The idf score having 25th Percentile :",
np.percentile(tfidf_df['idf_value'],[25]))
print("The idf score having 65th Percentile :",
np.percentile(tfidf_df['idf_value'],[65]))
```

```
The idf score having 20th Percentile : [8.58155653]
The idf score having 25th Percentile : [9.09238215]
The idf score having 65th Percentile : [11.10728517]
```

```
tfidf_final_essay=tfidf_df[( tfidf_df['idf_value']>=1) &
(tfidf_df['idf_value'] <=11.107)]
```

```
tfidf_final=tfidf_final_essay['feature_name'].tolist()
```

```
print(len(tfidf_final))
```

```
25375
```

##2.1 Text(Essay) Vectorization

```
# Apply tokenizer
t = Tokenizer(num_words=len(tfidf_final))
t = Tokenizer()
t.fit_on_texts(tfidf_final)
t.fit_on_texts(X_train['essay'].values)
vocab_size = len(t.word_index) + 1

X_tr_pre_seq_essay = t.texts_to_sequences(X_train['essay'].values)
X_ts_pre_seq_essay = t.texts_to_sequences(X_test['essay'].values)
X_cv_pre_seq_essay = t.texts_to_sequences(X_cv['essay'].values)
# padd sequence

max_length = 800
X_tr_pad_essay = pad_sequences(X_tr_pre_seq_essay,maxlen=max_length)
X_ts_pad_essay = pad_sequences(X_ts_pre_seq_essay,maxlen=max_length)
X_cv_pad_essay = pad_sequences(X_cv_pre_seq_essay,maxlen=max_length)

vocab_size = len(t.word_index) +1
print((vocab_size))
```

```
41370
```

```
tokenizer=t
```

```
glove_v = open("/content/drive/MyDrive/Colab
Notebooks/glove_vectors","rb")
total_g_w = pickle.load(glove_v)
```

```
import numpy as np

# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size, 300))
for word, i in t.word_index.items():
      embedding_vector = total_g_w.get(word)
      if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector
```

```
tf.keras.backend.clear_session()

essay_input = Input(shape=(800,), name="essay")
featurized_essay = Embedding(vocab_size, 300,
weights=[embedding_matrix], input_length=4, trainable=False)
(essay_input)
featurized_essay = LSTM(100)(featurized_essay)
featurized_essay = Flatten()(featurized_essay)

embedding_matrix.shape

(41370, 300)
```

#categorical Feature

```
t2 = Tokenizer()
t2.fit_on_texts(X_train['school_state'].values)
vocab_size = len(t2.word_index) + 1

X_train_school_state_pre_seq =
t2.texts_to_sequences(X_train['school_state'])
X_test_school_state_pre_seq =
t2.texts_to_sequences(X_test['school_state'])
X_cv_school_state_pre_seq =
t2.texts_to_sequences(X_cv['school_state'])

max_length = 1
X_train_school_state_pad_seq =
pad_sequences(X_train_school_state_pre_seq,maxlen=max_length)
X_test_school_state_pad_seq =
pad_sequences(X_test_school_state_pre_seq,maxlen=max_length)
X_cv_school_state_pad_seq =
pad_sequences(X_cv_school_state_pre_seq,maxlen=max_length)

#print(vocab_size)

embedding_size = int(np.ceil((Unique_school_state) / 2))
school_state_input = Input(shape=(1,), name="school_state")
school_state_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size)(school_state_input)
school_state_feature = Flatten()(school_state_feature)

X_train_school_state_pad_seq.shape

(49041, 1)
```

Project Grade Feature

```
t3 = Tokenizer()
t3.fit_on_texts(X_train['project_grade_category'].values)
vocab_size = len(t3.word_index) + 1
```

```
#print(vocab_size)

X_train_project_grade_category_pre_seq =
t3.texts_to_sequences(X_train['project_grade_category'])
X_test_project_grade_category_pre_seq =
t3.texts_to_sequences(X_test['project_grade_category'])
X_cv_project_grade_category_pre_seq =
t3.texts_to_sequences(X_cv['project_grade_category'])

max_length = 5
X_train_project_grade_category_pad_seq =
pad_sequences(X_train_project_grade_category_pre_seq,maxlen=max_length
)
X_test_project_grade_category_pad_seq =
pad_sequences(X_test_project_grade_category_pre_seq,maxlen=max_length)
X_cv_project_grade_category_pad_seq =
pad_sequences(X_cv_project_grade_category_pre_seq,maxlen=max_length)

embedding_size = int(np.ceil((Unique_school_state) / 2))

project_grade_category_input = Input(shape=(5,),
name="project_grade_category")
project_grade_category_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size)(project_grade_category_input)
project_grade_category_feature = Flatten()
(project_grade_category_feature)

X_train_project_grade_category_pad_seq.shape
```

```
(49041, 5)
```

Clean Categories Feature

```
t4 = Tokenizer()
t4.fit_on_texts(X_train['clean_categories'].values)
vocab_size = len(t4.word_index) + 1

#print(vocab_size)

X_train_clean_categories_pre_seq =
t4.texts_to_sequences(X_train['clean_categories'])
X_test_clean_categories_pre_seq =
t4.texts_to_sequences(X_test['clean_categories'])
X_cv_clean_categories_pre_seq =
t4.texts_to_sequences(X_cv['clean_categories'])

max_length = 5
X_train_clean_categories_pad_seq =
pad_sequences(X_train_clean_categories_pre_seq,maxlen=max_length)
X_test_clean_categories_pad_seq =
pad_sequences(X_test_clean_categories_pre_seq,maxlen=max_length)
```

```
X_cv_clean_categories_pad_seq =
pad_sequences(X_cv_clean_categories_pre_seq,maxlen=max_length)

embedding_size = int(np.ceil((Unique_school_state) / 2))

project_subject_categories_input = Input(shape=(5,),
name="clean_categories")
project_subject_categories_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size)(project_subject_categories_input)
project_subject_categories_feature = Flatten()
(project_subject_categories_feature)

X_train_clean_categories_pad_seq.shape
```

```
(49041, 5)
```

Clean SubCategories Feature

```
t5 = Tokenizer()
t5.fit_on_texts(X_train['clean_subcategories'].values)
vocab_size = len(t5.word_index) + 1

X_train_clean_subcategories_pre_seq =
t5.texts_to_sequences(X_train['clean_subcategories'])
X_test_clean_subcategories_pre_seq =
t5.texts_to_sequences(X_test['clean_subcategories'])
X_cv_clean_subcategories_pre_seq =
t5.texts_to_sequences(X_cv['clean_subcategories'])

max_length = 5
X_train_clean_subcategories_pad_seq =
pad_sequences(X_train_clean_subcategories_pre_seq,maxlen=max_length)
X_test_clean_subcategories_pad_seq =
pad_sequences(X_test_clean_subcategories_pre_seq,maxlen=max_length)
X_cv_clean_subcategories_pad_seq =
pad_sequences(X_cv_clean_subcategories_pre_seq,maxlen=max_length)

embedding_size = int(np.ceil((Unique_school_state) / 2))

project_subject_subcategories_input = Input(shape=(5,),
name="clean_subcategories")
project_subject_subcategories_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size)(project_subject_subcategories_input)
project_subject_subcategories_feature = Flatten()
(project_subject_subcategories_feature)

X_train_clean_subcategories_pad_seq.shape
```

```
(49041, 5)
```

Teacher Prefix Feature

```
t6 = Tokenizer()
t6.fit_on_texts(X_train['teacher_prefix'].values)

vocab_size = len(t6.word_index) + 1

#print(vocab_size)

X_train_teacher_prefix_pre_seq =
t6.texts_to_sequences(X_train['teacher_prefix'])
X_test_teacher_prefix_pre_seq =
t6.texts_to_sequences(X_test['teacher_prefix'])
X_cv_teacher_prefix_pre_seq =
t6.texts_to_sequences(X_cv['teacher_prefix'])

max_length = 1
X_train_teacher_prefix_pad_seq =
pad_sequences(X_train_teacher_prefix_pre_seq,maxlen=max_length)
X_test_teacher_prefix_pad_seq =
pad_sequences(X_test_teacher_prefix_pre_seq,maxlen=max_length)
X_cv_teacher_prefix_pad_seq =
pad_sequences(X_cv_teacher_prefix_pre_seq,maxlen=max_length)

embedding_size = int(np.ceil((Unique_school_state) / 2))

teacher_prefix_input = Input(shape=(1,), name="teacher_prefix")
teacher_prefix_feature = Embedding(input_dim=vocab_size
+1,output_dim=embedding_size)(teacher_prefix_input)
teacher_prefix_feature = Flatten()(teacher_prefix_feature)

X_train_teacher_prefix_pad_seq.shape
```

```
(49041, 1)
```

Teacher_number_of_previously_posted_projects and Price

```
#Numerical feature -teacher_number_of_previously_posted_projects and
price# Input_school_state
from sklearn.preprocessing import StandardScaler
#from sklearn.preprocessing import scalaralizer

teacher_n_scalar = StandardScaler()
teacher_n_scalar.fit(X_train['teacher_number_of_previously_posted_proj
ects'].values.reshape(-1,1))
X_train_teacher_number_of_previously_posted_project =
teacher_n_scalar.transform(X_train['teacher_number_of_previously_poste
d_projects'].values.reshape(-1, 1))
X_test_teacher_number_of_previously_posted_project =
teacher_n_scalar.transform(X_test['teacher_number_of_previously_posted
_projects'].values.reshape(-1, 1))
```

```
X_cv_teacher_number_of_previously_posted_project =
teacher_n_scalar.transform(X_cv['teacher_number_of_previously_posted_p
rojects'].values.reshape(-1, 1))


price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1))
X_train_price_scalar =
price_scalar.transform(X_train['price'].values.reshape(-1,1))
X_test_price_scalar =
price_scalar.transform(X_test['price'].values.reshape(-1,1))
X_cv_price_scalar =
price_scalar.transform(X_cv['price'].values.reshape(-1,1))

# to cancatenate numeric feature reshaping array
X_train_price_scalar = X_train_price_scalar.reshape(-1,1)
X_test_price_scalar = X_test_price_scalar.reshape(-1,1)
X_cv_price_scalar = X_cv_price_scalar.reshape(-1,1)

X_train_num_teacher_number_of_previously_posted_projects_and_price =
np.hstack((X_train_teacher_number_of_previously_posted_project,X_train
_price_scalar))
X_test_num_teacher_number_of_previously_posted_projects_and_price =
np.hstack((X_test_teacher_number_of_previously_posted_project,X_test_p
rice_scalar))
X_cv_num_teacher_number_of_previously_posted_projects_and_price =
np.hstack((X_cv_teacher_number_of_previously_posted_project,X_cv_price
_scalar))

numerical_input = Input(shape=(2,))
numeric_dense = Dense(10)(numerical_input )
```

##2.3 Defining the model-"Numerical features"

```
from keras.layers import concatenate
con_data_2 = concatenate([featurized_essay,school_state_feature,

teacher_prefix_feature,project_grade_category_feature,
                        project_subject_categories_feature,
                        project_subject_subcategories_feature,
                        numeric_dense])
```

#Model 2

```
# Layer 1
model2 = Dense(256, activation = 'relu',
kernel_initializer="glorot_normal",
kernel_regularizer = regularizers.l2(0.01))(con_data_2)
model2 = Dropout(0.4)(model2)
# Layer 2
```

```
model2= Dense(128, activation =
'relu',kernel_initializer="glorot_normal",
kernel_regularizer = regularizers.l2(0.01))(model2)
model2= Dropout(0.4)(model2)
# Layer 3
model2 = Dense(64, activation = 'relu',kernel_initializer="he_normal",
kernel_regularizer = regularizers.l2(0.01))(model2)
model2 = Dropout(0.4)(model2)
# Output layer
output = Dense(2, activation = 'softmax', name= 'Model_2_output')
(model2)

#Model_1
Model_2 = Model(inputs = [essay_input,teacher_prefix_input,
                          school_state_input,
                          project_grade_category_input,
                          project_subject_categories_input,
project_subject_subcategories_input,
                          numerical_input], outputs = [output])
print(Model_2.summary())
```

```
Model: "model"
__________________________________________________________________________
_____
 Layer (type)                   Output Shape         Param #
Connected to
==========================================================================
=============================
 essay (InputLayer)             [(None, 800)]        0           []



 embedding (Embedding)          (None, 800, 300)     12411000
['essay[0][0]']


 school_state (InputLayer)      [(None, 1)]          0           []



 teacher_prefix (InputLayer)    [(None, 1)]          0           []



 project_grade_category (InputL  [(None, 5)]         0           []

 ayer)
```

```
 clean_categories (InputLayer)  [(None, 5)]          0           []


 clean_subcategories (InputLaye  [(None, 5)]         0           []

 r)


 lstm (LSTM)                    (None, 100)          160400
['embedding[0][0]']

 embedding_1 (Embedding)        (None, 1, 3)         159
['school_state[0][0]']

 embedding_5 (Embedding)        (None, 1, 3)         21
['teacher_prefix[0][0]']

 embedding_2 (Embedding)        (None, 5, 3)         33
['project_grade_category[0][0]']

 embedding_3 (Embedding)        (None, 5, 3)         51
['clean_categories[0][0]']

 embedding_4 (Embedding)        (None, 5, 3)         117
['clean_subcategories[0][0]']

 input_1 (InputLayer)           [(None, 2)]          0           []


 flatten (Flatten)              (None, 100)          0
['lstm[0][0]']

 flatten_1 (Flatten)            (None, 3)            0
['embedding_1[0][0]']

 flatten_5 (Flatten)            (None, 3)            0
['embedding_5[0][0]']
```

```
 flatten_2 (Flatten)            (None, 15)           0
['embedding_2[0][0]']


 flatten_3 (Flatten)            (None, 15)           0
['embedding_3[0][0]']


 flatten_4 (Flatten)            (None, 15)           0
['embedding_4[0][0]']


 dense (Dense)                  (None, 10)           30
['input_1[0][0]']


 concatenate (Concatenate)      (None, 161)          0
['flatten[0][0]',

'flatten_1[0][0]',

'flatten_5[0][0]',

'flatten_2[0][0]',

'flatten_3[0][0]',

'flatten_4[0][0]',

'dense[0][0]']


 dense_1 (Dense)                (None, 256)          41472
['concatenate[0][0]']


 dropout (Dropout)              (None, 256)          0
['dense_1[0][0]']


 dense_2 (Dense)                (None, 128)          32896
['dropout[0][0]']


 dropout_1 (Dropout)            (None, 128)          0
['dense_2[0][0]']
```

```
 dense_3 (Dense)                (None, 64)           8256
['dropout_1[0][0]']


 dropout_2 (Dropout)            (None, 64)           0
['dense_3[0][0]']


 Model_2_output (Dense)         (None, 2)            130
['dropout_2[0][0]']


==================================================================
============================
Total params: 12,654,565
Trainable params: 243,565
Non-trainable params: 12,411,000
__________________________________________________________________
_____________________________
None
```

```
#Train data
train_dt_2 =
[X_tr_pad_essay,X_train_teacher_prefix_pad_seq,X_train_school_state_pa
d_seq,X_train_project_grade_category_pad_seq,X_train_clean_subcategori
es_pad_seq,X_train_clean_categories_pad_seq,X_train_num_teacher_number
_of_previously_posted_projects_and_price]
# Test data
test_dt_2 =
[X_ts_pad_essay,X_test_teacher_prefix_pad_seq,X_test_school_state_pad_
seq,X_test_project_grade_category_pad_seq,X_test_clean_subcategories_p
ad_seq,X_test_clean_categories_pad_seq,
X_test_num_teacher_number_of_previously_posted_projects_and_price]
# CV data
cv_dt_2 =
[X_cv_pad_essay,X_cv_teacher_prefix_pad_seq,X_cv_school_state_pad_seq,
X_cv_project_grade_category_pad_seq,X_cv_clean_subcategories_pad_seq,X
_cv_clean_categories_pad_seq,
X_cv_num_teacher_number_of_previously_posted_projects_and_price]

def aoc_roc(y_true, y_pred):
  return tf.py_function(roc_auc_score, (y_true, y_pred), tf.double)

y_train_dt_2 = np_utils.to_categorical(y_train, 2)
y_cv_dt_2 = np_utils.to_categorical(y_cv, 2)
y_test_dt_2 = np_utils.to_categorical(y_test, 2)
```

##2.6 Compiling and fititng your model

```
Model_2.compile(optimizer = 'rmsprop', loss =
'categorical_crossentropy', metrics = [aoc_roc])
```

```
%reload_ext tensorboard

log_dir_2="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M
%S")
tensorboard_2=
tf.keras.callbacks.TensorBoard(log_dir=log_dir_2,histogram_freq=1)

filepath="/content/drive/MyDrive/Colab
Notebooks/weights_copy_2.best.hdf5"

earlystop_1 = EarlyStopping(monitor='val_loss', patience=2, verbose=1)
checkpoint = ModelCheckpoint(filepath, monitor='val_auroc', verbose=1,
save_best_only=True, mode='max')
callbk_list = [checkpoint,earlystop_1,tensorboard_2]

history_2 =
Model_2.fit(train_dt_2 ,y_train_dt_2,batch_size=512,epochs=15,validati
on_data=(cv_dt_2,y_cv_dt_2),callbacks=callbk_list)

Epoch 1/15
96/96 [==============================] - ETA: 0s - loss: 2.4143 -
aoc_roc: 0.5508

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 29s 268ms/step - loss:
2.4143 - aoc_roc: 0.5508 - val_loss: 0.9351 - val_aoc_roc: 0.6438
Epoch 2/15
96/96 [==============================] - ETA: 0s - loss: 0.6154 -
aoc_roc: 0.6323

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 23s 244ms/step - loss:
0.6154 - aoc_roc: 0.6323 - val_loss: 0.4497 - val_aoc_roc: 0.6962
Epoch 3/15
96/96 [==============================] - ETA: 0s - loss: 0.4376 -
aoc_roc: 0.6269

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 26s 270ms/step - loss:
0.4376 - aoc_roc: 0.6269 - val_loss: 0.4106 - val_aoc_roc: 0.7012
Epoch 4/15
96/96 [==============================] - ETA: 0s - loss: 0.4253 -
aoc_roc: 0.6134

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.
```

```
 96/96 [==============================] - 23s 237ms/step - loss:
0.4253 - aoc_roc: 0.6134 - val_loss: 0.4158 - val_aoc_roc: 0.6829
Epoch 5/15
96/96 [==============================] - ETA: 0s - loss: 0.4107 -
aoc_roc: 0.6841

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 25s 258ms/step - loss:
0.4107 - aoc_roc: 0.6841 - val_loss: 0.3999 - val_aoc_roc: 0.7159
Epoch 6/15
96/96 [==============================] - ETA: 0s - loss: 0.4059 -
aoc_roc: 0.6982

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 23s 240ms/step - loss:
0.4059 - aoc_roc: 0.6982 - val_loss: 0.4060 - val_aoc_roc: 0.7254
Epoch 7/15
96/96 [==============================] - ETA: 0s - loss: 0.4023 -
aoc_roc: 0.7095

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 23s 241ms/step - loss:
0.4023 - aoc_roc: 0.7095 - val_loss: 0.3979 - val_aoc_roc: 0.7312
Epoch 8/15
96/96 [==============================] - ETA: 0s - loss: 0.3980 -
aoc_roc: 0.7198

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 25s 258ms/step - loss:
0.3980 - aoc_roc: 0.7198 - val_loss: 0.3873 - val_aoc_roc: 0.7424
Epoch 9/15
96/96 [==============================] - ETA: 0s - loss: 0.3947 -
aoc_roc: 0.7268

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 23s 237ms/step - loss:
0.3947 - aoc_roc: 0.7268 - val_loss: 0.3863 - val_aoc_roc: 0.7467
Epoch 10/15
96/96 [==============================] - ETA: 0s - loss: 0.3891 -
aoc_roc: 0.7349

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.
```

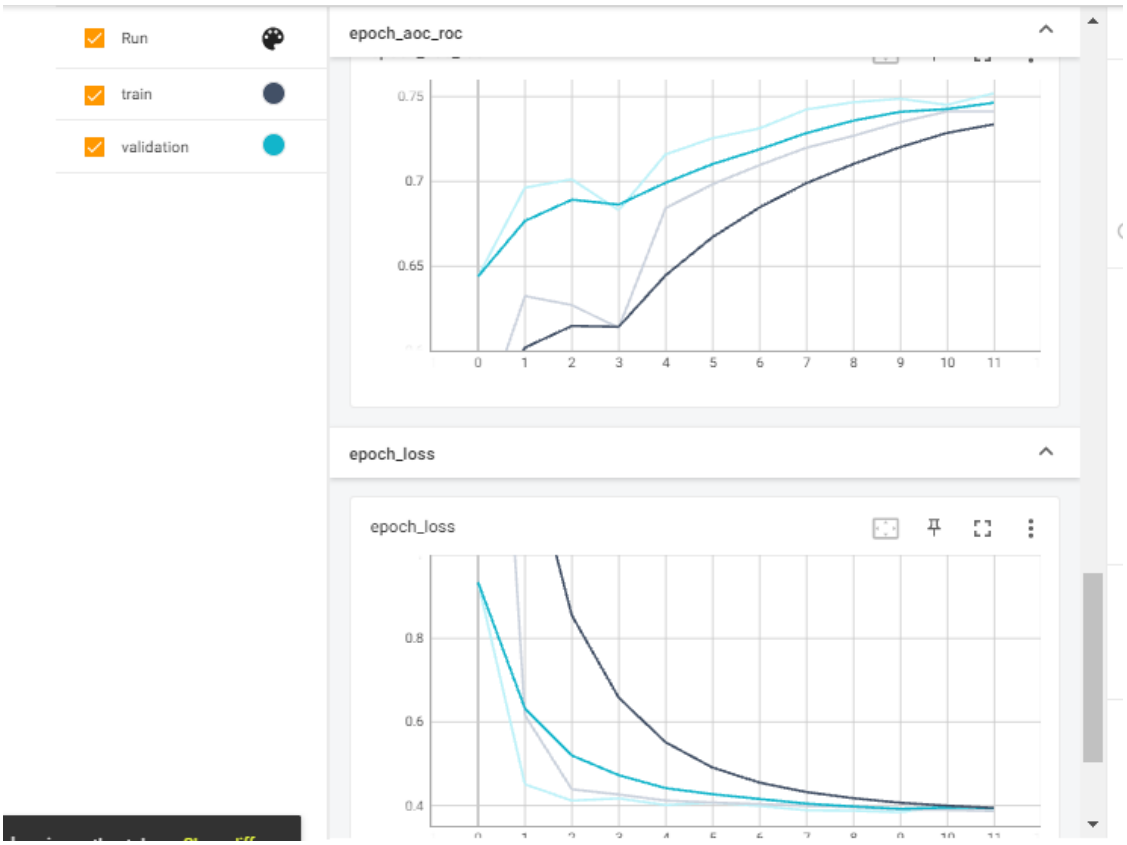
```
 96/96 [==============================] - 25s 258ms/step - loss:
0.3891 - aoc_roc: 0.7349 - val_loss: 0.3824 - val_aoc_roc: 0.7487
Epoch 11/15
96/96 [==============================] - ETA: 0s - loss: 0.3879 -
aoc_roc: 0.7412

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 25s 262ms/step - loss:
0.3879 - aoc_roc: 0.7412 - val_loss: 0.3989 - val_aoc_roc: 0.7451
Epoch 12/15
96/96 [==============================] - ETA: 0s - loss: 0.3856 -
aoc_roc: 0.7412

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 23s 240ms/step - loss:
0.3856 - aoc_roc: 0.7412 - val_loss: 0.3909 - val_aoc_roc: 0.7519
Epoch 12: early stopping
```

```
#tf.summary.create_file_writer('log_dir2')
```

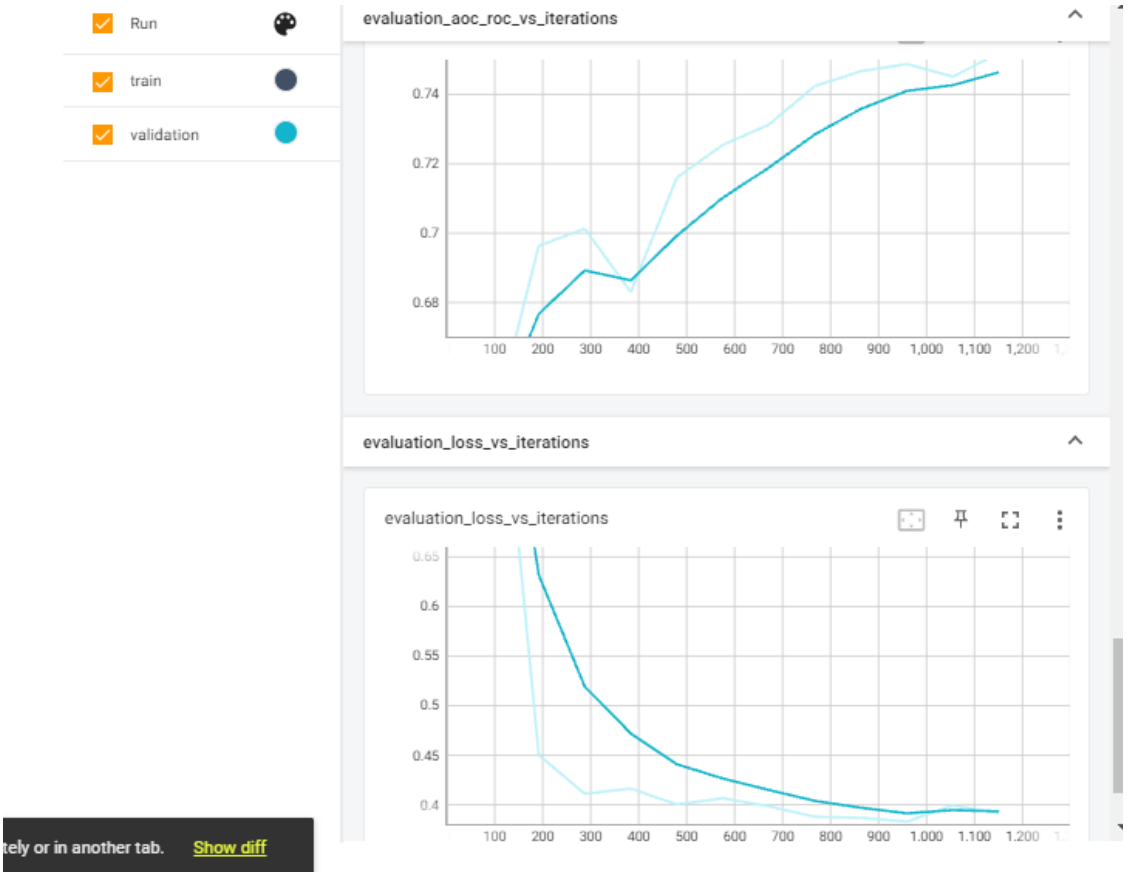
```
%reload_ext tensorboard
%tensorboard --logdir $log_dir_2
```

```
<IPython.core.display.Javascript object>
```

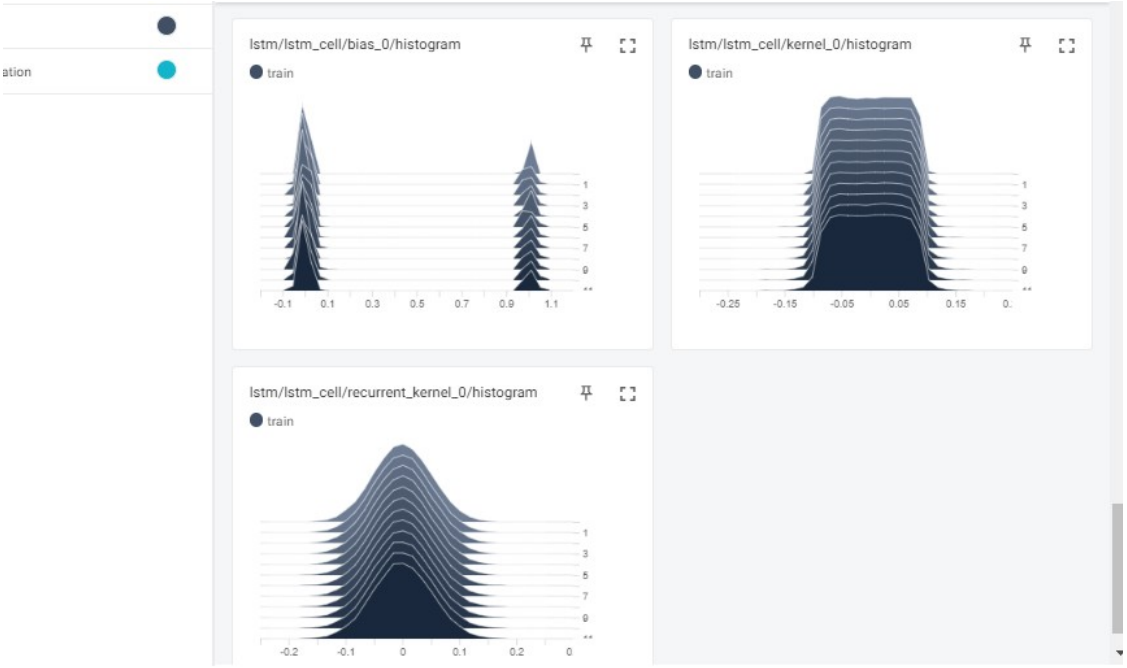
```
display.display(Image.open('/content/bb1.png'))
```




```
display.display(Image.open('/content/bb2.png'))
```




```
display.display(Image.open('/content/bb3.png'))
```

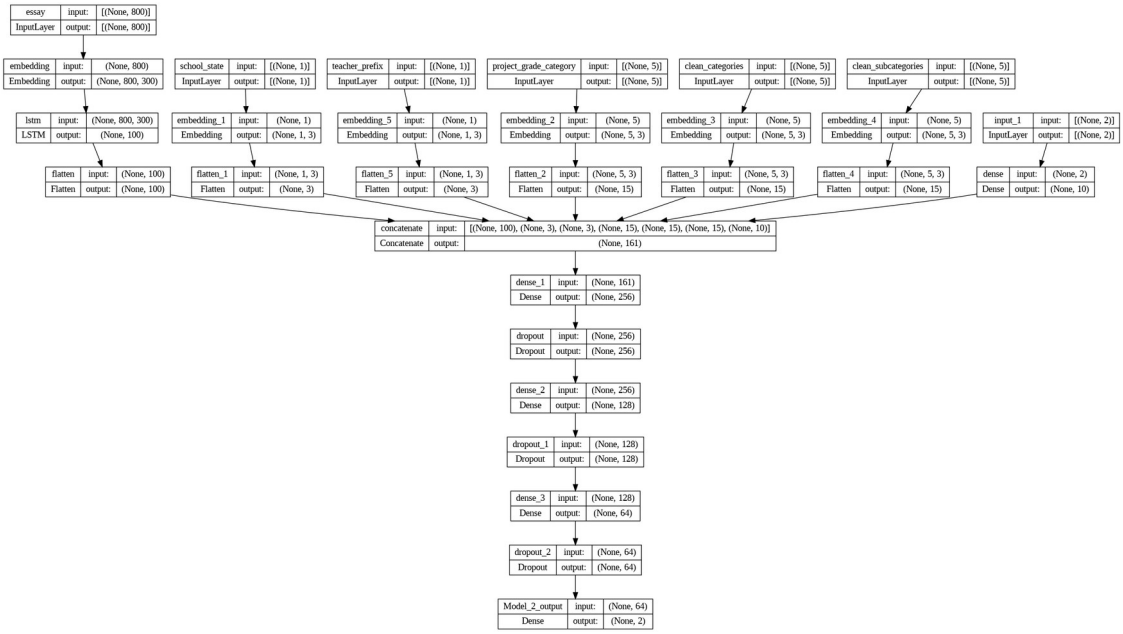



```
scores = Model_2.evaluate(test_dt_2, y_test_dt_2,
verbose=0,batch_size=512)
```

```
print("%s: %.2f%%" % (Model_2.metrics_names[1], scores[1]*100))
```

```
aoc_roc: 74.37%
```

```
from keras.utils.vis_utils import plot_model
plot_model(Model_2, show_shapes=True, show_layer_names=True,
to_file='model_2.png')
from IPython.display import Image
Image(retina=True, filename='model_2.png')
```



#3. Model 3

##3.1 Vectorization of Categorical features

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values)

train_clean_categories_ohe =
vectorizer.transform(X_train['clean_categories'].values)
test_clean_categories_ohe =
vectorizer.transform(X_test['clean_categories'].values)
cv_clean_categories_ohe =
vectorizer.transform(X_cv['clean_categories'].values)

vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values)
train_clean_subcategories_ohe =
vectorizer.transform(X_train['clean_subcategories'].values)
test_clean_subcategories_ohe =
vectorizer.transform(X_test['clean_subcategories'].values)
cv_clean_subcategories_ohe =
vectorizer.transform(X_cv['clean_subcategories'].values)
```

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values)
train_school_state_ohe =
vectorizer.transform(X_train['school_state'].values)
test_school_state_ohe =
vectorizer.transform(X_test['school_state'].values)
cv_school_state_ohe =
vectorizer.transform(X_cv['school_state'].values)

vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values)
train_teacher_prefix_ohe =
vectorizer.transform(X_train['teacher_prefix'].values)
test_teacher_prefix_ohe =
vectorizer.transform(X_test['teacher_prefix'].values)
cv_teacher_prefix_ohe =
vectorizer.transform(X_cv['teacher_prefix'].values)

vectorizer = CountVectorizer()
vectorizer.fit(X_train['project_grade_category'].values)
train_project_grade_ohe =
vectorizer.transform(X_train['project_grade_category'].values)
test_project_grade_ohe =
vectorizer.transform(X_test['project_grade_category'].values)
cv_project_grade_ohe =
vectorizer.transform(X_cv['project_grade_category'].values)
```

Stacking All the Other Than Text Feature

```
from scipy.sparse import hstack
stack_train = hstack((train_school_state_ohe,
train_teacher_prefix_ohe, train_project_grade_ohe,
train_clean_categories_ohe,
train_clean_subcategories_ohe,X_train_num_teacher_number_of_previously
_posted_projects_and_price))
stack_test = hstack((test_school_state_ohe, test_teacher_prefix_ohe,
test_project_grade_ohe, test_clean_categories_ohe,
test_clean_subcategories_ohe,X_test_num_teacher_number_of_previously_p
osted_projects_and_price))
stack_cv = hstack((cv_school_state_ohe, cv_teacher_prefix_ohe,
cv_project_grade_ohe, cv_clean_categories_ohe,
cv_clean_subcategories_ohe,
X_cv_num_teacher_number_of_previously_posted_projects_and_price))

# converting the data series object to dense series object
stacked_train_dense=stack_train.todense()
stacked_test_dense=stack_test.todense()
stacked_cv_dense=stack_cv.todense()
```

```
# Expanding the shape of the array
other_than_text_train = np.expand_dims(stacked_train_dense,2)
other_than_text_test = np.expand_dims(stacked_test_dense,2)
other_than_text_cv = np.expand_dims(stacked_cv_dense,2)

input_other_than_text_only =
Input(shape=(101,1),name="other_than_text_train")
conv1D_1 = Conv1D(filters=128, kernel_size=3,
activation='relu',kernel_initializer="he_normal")
(input_other_than_text_only)
conv1D_2 = Conv1D(filters=128, kernel_size=3,
activation='relu',kernel_initializer="he_normal")(conv1D_1)
flatten_data_otherthan_essay = Flatten()(conv1D_2)

concatenated_model_3 =
concatenate([featurized_essay,flatten_data_otherthan_essay])
dense_1 =
Dense(512,activation="relu",kernel_initializer="glorot_normal",kernel_
regularizer=regularizers.l2(0.01))(concatenated_model_3)
drop_1 = Dropout(0.5)(dense_1)
dense_2 =
Dense(128,activation="relu",kernel_initializer="he_normal",kernel_regu
larizer=regularizers.l2(0.01))(drop_1)
drop_2 = Dropout(0.5)(dense_2)
dense_3 =
Dense(64,activation="relu",kernel_initializer="glorot_normal",kernel_r
egularizer=regularizers.l2(0.01))(drop_2)
output_3 = Dense(2, activation='softmax', name='output')(dense_3)

Model_3 =
Model(inputs=[essay_input,input_other_than_text_only],outputs=[output_
3])

X_train_3 = [X_tr_pad_essay,other_than_text_train]
X_test_3 = [X_ts_pad_essay,other_than_text_test]
X_cv_3 = [X_cv_pad_essay,other_than_text_cv]
```

##3.2 Compiling and fititng The model

```
Model_3.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=[aoc_roc])

%reload_ext tensorboard

log_dir_3="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M
%S")
tensorboard_3=
tf.keras.callbacks.TensorBoard(log_dir=log_dir_3,histogram_freq=1)

filepath="/content/drive/MyDrive/Colab
Notebooks/weights_copy_3.best.hdf5"
```

```
earlystop_1 = EarlyStopping(monitor='val_loss', patience=7, verbose=1)
checkpoint = ModelCheckpoint(filepath, monitor='val_auroc', verbose=1,
save_best_only=True, mode='max')
callbk_list = [checkpoint,earlystop_1,tensorboard_3]

history_3 =
Model_3.fit(X_train_3,y_train_dt_2,batch_size=512,epochs=15,validation
_data=(X_cv_3,y_cv_dt_2),callbacks=callbk_list)

Epoch 1/15
96/96 [==============================] - ETA: 0s - loss: 0.3637 -
aoc_roc: 0.7994

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 30s 278ms/step - loss:
0.3637 - aoc_roc: 0.7994 - val_loss: 0.3765 - val_aoc_roc: 0.7574
Epoch 2/15
96/96 [==============================] - ETA: 0s - loss: 0.3525 -
aoc_roc: 0.8054

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 26s 269ms/step - loss:
0.3525 - aoc_roc: 0.8054 - val_loss: 0.3986 - val_aoc_roc: 0.7576
Epoch 3/15
96/96 [==============================] - ETA: 0s - loss: 0.3437 -
aoc_roc: 0.8163

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 26s 274ms/step - loss:
0.3437 - aoc_roc: 0.8163 - val_loss: 0.3816 - val_aoc_roc: 0.7583
Epoch 4/15
96/96 [==============================] - ETA: 0s - loss: 0.3366 -
aoc_roc: 0.8246

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 28s 292ms/step - loss:
0.3366 - aoc_roc: 0.8246 - val_loss: 0.3828 - val_aoc_roc: 0.7561
Epoch 5/15
96/96 [==============================] - ETA: 0s - loss: 0.3267 -
aoc_roc: 0.8369

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.
```

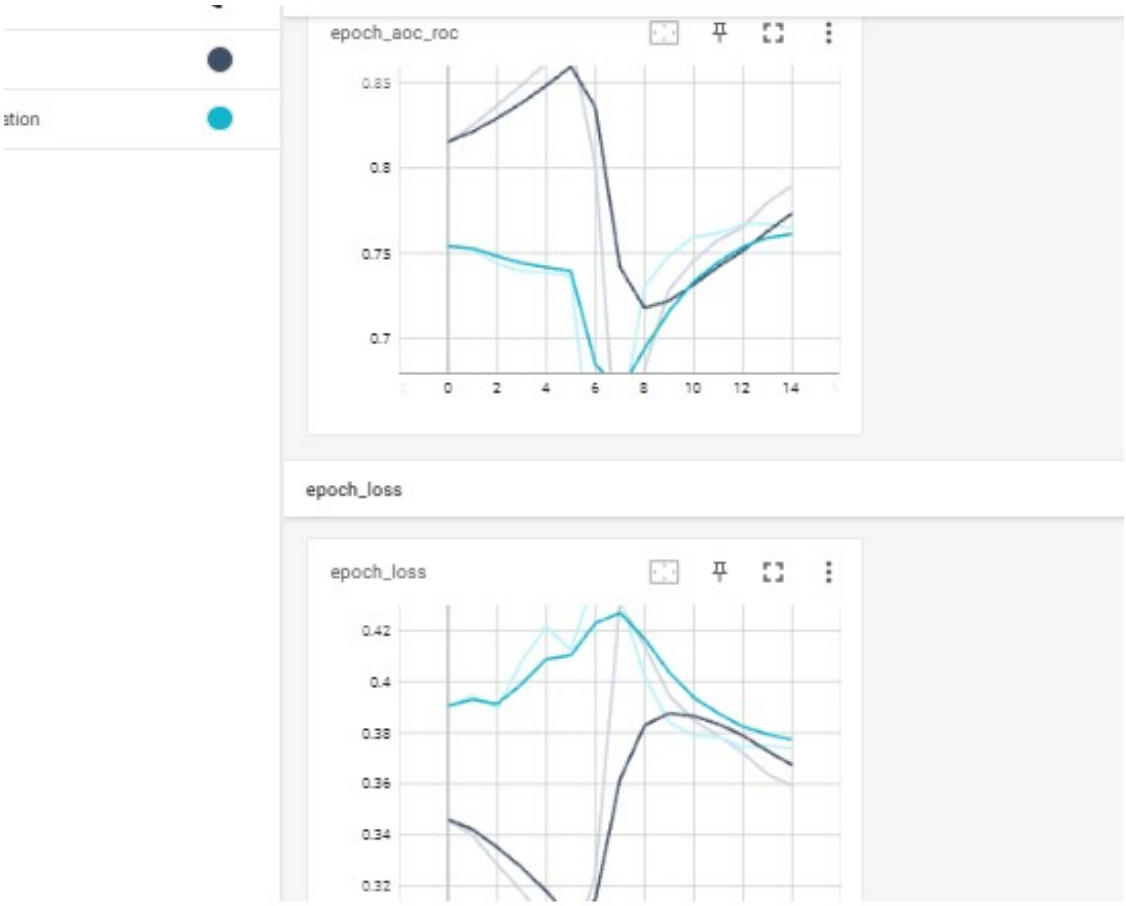
```
 96/96 [==============================] - 28s 289ms/step - loss:
0.3267 - aoc_roc: 0.8369 - val_loss: 0.3953 - val_aoc_roc: 0.7476
Epoch 6/15
96/96 [==============================] - ETA: 0s - loss: 0.3386 -
aoc_roc: 0.8147

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 29s 304ms/step - loss:
0.3386 - aoc_roc: 0.8147 - val_loss: 0.4093 - val_aoc_roc: 0.7479
Epoch 7/15
96/96 [==============================] - ETA: 0s - loss: 0.3361 -
aoc_roc: 0.8217

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 27s 284ms/step - loss:
0.3361 - aoc_roc: 0.8217 - val_loss: 0.3922 - val_aoc_roc: 0.7476
Epoch 8/15
96/96 [==============================] - ETA: 0s - loss: 0.3125 -
aoc_roc: 0.8506

WARNING:tensorflow:Can save best model only with val_auroc available,
skipping.

 96/96 [==============================] - 26s 275ms/step - loss:
0.3125 - aoc_roc: 0.8506 - val_loss: 0.3988 - val_aoc_roc: 0.7451
Epoch 8: early stopping
```

```
#tf.summary.create_file_writer('log_dir3')
```

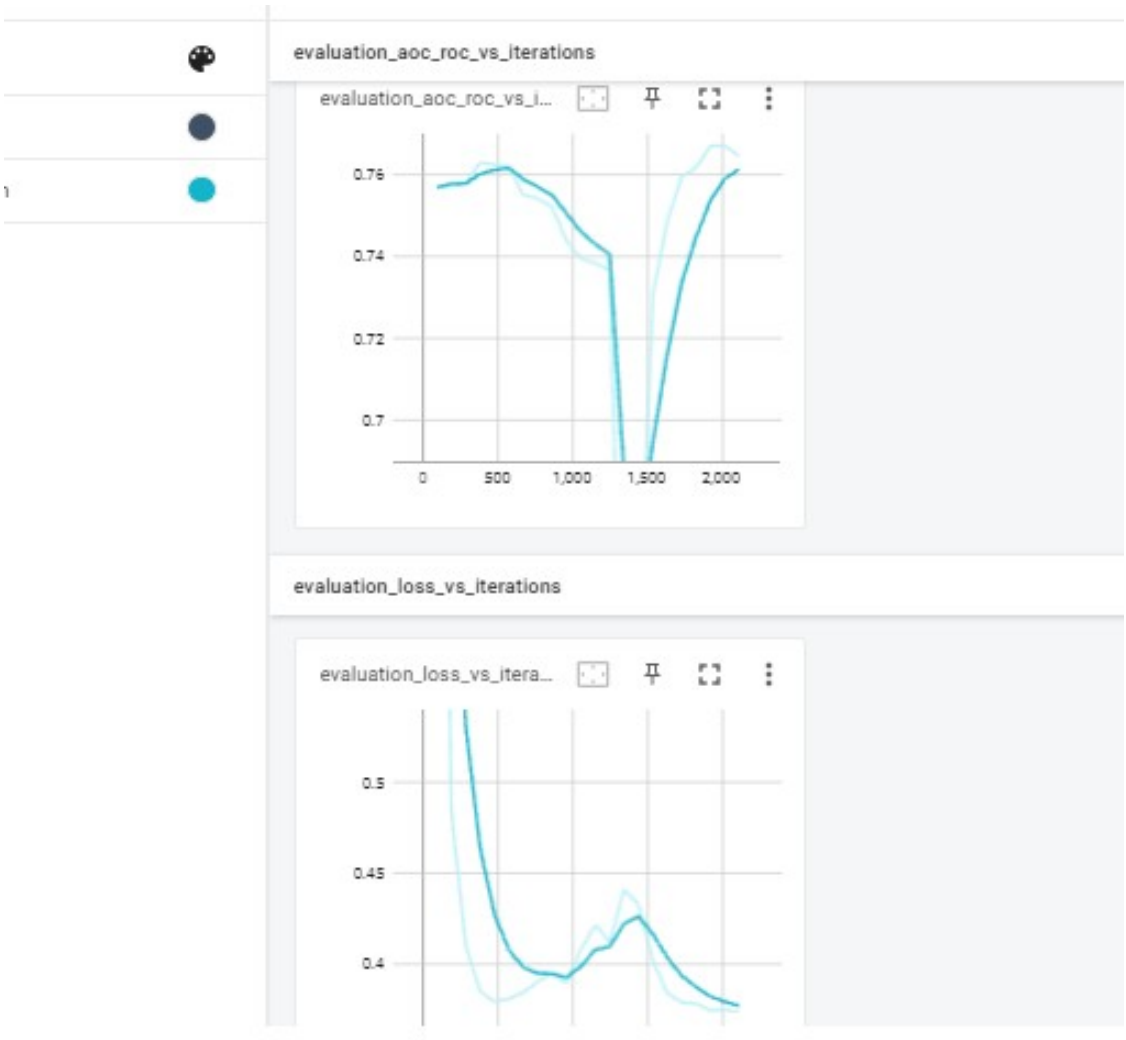
```
%reload_ext tensorboard
%tensorboard --logdir $log_dir_3
```

```
<IPython.core.display.Javascript object>
```

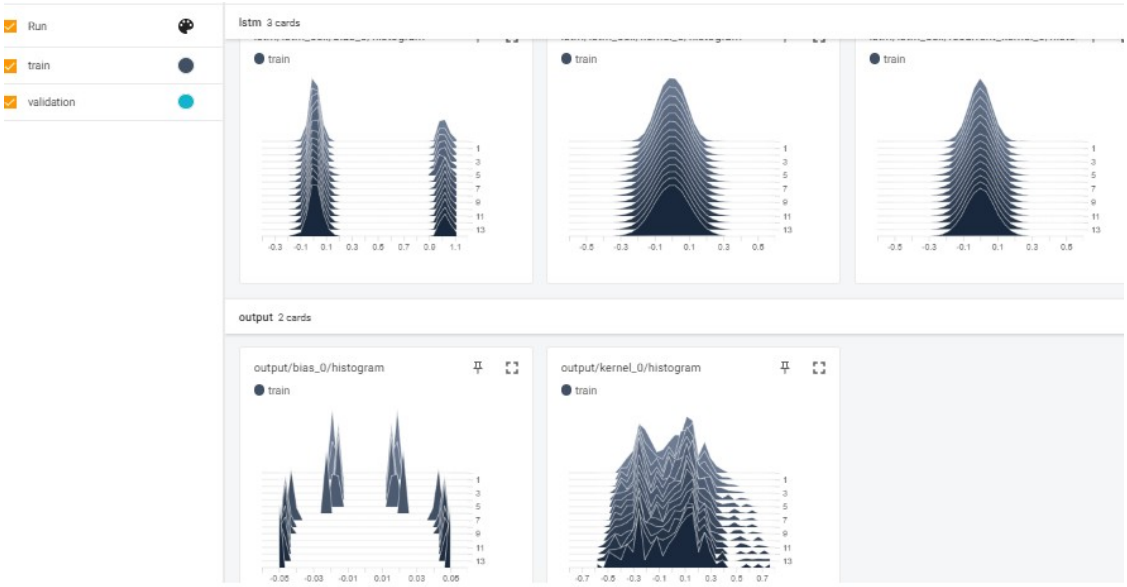
```
display.display(Image.open('/content/cc1.png'))
```




```
display.display(Image.open('/content/cc2.png'))
```



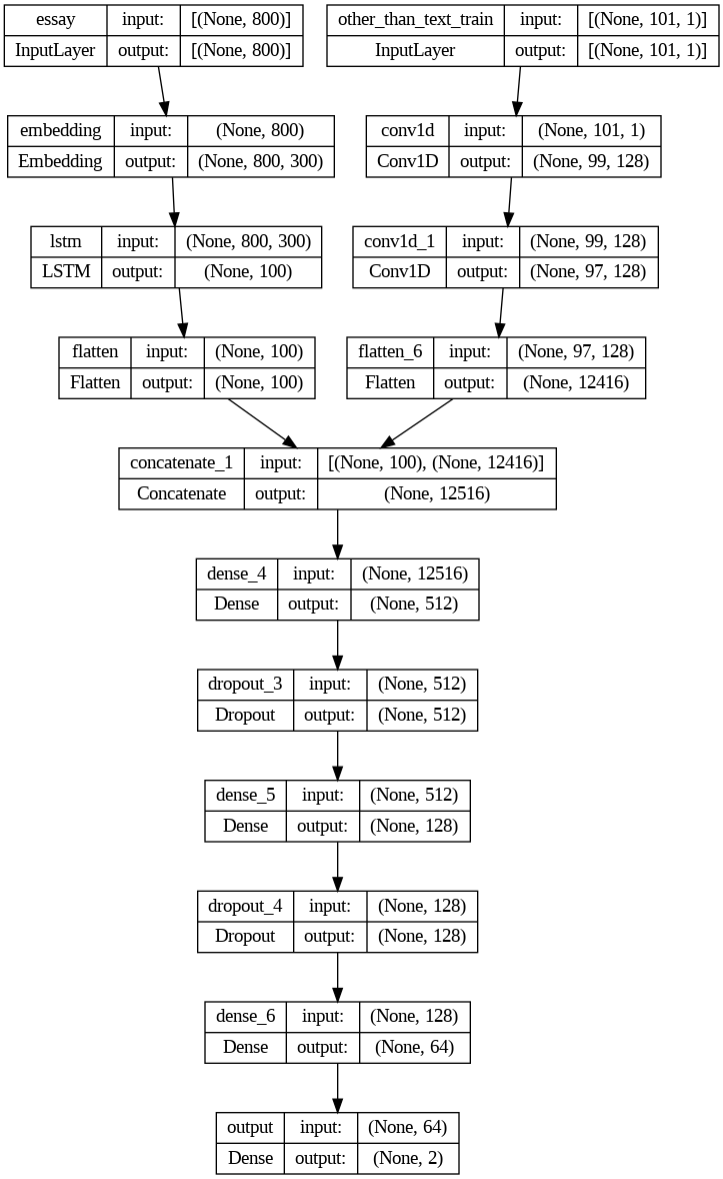

```
display.display(Image.open('/content/cc3.png'))
```




```
scores = Model_3.evaluate(X_test_3, y_test_dt_2,
verbose=0,batch_size=512)

print("%s: %.2f%%" % (Model_3.metrics_names[1], scores[1]*100))
```

```
aoc_roc: 74.56%
```

```
from keras.utils.vis_utils import plot_model
plot_model(Model_3, show_shapes=True, show_layer_names=True,
to_file='model_3.png')
from IPython.display import Image
Image(retina=True, filename='model_3.png')
```




```
from prettytable import PrettyTable
myTable =
PrettyTable(['Model␣↪','Optimizer','Max_Train_Accuracy','Max_Validatio
n_Accuracy'])
myTable.add_row(['Model-1','Rmsprop','0.7411','0.7576'])
myTable.add_row(['Model-2','Rmsprop','0.7412','0.7519'])
myTable.add_row(['Model-3','Rmsprop','0.8506','0.7451'])
print(myTable)
```

```
+---------+-----------+--------------------+-------------------------+
| Model␣↪ | Optimizer | Max_Train_Accuracy | Max_Validation_Accuracy |
+---------+-----------+--------------------+-------------------------+
| Model-1 |  Rmsprop  |       0.7411       |          0.7576         |
| Model-2 |  Rmsprop  |       0.7412       |          0.7519         |
```

Model-3	Rmsprop	0.8506	0.7451