Donors Choose - Model 2

In [1]:

```
# importing required libraries
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
from keras.layers import Input, Embedding, LSTM, Dropout, BatchNormalization, Dense, concat
from keras.preprocessing.text import Tokenizer, one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Model, load_model
from keras.utils import np_utils
from keras import regularizers
from keras.optimizers import *
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard, ReduceLROnPlateau

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split

import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
import re
from tqdm import tqdm
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import pickle
from sklearn.preprocessing import StandardScaler
from scipy.sparse import hstack
```

```
Using TensorFlow backend.
```

In [3]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clien
t_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.co
m&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2
F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.co
m%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.re
adonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&respon
se_type=code (https://accounts.google.com/o/oauth2/auth?client_id=9473189898
03-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=
urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleap
is.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:
··········
Mounted at /content/drive
```

In [4]:

```
# reading datasets
project_data = pd.read_csv("drive/My Drive/ML_data/preprocessed_data.csv")
project_data.head()
```

Out[4]:

	Unnamed: 0	Unnamed: 0.1	id	teacher_id	teacher_prefix	school_st
0	0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs	
1	1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr	
2	2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms	
3	3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs	
4	4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	mrs	

In [5]:

```
project_data["project_is_approved"].value_counts()
```

Out[5]:

```
1    92706
0    16542
Name: project_is_approved, dtype: int64
```

In [6]:

```
project_data.columns
```

Out[6]:

```
Index(['Unnamed: 0', 'Unnamed: 0.1', 'id', 'teacher_id', 'teacher_prefix',
       'school_state', 'project_submitted_datetime', 'project_grade_categor
y',
       'project_subject_categories', 'project_subject_subcategories',
       'project_title', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'essay', 'price', 'quantity', 'std_price', 'nrm_price',
       'project_summary_numerical'],
      dtype='object')
```

In [7]:

```
# checking for null values
project_data.isnull().sum()
```

Out[7]:

```
Unnamed: 0                                           0
Unnamed: 0.1                                         0
id                                                   0
teacher_id                                           0
teacher_prefix                                       0
school_state                                         0
project_submitted_datetime                           0
project_grade_category                               0
project_subject_categories                           0
project_subject_subcategories                        0
project_title                                       43
project_essay_1                                      0
project_essay_2                                      0
project_essay_3                                 105490
project_essay_4                                 105490
project_resource_summary                             0
teacher_number_of_previously_posted_projects         0
project_is_approved                                  0
essay                                                0
price                                                0
quantity                                             0
std_price                                            0
nrm_price                                            0
project_summary_numerical                            0
dtype: int64
```

In [0]:

```
# filling the null values with ''
project_data['project_title'] = project_data['project_title'].fillna('')
```

In [0]:

```
# combining essay and project_title columns
project_data['cleaned_text'] = project_data['essay'] + project_data['project_title']
```

In [0]:

```
# dropping unecessary columns
project_data = project_data.drop(['Unnamed: 0', 'Unnamed: 0.1', 'id', 'teacher_id', 'projec
                                  'project_resource_summary', 'std_price', 'nrm_price', 'pr
                                  'project_essay_3', 'project_essay_4', 'essay', 'project_t
```

In [11]:

```
# columns left after dropping unecessary columns
project_data.columns
```

Out[11]:

```
Index(['teacher_prefix', 'school_state', 'project_grade_category',
       'project_subject_categories', 'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'price', 'quantity', 'project_summary_numerical', 'cleaned_text'],
      dtype='object')
```

In [0]:

```
target = project_data['project_is_approved']
features = project_data.drop(['project_is_approved'], axis=1)
```

In [0]:

```
# splitting the dataset into train(75%) and test(25%) set
X_train, X_test, y_train, y_test = train_test_split(features, target, stratify=target, test
```

In [14]:

```
print('Shape of Train data', X_train.shape)
print('Shape of Test data', X_test.shape)
```

```
Shape of Train data (81936, 10)
Shape of Test data (27312, 10)
```

Filtering Text Data (essays & project_title) based on idf values

In [0]:

```
tfidf = TfidfVectorizer()
combine_tfidf = tfidf.fit_transform(X_train['cleaned_text'])

# converting to dictionary
combine_dict = dict(zip(tfidf.get_feature_names(),list(tfidf.idf_)))
```

In [0]:

```
tfidf_df = pd.DataFrame(list(combine_dict.items()), columns=['Words', 'IDF Values'])
tfidf_df = tfidf_df.sort_values(by ='IDF Values' )
```

In [17]:

```
# finding the min & max idf values
print(tfidf_df['IDF Values'].min())
print(tfidf_df['IDF Values'].max())
```

```
1.0074234116042697
11.62055875771544
```

In [18]:

```
# based on the idf values we prepare the corpus, thereby leaving the words with lower idf v
corpus = tfidf_df[(tfidf_df['IDF Values'] >= 2) & (tfidf_df['IDF Values'] <=11 )]
corpus.shape
```

Out[18]:

```
(27769, 2)
```

In [19]:

```
vocab = corpus["Words"].tolist()
vocab[:10]
```

Out[19]:

```
['new',
 'year',
 'one',
 'would',
 'time',
 'student',
 'want',
 'skills',
 'grade',
 'reading']
```

Tokenizing the Text

In [0]:

```
# convert the sentences (strings) into integers
tokenizer = Tokenizer()
tokenizer.fit_on_texts(vocab)
sequences_train = tokenizer.texts_to_sequences(X_train['cleaned_text'])
sequences_test = tokenizer.texts_to_sequences(X_test['cleaned_text'])
```

In [21]:

```
# get word -> integer mapping
word2idx = tokenizer.word_index
print('Found %s unique tokens.' % len(word2idx))
```

```
Found 27769 unique tokens.
```

Padding the sequences

The sequences have different lengths and Keras prefers inputs to be vectorized and all inputs to have the same length. We will pad all input sequences to have the length of 250

In [22]:

```
encoded_train = pad_sequences(sequences_train, maxlen=250, padding='post', truncating='post
print('Shape of data tensor:', encoded_train.shape)
```

```
Shape of data tensor: (81936, 250)
```

In [23]:

```
encoded_test = pad_sequences(sequences_test, maxlen=250,padding='post', truncating='post')
print('Shape of data tensor:', encoded_test.shape)
```

```
Shape of data tensor: (27312, 250)
```

Getting the vector representation using Glove vectors

In [0]:

```
# Loading Embedding File
pickle_in = open('drive/My Drive/ML_data/glove_vectors','rb')
glove_words = pickle.load(pickle_in)
```

In [0]:

```
num_words = len(word2idx) + 1
embedding_matrix = np.zeros((num_words, 300))
for word, i in word2idx.items():
    if i < len(vocab):
        embedding_vector = glove_words.get(word)
        if embedding_vector is not None:
          # words not found in embedding index will be all zeros.
          embedding_matrix[i] = embedding_vector
```

In [26]:

```
print(num_words)
print('--------------')
print(embedding_matrix.shape)
```

```
27770
--------------
(27770, 300)
```

Vectorizing all the categorical features using CountVectorizer

In [0]:

```
vect = CountVectorizer(binary=True)

train_prefix = vect.fit_transform(X_train["teacher_prefix"])
test_prefix = vect.transform(X_test["teacher_prefix"])
```

In [0]:

```
vect = CountVectorizer(binary=True)

train_state = vect.fit_transform(X_train["school_state"])
test_state = vect.transform(X_test["school_state"])
```

In [0]:

```
vect = CountVectorizer(binary=True)

train_grade = vect.fit_transform(X_train["project_grade_category"])
test_grade = vect.transform(X_test["project_grade_category"])
```

In [0]:

```
vect = CountVectorizer(binary=True)

train_subcat = vect.fit_transform(X_train["project_subject_categories"])
test_subcat = vect.transform(X_test["project_subject_categories"])
```

In [0]:

```
vect = CountVectorizer(binary=True)

train_subcat_1 = vect.fit_transform(X_train["project_subject_subcategories"])
test_subcat_1 = vect.transform(X_test["project_subject_subcategories"])
```

Reshaping & Standardizing numerical features

In [0]:

```
num_train_1=X_train['project_summary_numerical'].values.reshape(-1, 1)
num_train_2=X_train['price'].values.reshape(-1, 1)
num_train_3=X_train['quantity'].values.reshape(-1, 1)
num_train_4=X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1)

num_test_1=X_test['project_summary_numerical'].values.reshape(-1, 1)
num_test_2=X_test['price'].values.reshape(-1, 1)
num_test_3=X_test['quantity'].values.reshape(-1, 1)
num_test_4=X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1)
```

In [0]:

```
# concatenating train numerical features
num_train=np.concatenate((num_train_1,num_train_2,num_train_3,num_train_4),axis=1)

# concatenating test numerical features
num_test=np.concatenate((num_test_1,num_test_2,num_test_3,num_test_4),axis=1)

# Standardizing the features
norm=StandardScaler()
norm_train=norm.fit_transform(num_train)
norm_test=norm.transform(num_test)
```

In [0]:

```
# concatencating categorical features
cat_train = hstack([train_prefix,train_state,train_grade,train_subcat,train_subcat_1]).tode
cat_test = hstack([test_prefix,test_state,test_grade,test_subcat,test_subcat_1]).todense()
```

In [0]:

```
# concatenating the numerical & categorical features
all_train = np.hstack((cat_train,norm_train))
all_test = np.hstack((cat_test,norm_test))
```

In [0]:

```
final_train = np.expand_dims(all_train,2)
final_test = np.expand_dims(all_test,2)
```

In [37]:

```
print(final_train.shape)
print('----------------')
print(final_test.shape)
```

```
(81936, 512, 1)
----------------
(27312, 512, 1)
```

Defining model architecture

In [38]:

```
# load pre-trained word embeddings into an Embedding layer
# note that we set trainable = False so as to keep the embeddings of fixed sized
embedding_layer = Embedding(
  num_words,
  300,
  weights=[embedding_matrix],
  input_length=250,
  trainable=False
)
input_text = Input(shape=(250,),name="input_text")
x = embedding_layer(input_text)
x = LSTM(100,recurrent_dropout=0.5,kernel_regularizer=regularizers.l2(0.001),return_sequenc
flatten_1 = Flatten()(x)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Ple
ase use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please u
se tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Plea
se use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:190: The name tf.get_default_session is deprecated.
Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please u
se tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:3733: 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`.
```

In [39]:

```
# Now will prepare all the remaining categorical features
# Teacher Prefix
no_of_unique_prefix  = X_train["teacher_prefix"].nunique()
embedding_size_prefix = int(min(np.ceil((no_of_unique_prefix)/2), 50 ))
print('Unique Categories:', no_of_unique_prefix, 'Embedding Size:', embedding_size_prefix)


# Defining Input and Embedding Layer for the same

input_prefix = Input(shape=(1,), name="teacher_prefix")
embedding_prefix = Embedding(no_of_unique_prefix, embedding_size_prefix, name="emb_pre", tr
flatten_2 = Flatten()(embedding_prefix)

lb = LabelEncoder()
encoder_prefix_train = lb.fit_transform(X_train["teacher_prefix"])
encoder_prefix_test = lb.transform(X_test["teacher_prefix"])
```

```
Unique Categories: 5 Embedding Size: 3
```

In [40]:

```
# School State
no_of_unique_state  = X_train["school_state"].nunique()
embedding_size_state= int(min(np.ceil((no_of_unique_state)/2), 50 ))
print('Unique Categories:', no_of_unique_state, 'Embedding Size:', embedding_size_state)


# Defining Input and Embedding Layer for the same

input_state = Input(shape=(1,), name="school_prefix")
embedding_state = Embedding(no_of_unique_state, embedding_size_state, name="emb_state", tra
flatten_3 = Flatten()(embedding_state)


encoder_state_train = lb.fit_transform(X_train["school_state"])
# encoder_state_cv = lb.transform(X_cv["school_state"])
encoder_state_test = lb.transform(X_test["school_state"])
```

```
Unique Categories: 51 Embedding Size: 26
```

In [41]:

```
# For project_grade_category
no_of_unique_grade  = X_train["project_grade_category"].nunique()
embedding_size_grade = int(min(np.ceil((no_of_unique_grade)/2), 50 ))
print('Unique Categories:', no_of_unique_grade, 'Embedding Size:', embedding_size_grade)


# Defining Input and Embedding Layer for the same

input_grade= Input(shape=(1,),name="grade_cat")
embedding_grade = Embedding(no_of_unique_grade, embedding_size_grade, name="emb_grade", tra
flatten_4 = Flatten()(embedding_grade)


encoder_grade_train = lb.fit_transform(X_train["project_grade_category"])
# encoder_grade_cv = lb.transform(X_cv["project_grade_category"])
encoder_grade_test = lb.transform(X_test["project_grade_category"])
```

```
Unique Categories: 4 Embedding Size: 2
```

In [42]:

```
# For project_subject_categories
no_of_unique_subcat  = X_train["project_subject_categories"].nunique()
embedding_size_subcat = int(min(np.ceil((no_of_unique_subcat)/2), 50 ))
print('Unique Categories:', no_of_unique_subcat,'Embedding Size:', embedding_size_subcat)


# Defining Input and Embedding Layer for the same

input_subcat= Input(shape=(1,),name="sub_cat")
embedding_subcat = Embedding(no_of_unique_subcat,embedding_size_subcat,name="emb_subcat",tr
flatten_5 = Flatten()(embedding_subcat)

le = LabelEncoder()
le.fit(X_train["project_subject_categories"])
X_test["project_subject_categories"] = X_test["project_subject_categories"].map(lambda s: '

le.classes_ = np.append(le.classes_, '<unknown>')
encoder_subcat_train = le.transform(X_train["project_subject_categories"])
encoder_subcat_test= le.transform(X_test["project_subject_categories"])
```

```
Unique Categories: 51 Embedding Size: 26
```

In [43]:

```
# For project_subject_subcategories
no_of_unique_subcat_1  = X_train["project_subject_subcategories"].nunique()
embedding_size_subcat_1 = int(min(np.ceil((no_of_unique_subcat_1)/2), 50 ))
print('Unique Categories:', no_of_unique_subcat_1,'Embedding Size:', embedding_size_subcat_

# Defining Input and Embedding Layer for the same

input_subcat_1= Input(shape=(1,),name="sub_cat_1")
embedding_subcat_1 = Embedding(no_of_unique_subcat_1,embedding_size_subcat_1,name="emb_subc
flatten_6 = Flatten()(embedding_subcat_1)


le = LabelEncoder()
le.fit(X_train["project_subject_subcategories"])
X_test["project_subject_subcategories"] = X_test["project_subject_subcategories"].map(lambd

le.classes_ = np.append(le.classes_, '<unknown>')
encoder_subcat_1_train = le.transform(X_train["project_subject_subcategories"])
encoder_subcat_1_test= le.transform(X_test["project_subject_subcategories"])
```

```
Unique Categories: 397 Embedding Size: 50
```

In [0]:

```
# Defining the Input and Embedding Layer for the same

num_feats = Input(shape=(4,),name="numerical_features")
num_feats_ = Dense(100,activation="relu",kernel_initializer="he_normal")(num_feats)
```

In [45]:

```
print("Building Model-2")
x_concatenate = concatenate([flatten_1, flatten_2, flatten_3, flatten_4, flatten_5, flatter

x = Dense(128,activation="relu", kernel_initializer="he_normal",kernel_regularizer=regulari

x=Dropout(0.5)(x)

x = Dense(256,activation="relu",kernel_initializer="he_normal",kernel_regularizer=regulariz

x=Dropout(0.5)(x)

x = Dense(64,activation="relu", kernel_initializer="he_normal",kernel_regularizer=regulariz

output = Dense(2, activation='softmax', name='output')(x)
model_2 = Model(inputs=[input_text, input_prefix, input_state, input_grade,
                        input_subcat, input_subcat_1, num_feats],outputs=[output])
```

```
Building Model-2
```

In [0]:

```
train_data_2 = [encoded_train,encoder_prefix_train,encoder_state_train,
              encoder_grade_train,encoder_subcat_train,encoder_subcat_1_train,norm_train]
test_data_2 = [encoded_test,encoder_prefix_test,encoder_state_test,encoder_grade_test,
             encoder_subcat_test,encoder_subcat_1_test,norm_test]

Y_train = np_utils.to_categorical(y_train, 2)
Y_test = np_utils.to_categorical(y_test, 2)
```

In [0]:

```
checkpoint_2 = ModelCheckpoint("model_2.h5",
                             monitor="val_loss",
                             mode="min",
                             save_best_only = True,
                             verbose=1)

earlystop_2 = EarlyStopping(monitor = 'val_loss',
                            mode="min",
                            min_delta = 0,
                            patience = 5,
                            verbose = 1,
                            restore_best_weights = True)

reduce_lr_2 = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.2, patience = 2, verbose =

tensorboard_2 = TensorBoard(log_dir='graph_2', histogram_freq=0, batch_size=512, write_grap

callbacks_2 = [checkpoint_2,earlystop_2,tensorboard_2,reduce_lr_2]
```

In [0]:

```
# Defining Custom ROC-AUC function
from sklearn.metrics import roc_auc_score

def auc1(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score(y_true, y_pred)

def auroc(y_true, y_pred):
    return tf.py_func(auc1, (y_true, y_pred), tf.double)
```

In [0]:

```
adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

In [50]:

```
model_2.compile(optimizer=adam, loss='categorical_crossentropy', metrics=[auroc])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimiz
ers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.
v1.train.Optimizer instead.

WARNING:tensorflow:From <ipython-input-48-a7e6cba44e56>:10: py_func (from te
nsorflow.python.ops.script_ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function 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.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.
```

In [51]:

```
history_2 = model_2.fit(train_data_2, Y_train, batch_size=512,
                        epochs=30, validation_data=(test_data_2,Y_test), callbacks=callback
```

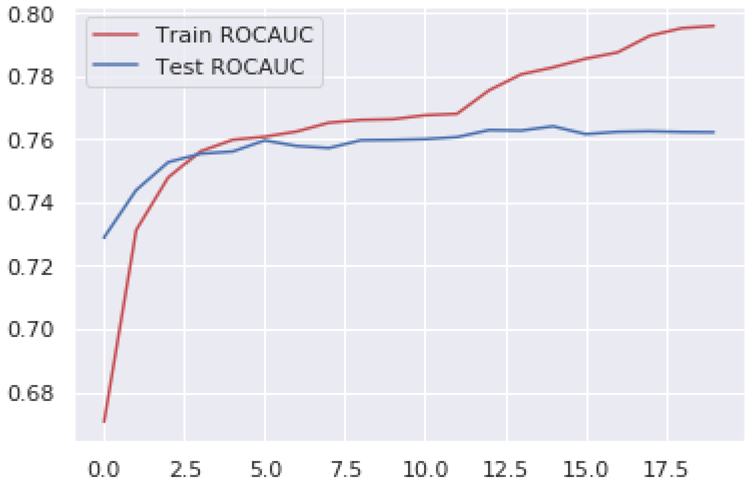
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/py
thon/ops/math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tens
orflow.python.ops.array_ops) is deprecated and will be removed in a future v
ersion.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 81936 samples, validate on 27312 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbac
ks.py:1122: The name tf.summary.merge_all is deprecated. Please use tf.compa
t.v1.summary.merge_all instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbac
ks.py:1125: The name tf.summary.FileWriter is deprecated. Please use tf.comp
at.v1.summary.FileWriter instead.

Epoch 1/30
81936/81936 [==============================] - 75s 910us/step - loss: 1.0141
- auroc: 0.6706 - val_loss: 0.6913 - val_auroc: 0.7289

Epoch 00001: val_loss improved from inf to 0.69125, saving model to model_2.
h5
Epoch 2/30
81936/81936 [==============================] - 73s 896us/step - loss: 0.5739
- auroc: 0.7312 - val_loss: 0.5311 - val_auroc: 0.7439

Epoch 00002: val_loss improved from 0.69125 to 0.53105, saving model to mode
l_2.h5
Epoch 3/30
81936/81936 [==============================] - 73s 896us/step - loss: 0.4681
- auroc: 0.7479 - val_loss: 0.4472 - val_auroc: 0.7527

Epoch 00003: val_loss improved from 0.53105 to 0.44721, saving model to mode
l_2.h5
Epoch 4/30
81936/81936 [==============================] - 74s 902us/step - loss: 0.4277
- auroc: 0.7562 - val_loss: 0.4180 - val_auroc: 0.7554

Epoch 00004: val_loss improved from 0.44721 to 0.41802, saving model to mode
l_2.h5
Epoch 5/30
81936/81936 [==============================] - 73s 892us/step - loss: 0.4077
- auroc: 0.7598 - val_loss: 0.4179 - val_auroc: 0.7561

Epoch 00005: val_loss improved from 0.41802 to 0.41786, saving model to mode
l_2.h5
Epoch 6/30
81936/81936 [==============================] - 73s 894us/step - loss: 0.3976
- auroc: 0.7608 - val_loss: 0.4140 - val_auroc: 0.7596

Epoch 00006: val_loss improved from 0.41786 to 0.41397, saving model to mode
l_2.h5
Epoch 7/30
81936/81936 [==============================] - 74s 898us/step - loss: 0.3934
- auroc: 0.7625 - val_loss: 0.3929 - val_auroc: 0.7578

Epoch 00007: val_loss improved from 0.41397 to 0.39290, saving model to mode
```

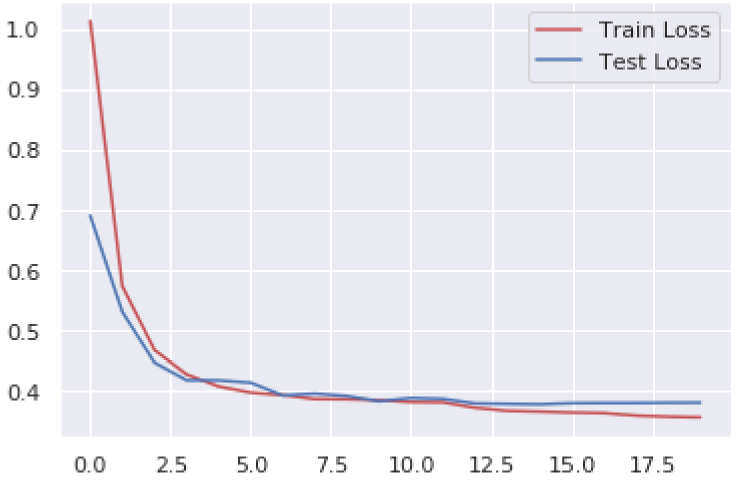
```
l_2.h5
Epoch 8/30
81936/81936 [==============================] - 74s 905us/step - loss: 0.3869
- auroc: 0.7652 - val_loss: 0.3960 - val_auroc: 0.7572

Epoch 00008: val_loss did not improve from 0.39290
Epoch 9/30
81936/81936 [==============================] - 74s 905us/step - loss: 0.3866
- auroc: 0.7661 - val_loss: 0.3919 - val_auroc: 0.7597

Epoch 00009: val_loss improved from 0.39290 to 0.39190, saving model to mode
l_2.h5
Epoch 10/30
81936/81936 [==============================] - 74s 906us/step - loss: 0.3849
- auroc: 0.7663 - val_loss: 0.3831 - val_auroc: 0.7598

Epoch 00010: val_loss improved from 0.39190 to 0.38313, saving model to mode
l_2.h5
Epoch 11/30
81936/81936 [==============================] - 75s 911us/step - loss: 0.3820
- auroc: 0.7676 - val_loss: 0.3885 - val_auroc: 0.7600

Epoch 00011: val_loss did not improve from 0.38313
Epoch 12/30
81936/81936 [==============================] - 75s 910us/step - loss: 0.3813
- auroc: 0.7680 - val_loss: 0.3870 - val_auroc: 0.7607

Epoch 00012: val_loss did not improve from 0.38313

Epoch 00012: ReduceLROnPlateau reducing learning rate to 0.00020000000949949
026.
Epoch 13/30
81936/81936 [==============================] - 74s 904us/step - loss: 0.3723
- auroc: 0.7755 - val_loss: 0.3796 - val_auroc: 0.7629

Epoch 00013: val_loss improved from 0.38313 to 0.37965, saving model to mode
l_2.h5
Epoch 14/30
81936/81936 [==============================] - 74s 902us/step - loss: 0.3673
- auroc: 0.7806 - val_loss: 0.3788 - val_auroc: 0.7627

Epoch 00014: val_loss improved from 0.37965 to 0.37884, saving model to mode
l_2.h5
Epoch 15/30
81936/81936 [==============================] - 74s 906us/step - loss: 0.3660
- auroc: 0.7828 - val_loss: 0.3781 - val_auroc: 0.7641

Epoch 00015: val_loss improved from 0.37884 to 0.37815, saving model to mode
l_2.h5
Epoch 16/30
81936/81936 [==============================] - 75s 914us/step - loss: 0.3644
- auroc: 0.7855 - val_loss: 0.3800 - val_auroc: 0.7616

Epoch 00016: val_loss did not improve from 0.37815
Epoch 17/30
81936/81936 [==============================] - 74s 908us/step - loss: 0.3635
- auroc: 0.7875 - val_loss: 0.3804 - val_auroc: 0.7624

Epoch 00017: val_loss did not improve from 0.37815

Epoch 00017: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e
```

```
-05.
Epoch 18/30
81936/81936 [==============================] - 74s 903us/step - loss: 0.3594
- auroc: 0.7928 - val_loss: 0.3804 - val_auroc: 0.7625

Epoch 00018: val_loss did not improve from 0.37815
Epoch 19/30
81936/81936 [==============================] - 74s 902us/step - loss: 0.3575
- auroc: 0.7952 - val_loss: 0.3807 - val_auroc: 0.7623

Epoch 00019: val_loss did not improve from 0.37815

Epoch 00019: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-
06.
Epoch 20/30
81936/81936 [==============================] - 74s 903us/step - loss: 0.3565
- auroc: 0.7959 - val_loss: 0.3809 - val_auroc: 0.7622

Epoch 00020: val_loss did not improve from 0.37815
Restoring model weights from the end of the best epoch
Epoch 00020: early stopping
```

In [52]:

```
sns.set()
plt.plot(history_2.history['auroc'], 'r')
plt.plot(history_2.history['val_auroc'], 'b')
plt.legend({'Train ROCAUC': 'r', 'Test ROCAUC':'b'})
plt.show()


plt.plot(history_2.history['loss'], 'r')
plt.plot(history_2.history['val_loss'], 'b')
plt.legend({'Train Loss': 'r', 'Test Loss':'b'})
plt.show()
```





AUC Score for Model 2 - 0.7622