In [1]:

```
from numpy import array
from numpy import asarray
from numpy import zeros
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers import Input, Embedding, LSTM, Dense, concatenate, Dropout
from keras.models import Model


import pandas as pd
import numpy as np

import pickle
from tqdm import tqdm
import os

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
```

```
Using TensorFlow backend.
```

In [2]:

```
data=pd.read_csv("fully_processed_data.csv")
data.head(2)
```

Out[2]:

	Unnamed: 0	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	pro
0	0	in	mrs	grades_prek_2	0	
1	1	fl	mr	grades_6_8	7	

In [3]:

```
from sklearn.model_selection import train_test_split

X=data.drop('project_is_approved', axis=1)
y=data['project_is_approved']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
del(data)
```

Total Text Data

Build training data

In [4]:

```
docs_essay_train=list(X_train.essay.values)
labels_train=np.array(y_train)
```

In [5]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_essay_train)
vocab_size_train = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_essay_train = tokens.texts_to_sequences(docs_essay_train)
#print(encoded_docs)
print(vocab_size_train)
```

```
51537
```

In [6]:

```
max_len=0
all_lengths=[]
for sent in docs_essay_train:
    length=len(sent.split())
    all_lengths.append(length)
print(max(all_lengths))
```

```
320
```

In [7]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_essay_train = pad_sequences(encoded_docs_essay_train, maxlen=max_length, padding='post')
print(padded_docs_essay_train)
```

```
[[  41  100  170 ...    0    0    0]
 [ 538  489 3840 ...    0    0    0]
 [ 173   72  651 ...    0    0    0]
 ...
 [   1   86  534 ...    0    0    0]
 [ 788 4659    4 ...    0    0    0]
 [5403 8112 9655 ...    0    0    0]]
```

In [8]:

```
#Load the whole embedding into memory
embeddings_index = dict()
file = open('glove.6B.300d.txt')
for line in file:
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
file.close()

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

#Create a weight matrix for words in training docs
embedding_matrix = zeros((vocab_size_train, 300))
for word, i in tqdm(tokens.word_index.items()):
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector #embedding_matrix.shape: (9049, 300)

print(len(embedding_matrix))
print(len(embedding_matrix[0]))
```

```
100%|██████████| 51536/51536 [00:00<00:00, 284795.32it/s]

Loaded 400000 word vectors.
51537
300
```

In [9]:

```
#Get the flattened LSTM output for input text
input_layer1 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size_train, output_dim=300, weights=[embedding_matrix], input_length=max_le
ngth, trainable=False)(input_layer1)
lstm_out = LSTM(32, return_sequences=True)(embedding)
flatten_lstm_out = Flatten()(lstm_out)
```

```
WARNING:tensorflow:From /root/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/op_d
ef_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be re
moved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

Building test data

In [10]:

```
docs_essay_test=list(X_test.essay.values)
labels_test=np.array(y_test)
```

In [11]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_essay_test)
vocab_size_test = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_essay_test = tokens.texts_to_sequences(docs_essay_test)
#print(encoded_docs)
print(vocab_size_test)
```

```
30232
```

In [12]:

```
#Pad documents to a max length
padded_docs_essay_test = pad_sequences(encoded_docs_essay_test, maxlen=max_length, padding='post')
print(padded_docs_essay_test)
```

```
[[   1  592  460 ...    0    0    0]
 [  68   78  122 ...    0    0    0]
 [ 170    5    6 ...    0    0    0]
 ...
 [   1 1188    6 ...    0    0    0]
 [   2  670  260 ...    0    0    0]
 [  63   31    1 ...    0    0    0]]
```

Categorical data: school_state

Building train data

In [13]:

```
docs_school_state_train=list(X_train.school_state.values)
```

In [14]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_school_state_train)
vocab_size_train = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_school_state_train = tokens.texts_to_sequences(docs_school_state_train)
#print(encoded_docs)
print(vocab_size_train)
```

```
52
```

In [15]:

```
max_len=0
all_lengths=[]
for sent in docs_school_state_train:
    length=len(sent.split())
    all_lengths.append(length)
print(max(all_lengths))
```

```
1
```

In [16]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_school_state_train = pad_sequences(encoded_docs_school_state_train, maxlen=max_length, padding='post'
)
#print(padded_docs_school_state)
```

In [17]:

```
#Get the flattened LSTM output for input text
input_layer2 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size_train, output_dim=5, input_length=max_length, trainable=True)(input_la
yer2)
flatten_school_state = Flatten()(embedding)
```

Building train data

In [18]:

```
docs_school_state_test=list(X_test.school_state.values)
```

In [19]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_school_state_test)
vocab_size_test = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_school_state_test = tokens.texts_to_sequences(docs_school_state_test)
#print(encoded_docs)
print(vocab_size_test)
```

```
52
```

In [20]:

```
# pad documents to a max length of 4 words
padded_docs_school_state_test = pad_sequences(encoded_docs_school_state_test, maxlen=max_length, padding='post')
#print(padded_docs_school_state_test)
```

Categorical data: project_grade_category

Building Train Data

In [21]:

```
docs_project_grade_category_train=list(X_train.project_grade_category.values)
```

In [22]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_project_grade_category_train)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_project_grade_category_train = tokens.texts_to_sequences(docs_project_grade_category_train)
#print(encoded_docs)
print(vocab_size)
```

```
10
```

In [23]:

```
max_len=0
all_lengths=[]
for sent in docs_project_grade_category_train:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
13
```

In [24]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_project_grade_category_train = pad_sequences(encoded_docs_project_grade_category_train, maxlen=max_le
ngth, padding='post')
print(padded_docs_project_grade_category_train)
```

```
[[1 4 5 ... 0 0 0]
 [1 4 5 ... 0 0 0]
 [1 4 5 ... 0 0 0]
 ...
 [1 2 3 ... 0 0 0]
 [1 4 5 ... 0 0 0]
 [1 2 3 ... 0 0 0]]
```

In [25]:

```
#Get the flattened LSTM output for input text
input_layer3 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer3)
flatten_project_grade_category = Flatten()(embedding)
```

Building Test Data

In [26]:

```
docs_project_grade_category_test=list(X_test.project_grade_category.values)
```

In [27]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_project_grade_category_test)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_project_grade_category_test = tokens.texts_to_sequences(docs_project_grade_category_test)
#print(encoded_docs)
print(vocab_size)
```

```
10
```

In [28]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_project_grade_category_test = pad_sequences(encoded_docs_project_grade_category_test, maxlen=max_leng
th, padding='post')
print(padded_docs_project_grade_category_test)
```

```
[[1 2 3 ... 0 0 0]
 [1 4 5 ... 0 0 0]
 [1 4 5 ... 0 0 0]
 ...
 [1 4 5 ... 0 0 0]
 [1 6 7 ... 0 0 0]
 [1 2 3 ... 0 0 0]]
```

Categorical data: clean_categories

Building train data

In [29]:

```
docs_clean_categories_train=list(X_train.clean_categories.values)
```

In [30]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_categories_train)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_clean_categories_train = tokens.texts_to_sequences(docs_clean_categories_train)
#print(encoded_docs)
print(vocab_size)
```

```
18
```

In [31]:

```
max_len=0
all_lengths=[]
for sent in encoded_docs_clean_categories_train:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
5
```

In [32]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_clean_categories_train = pad_sequences(encoded_docs_clean_categories_train, maxlen=max_length, paddin
g='post')
print(padded_docs_clean_categories_train)
```

```
[[1 2 0 0 0]
 [5 6 0 0 0]
 [1 2 3 4 0]
 ...
 [3 4 0 0 0]
 [3 4 0 0 0]
 [1 2 3 4 0]]
```

In [33]:

```
#Get the flattened LSTM output for input text
input_layer4 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer4)
flatten_clean_categories = Flatten()(embedding)
```

Building test data

In [34]:

```
docs_clean_categories_test=list(X_test.clean_categories.values)
```

In [35]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_categories_test)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_clean_categories_test = tokens.texts_to_sequences(docs_clean_categories_test)
#print(encoded_docs)
print(vocab_size)
```

```
18
```

In [36]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_clean_categories_test = pad_sequences(encoded_docs_clean_categories_test, maxlen=max_length, padding=
'post')
print(padded_docs_clean_categories_test)
```

```
[[ 1  2 11 12  0]
 [ 1  2  3  4  0]
 [ 7  8  0  0  0]
 ...
 [ 5  6  0  0  0]
 [ 5  6  0  0  0]
 [ 1  2  0  0  0]]
```

Categorical data: clean_subcategories

Building train data

In [37]:

```
docs_clean_subcategories_train=list(X_train.clean_subcategories.values)
```

In [38]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_subcategories_train)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_clean_subcategories_train = tokens.texts_to_sequences(docs_clean_subcategories_train)
#print(encoded_docs)
print(vocab_size)
```

```
49
```

In [39]:

```
max_len=0
all_lengths=[]
for sent in encoded_docs_clean_subcategories_train:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
6
```

In [40]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_clean_subcategories_train = pad_sequences(encoded_docs_clean_subcategories_train, maxlen=max_length,
padding='post')
print(padded_docs_clean_subcategories_train)
```

```
[[ 1  3  4  0  0  0]
 [ 5 10  0  0  0  0]
 [ 3  4  2  0  0  0]
 ...
 [ 9  8  2  0  0  0]
 [ 5 20 11  2  0  0]
 [ 1  2  0  0  0  0]]
```

In [41]:

```
#Get the flattened LSTM output for input text
input_layer5 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer5)
flatten_clean_subcategories = Flatten()(embedding)
```

Building test data

In [42]:

```
docs_clean_subcategories_test=list(X_test.clean_subcategories.values)
```

In [43]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_subcategories_test)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_clean_subcategories_test = tokens.texts_to_sequences(docs_clean_subcategories_test)
#print(encoded_docs)
print(vocab_size)
```

```
49
```

In [44]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_clean_subcategories_test = pad_sequences(encoded_docs_clean_subcategories_test, maxlen=max_length, pa
dding='post')
print(padded_docs_clean_subcategories_test)
```

```
[[ 1 13 12  0  0  0]
 [ 3  4  2  0  0  0]
 [ 6  7  0  0  0  0]
 ...
 [ 5 10 34 21  0  0]
 [ 5 10  0  0  0  0]
 [ 1  3  4  0  0  0]]
```

Categorical data: teacher_prefix

Building train data

In [45]:

```
docs_teacher_prefix_train=list(X_train.teacher_prefix.values)
```

In [46]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_teacher_prefix_train)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_teacher_prefix_train = tokens.texts_to_sequences(docs_teacher_prefix_train)
#print(encoded_docs)
print(vocab_size)
```

```
6
```

In [47]:

```
max_len=0
all_lengths=[]
for sent in encoded_docs_teacher_prefix_train:
    length=len(sent)
    all_lengths.append(length)
print(max(all_lengths))
```

```
1
```

In [48]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_teacher_prefix_train = pad_sequences(encoded_docs_teacher_prefix_train, maxlen=max_length, padding='p
ost')
#print(padded_docs_teacher_prefix)
```

In [49]:

```
#Get the flattened LSTM output for input text
input_layer6 = Input(shape=(max_length,))
embedding = Embedding(input_dim=vocab_size, output_dim=5, input_length=max_length, trainable=True)(input_layer6)
flatten_teacher_prefix = Flatten()(embedding)
```

Building testdata

In [50]:

```
docs_teacher_prefix_test=list(X_test.teacher_prefix.values)
```

In [51]:

```
#Prepare tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(docs_teacher_prefix_test)
vocab_size = len(tokens.word_index) + 1
#Integer encode the documents
encoded_docs_teacher_prefix_test = tokens.texts_to_sequences(docs_teacher_prefix_test)
#print(encoded_docs)
print(vocab_size)
```

```
6
```

In [52]:

```
# pad documents to a max length of 4 words
max_length = max(all_lengths)
padded_docs_teacher_prefix_test = pad_sequences(encoded_docs_teacher_prefix_test, maxlen=max_length, padding='pos
t')
#print(padded_docs_teacher_prefix)
```

teacher_number_of_previously_posted_projects, nrm_price, presence_of_the_numerical_digits

Building train data

In [53]:

```
numerical_df_train=X_train[['teacher_number_of_previously_posted_projects','presence_of_the_numerical_digits','nr
m_price']]
numerical_df_train.head(5)
```

Out[53]:

	teacher_number_of_previously_posted_projects	presence_of_the_numerical_digits	nrm_price
6013	0	0	0.024906
12257	14	0	0.012119
15921	1	0	0.013512
61791	1	0	0.003934
99000	0	0	0.014935

In [54]:

```
#Get the dense layer
input_layer7 = Input(shape=(3,))
dense_layer = Dense(3, activation='relu')(input_layer7)
```

Building test data

In [55]:

```
numerical_df_test=X_test[['teacher_number_of_previously_posted_projects','presence_of_the_numerical_digits','nrm_
price']]
numerical_df_test.head(5)
```

Out[55]:

	teacher_number_of_previously_posted_projects	presence_of_the_numerical_digits	nrm_price
60773	2	0	0.079946
49967	0	0	0.014935
45133	1	0	0.037735
106907	26	1	0.010642
100838	3	0	0.011325

Concatenation of all the layers and building the final model

In [56]:

```
x = concatenate([flatten_lstm_out, flatten_school_state, flatten_project_grade_category, flatten_clean_categories
, flatten_clean_subcategories, flatten_teacher_prefix, dense_layer])
x = Dense(64, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(64, activation='relu')(x)
x = Dropout(0.3)(x)
x = Dense(64, activation='relu')(x)

output = Dense(1, activation='softmax', name='output1')(x)
```

```
WARNING:tensorflow:From /root/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend
.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will
be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

Defining a custom metric AUC

In [57]:

```
"""
#https://datascience.stackexchange.com/questions/13746/how-to-define-a-custom-performance-metric-in-keras/20192#2
0192
import keras
import numpy as np
import sklearn.metrics as sklm


class Metrics(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.auc = []

    def on_epoch_end(self, epoch, logs={}):
        score = np.asarray(self.model.predict(self.validation_data[0]))
        print("Score:", score)
        predict = np.round(np.asarray(self.model.predict(self.validation_data[0])))
        print("predict:", predict)
        targ = self.validation_data[1]

        self.auc.append(sklm.roc_auc_score(targ, predict))
        return

metrics = Metrics()
"""
```

Out[57]:

```
'\n#https://datascience.stackexchange.com/questions/13746/how-to-define-a-custom-performance-metric-
in-keras/20192#20192\nimport keras\nimport numpy as np\nimport sklearn.metrics as sklm\n\n\nclass Me
trics(keras.callbacks.Callback):\n    def on_train_begin(self, logs={}):\n        self.auc = []\n\n
def on_epoch_end(self, epoch, logs={}):\n        score = np.asarray(self.model.predict(self.validati
on_data[0]))\n        print("Score:", score)\n        predict = np.round(np.asarray(self.model.predi
ct(self.validation_data[0])))\n        print("predict:", predict)\n        targ = self.validation_da
ta[1]\n\n        self.auc.append(sklm.roc_auc_score(targ, predict))\n        return\n    \nmetrics =
Metrics()\n'
```

In [58]:

```
from sklearn import metrics as sklm
import tensorflow as tf
def auc_roc(y_true, y_pred):
    try:
        return tf.py_func(sklm.roc_auc_score, (y_true, y_pred), tf.double)
    except ValueError:
        pass
```

Defining a TensorBoard callback object

In [59]:

```
from time import time
from tensorflow.python.keras.callbacks import TensorBoard
tensorboard = TensorBoard(log_dir="logs/{}".format(time))
```

Compiling the final model

In [60]:

```
model = Model(inputs=[input_layer1,input_layer2,input_layer3,input_layer4,input_layer5,input_layer6,input_layer7]
, outputs=output)
model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy',auc_roc])
model.fit(x=[padded_docs_essay_train, padded_docs_school_state_train, padded_docs_project_grade_category_train, p
added_docs_clean_categories_train, padded_docs_clean_subcategories_train, padded_docs_teacher_prefix_train, numer
ical_df_train],
          y=[labels_train],
          validation_data=([padded_docs_essay_test, padded_docs_school_state_test, padded_docs_project_grade_cate
gory_test, padded_docs_clean_categories_test, padded_docs_clean_subcategories_test, padded_docs_teacher_prefix_te
st, numerical_df_test],[labels_test]),
          epochs=10,
          batch_size=64,
          callbacks=[tensorboard])
```

```
WARNING:tensorflow:From <ipython-input-58-db88660b5ed6>:5: py_func (from tensorflow.python.ops.scrip
t_ops) is deprecated and will be removed in a future version.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, use
    tf.py_function, which takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.

WARNING:tensorflow:From /root/anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/math_ops.p
y:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Train on 87398 samples, validate on 21850 samples
Epoch 1/10
87398/87398 [==============================] - 537s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 2/10
87398/87398 [==============================] - 544s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 3/10
87398/87398 [==============================] - 563s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 4/10
87398/87398 [==============================] - 565s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 5/10
87398/87398 [==============================] - 564s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 6/10
87398/87398 [==============================] - 567s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 7/10
87398/87398 [==============================] - 567s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 8/10
87398/87398 [==============================] - 566s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 9/10
87398/87398 [==============================] - 564s 6ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
Epoch 10/10
87398/87398 [==============================] - 570s 7ms/step - loss: 2.4140 - acc: 0.8486 - auc_roc:
0.5000 - val_loss: 2.4136 - val_acc: 0.8486 - val_auc_roc: 0.5000
```

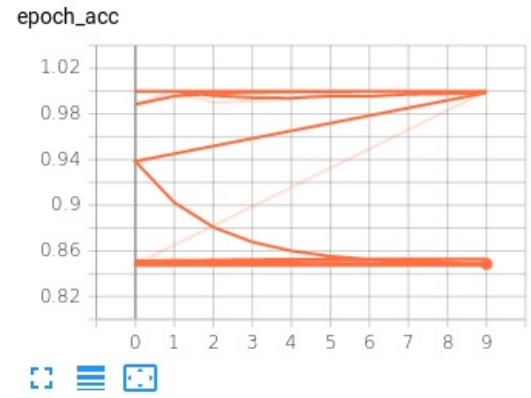
Out[60]:

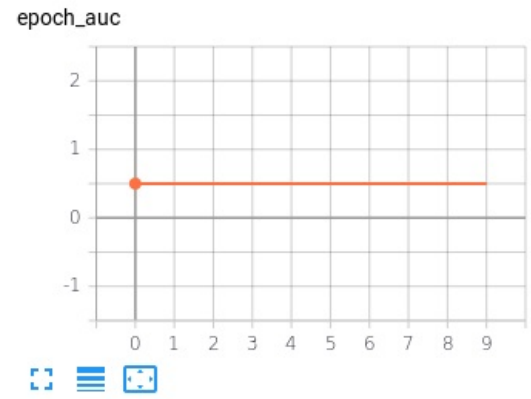
```
<keras.callbacks.History at 0x7f59162ca908>
```

In [61]:

```
!tensorboard --logdir=logs/
```

```
W0726 23:12:59.097108 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.111174 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.143583 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.155557 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.190407 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.239963 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.255520 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
TensorBoard 1.13.1 at http://saugata:6006 (Press CTRL+C to quit)
W0726 23:12:59.301306 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.325635 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.363682 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.382292 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.406032 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.407257 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.435978 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.437082 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.456918 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.458396 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.487738 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.518584 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.558245 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.563189 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.610628 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.615735 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.639674 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.641485 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
W0726 23:12:59.668507 139700309554944 plugin_event_accumulator.py:294] Found more than one graph eve
nt per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  O
verwriting the graph with the newest event.
W0726 23:12:59.669562 139700309554944 plugin_event_accumulator.py:302] Found more than one metagraph
event per run. Overwriting the metagraph with the newest event.
^C
```





In [62]:

```
loss, accuracy = model.evaluate([padded_docs_essay_test, padded_docs_school_state_test, padded_docs_project_grade
_category_test, padded_docs_clean_categories_test, padded_docs_clean_subcategories_test, padded_docs_teacher_pref
ix_test, numerical_df_test],
[labels_test], verbose=0)
print('Accuracy: %f' % (accuracy*100))
```

```
---------------------------------------------------------------------------
InvalidArgumentError                      Traceback (most recent call last)
<ipython-input-62-8caa044dc74d> in <module>
      1 loss, accuracy = model.evaluate([padded_docs_essay_test, padded_docs_school_state_test, padd
ed_docs_project_grade_category_test, padded_docs_clean_categories_test, padded_docs_clean_subcategor
ies_test, padded_docs_teacher_prefix_test, numerical_df_test],
----> 2 [labels_test], verbose=0)
      3 print('Accuracy: %f' % (accuracy*100))

~/anaconda3/lib/python3.7/site-packages/keras/engine/training.py in evaluate(self, x, y, batch_size,
verbose, sample_weight, steps)
   1111                                         batch_size=batch_size,
   1112                                         verbose=verbose,
-> 1113                                         steps=steps)
   1114
   1115     def predict(self, x,

~/anaconda3/lib/python3.7/site-packages/keras/engine/training_arrays.py in test_loop(model, f, ins,
batch_size, verbose, steps)
    390                 ins_batch[i] = ins_batch[i].toarray()
    391
--> 392             batch_outs = f(ins_batch)
    393             if isinstance(batch_outs, list):
    394                 if batch_index == 0:

~/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py in __call__(self, inputs
)
   2713                 return self._legacy_call(inputs)
   2714
-> 2715             return self._call(inputs)
   2716         else:
   2717             if py_any(is_tensor(x) for x in inputs):

~/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py in _call(self, inputs)
   2673             fetched = self._callable_fn(*array_vals, run_metadata=self.run_metadata)
   2674         else:
-> 2675             fetched = self._callable_fn(*array_vals)
   2676         return fetched[:len(self.outputs)]
   2677

~/anaconda3/lib/python3.7/site-packages/tensorflow/python/client/session.py in __call__(self, *args,
**kwargs)
   1437           ret = tf_session.TF_SessionRunCallable(
   1438               self._session._session, self._handle, args, status,
-> 1439               run_metadata_ptr)
   1440         if run_metadata:
   1441           proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)

~/anaconda3/lib/python3.7/site-packages/tensorflow/python/framework/errors_impl.py in __exit__(self,
type_arg, value_arg, traceback_arg)
    526             None, None,
    527             compat.as_text(c_api.TF_Message(self.status.status)),
--> 528             c_api.TF_GetCode(self.status.status))
    529     # Delete the underlying status object from memory otherwise it stays alive
    530     # as there is a reference to status from this from the traceback due to

InvalidArgumentError: ValueError: Only one class present in y_true. ROC AUC score is not defined in
that case.
Traceback (most recent call last):

  File "/root/anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/script_ops.py", line 207,
in __call__
    ret = func(*args)

  File "/root/anaconda3/lib/python3.7/site-packages/sklearn/metrics/ranking.py", line 356, in roc_au
c_score
    sample_weight=sample_weight)

  File "/root/anaconda3/lib/python3.7/site-packages/sklearn/metrics/base.py", line 77, in _average_b
inary_score
    return binary_metric(y_true, y_score, sample_weight=sample_weight)

  File "/root/anaconda3/lib/python3.7/site-packages/sklearn/metrics/ranking.py", line 324, in _binar
y_roc_auc_score
    raise ValueError("Only one class present in y_true. ROC AUC score "

ValueError: Only one class present in y_true. ROC AUC score is not defined in that case.


         [[{{node metrics/auc_roc/PyFunc}}]]
```

In [ ]:

```
loss
```

In [ ]:

```
predictions=model.predict(x=[padded_docs_essay_test, padded_docs_school_state_test, padded_docs_project_grade_category_test, padded_docs_clean_categories_test, padded_docs_clean_subcategories_test, padded_docs_teacher_prefix_test, numerical_df_test])
```

In [ ]:

```
limit=predictions.shape[0]
y_pred=[]
for i in range(limit):
    if(predictions[i][0]<0.5):
        y_pred.append(0)
    else:
        y_pred.append(1)
```

Accuracy Score

In [ ]:

```
from sklearn import metrics
acc_score=metrics.accuracy_score(y_test,y_pred)
acc_score
```

AUC Score

In [ ]:

```
from sklearn import metrics
auc_score=metrics.roc_auc_score(y_test,y_pred)
auc_score
```

In [ ]:

```
import pdfkit
pdfkit.from_file('Test Pad - Model 1.html', 'Test Pad - Model 1.pdf')
```

In [ ]:

```
import numpy as np
from sklearn.metrics import roc_auc_score
y_true = np.array([0, 0, 0, 0])
y_scores = np.array([1, 0, 0, 0])
try:
    roc_auc_score(y_true, y_scores)
except ValueError:
    pass
```