DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [0]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter("ignore")
warnings.warn("ignore")
import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn import model_selection
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os
```

In [0]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc
from sklearn import preprocessing
```

In [4]:

```
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers.core import Activation, Dropout, Dense ,Reshape
from keras.layers import Flatten, LSTM,Lambda
from keras.models import Model
from keras.layers.embeddings import Embedding
from keras.preprocessing.text import Tokenizer
from keras.layers import Input
from keras.layers import Concatenate
from keras.utils import to_categorical
from keras.layers import Conv1D, MaxPooling1D
```

```
Using TensorFlow backend.
```

1.1 Reading Data

In [0]:

```
#https://stackabuse.com/python-for-nlp-creating-multi-data-type-classification-models-with-keras/
#https://www.pyimagesearch.com/2019/01/21/regression-with-keras/
#https://github.com/mmortazavi/EntityEmbedding-Working_Example/blob/master/EntityEmbedding.ipynb
#https://www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/
#https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classifi
cation/
```

In [0]:

```
preprocessed_data = pd.read_csv('preprocessed_data.csv')
```

In [6]:

```
print("Number of data points in preprocessed data", preprocessed_data.shape)
```

```
Number of data points in preprocessed data (109248, 9)
Number of data points in preprocessed data (109248, 9)
```

In [7]:

```
preprocessed_data=preprocessed_data.sample(n=100000)
preprocessed_data.head()
```

Out[7]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	project_is_approved	clean
6876	ny	ms	grades_prek_2	16	1	m
105702	wv	ms	grades_9_12	6	1	h
62830	ca	mrs	grades_3_5	0	1	litera
46841	tn	ms	grades_prek_2	0	1	litera m
30042	wi	mr	grades_6_8	3	1	

In [0]:

```
X=preprocessed_data.drop(columns=['project_is_approved'],axis=1)
y=preprocessed_data['project_is_approved']
```

In [0]:

```
label_encoder = preprocessing.LabelEncoder()
y = label_encoder.fit_transform(y)
```

In [0]:

```
X_1, X_test, y_1, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=0,str
atify=y)

# split the train data set into cross validation train and cross validation test
X_train, X_cv, y_train, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.2, random_sta
te=0,stratify=y_1)
```

In [0]:

```
y_train = to_categorical(y_train)
y_cv    = to_categorical(y_cv)
y_test = to_categorical(y_test)
```

In [0]:

```
tokenizer = Tokenizer()
```

```
tokenizer.fit_on_texts(X_train['essay'].values)

X1_tr = np.array(tokenizer.texts_to_sequences(X_train['essay'].values))
X1_cv = np.array(tokenizer.texts_to_sequences(X_cv['essay'].values))
X1_test = np.array(tokenizer.texts_to_sequences(X_test['essay'].values))
```

In [0]:

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

maxlen = 200

X1_tr = pad_sequences(X1_tr, padding='post', maxlen=maxlen)
X1_cv = pad_sequences(X1_cv, padding='post', maxlen=maxlen)
X1_test = pad_sequences(X1_test, padding='post', maxlen=maxlen)
```

In [14]:

```
print(X1_tr.shape)
print(X1_cv.shape)
print(X1_test.shape)
```

```
(64000, 200)
(16000, 200)
(20000, 200)
```

In [0]:

```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [0]:

```
embeddings_dictionary = dict()
for word in glove_words:
    vector_dimensions = model[word]
    embeddings_dictionary [word] = vector_dimensions
```

In [0]:

```
embedding_matrix = np.zeros((vocab_size, 300))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

In [18]:

```
embedding_matrix.shape
```

Out[18]:

```
(45666, 300)
```

In [19]:

```
input_1 = Input(shape=(maxlen,),name='essay_input')
print(input_1.shape)
input_1_embedding = Embedding(vocab_size, 300, weights=[embedding_matrix], trainable=False )(input_
1)
print(input_1_embedding.shape)
input_1_lstm = LSTM(128,return_sequences=True)(input_1_embedding)
print(input_1_lstm.shape)
input_1_flatten=Flatten()(input_1_lstm)
print(input_1_flatten.shape)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please us
```

```
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please us
e tf.compat.v1.placeholder instead.

(?, 200)
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Plea
se use tf.compat.v1.get_default_graph instead.

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

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

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is deprecated. Please use tf
.compat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is deprecated. Plea
se use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initialized is
deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer is deprecated.
Please use tf.compat.v1.variables_initializer instead.

(?, 200, 300)
(?, ?, 128)
(?, ?)
```

In [0]:

```
categoricals=['school_state','teacher_prefix','project_grade_category','clean_categories','clean_su
bcategories']
numericals=['teacher_number_of_previously_posted_projects','price']
```

In [21]:

```
embed_cols=[i for i in X_train[categoricals]]

for i in embed_cols:
    print(i,X_train[i].nunique())
```

```
school_state 51
teacher_prefix 5
project_grade_category 4
clean_categories 51
clean_subcategories 384
```

In [0]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [23]:

```
vectorizer = CountVectorizer()
X2_tr = vectorizer.fit_transform(X_train['school_state'].values).toarray()
X2_cv = vectorizer.transform(X_cv['school_state'].values).toarray()
X2_test = vectorizer.transform(X_test['school_state'].values).toarray()
cat_emb_name= 'school_state_Embedding'
no_of_unique_cat  = X_train['school_state'].nunique()
embedding_size = int(min(np.ceil((no_of_unique_cat)/2), 50 ))
input_2 = Input(shape=(X2_tr.shape[1],),name='school_state_input')
```

```
print(input_2.shape)
input_2_embedding = Embedding(no_of_unique_cat, embedding_size,input_length=X2_tr.shape[1],
name=cat_emb_name)(input_2)
print(input_2_embedding.shape)
input_2_flatten=Flatten()(input_2_embedding)
print(input_2_flatten.shape)
```

```
(?, 51)
(?, 51, 26)
(?, ?)
```

In [24]:

```
print(X2_tr.shape)
print(X2_cv.shape)
print(X2_test.shape)
```

```
(64000, 51)
(16000, 51)
(20000, 51)
```

In [25]:

```
vectorizer = CountVectorizer()
X3_tr = vectorizer.fit_transform(X_train['teacher_prefix'].values.astype('U')).toarray()
X3_cv = vectorizer.transform(X_cv['teacher_prefix'].values.astype('U')).toarray()
X3_test = vectorizer.transform(X_test['teacher_prefix'].values.astype('U')).toarray()
cat_emb_name= 'teacher_prefix_Embedding'
no_of_unique_cat  = X_train['teacher_prefix'].nunique()
embedding_size = int(min(np.ceil((no_of_unique_cat)/2), 50 ))
input_3 = Input(shape=(X3_tr.shape[1],),name='teacher_prefix_input')
print(input_3.shape)
input_3_embedding = Embedding(no_of_unique_cat, embedding_size,input_length=X3_tr.shape[1],
name=cat_emb_name)(input_3)
print(input_3_embedding.shape)
input_3_flatten=Flatten()(input_3_embedding)
print(input_3_flatten.shape)
```

```
(?, 5)
(?, 5, 3)
(?, ?)
```

In [25]:

```
print(X3_tr.shape)
print(X3_cv.shape)
print(X3_test.shape)
```

```
(64000, 5)
(16000, 5)
(20000, 5)
```

In [26]:

```
vectorizer = CountVectorizer()
X4_tr = vectorizer.fit_transform(X_train['project_grade_category'].values).toarray()
X4_cv = vectorizer.transform(X_cv['project_grade_category'].values).toarray()
X4_test = vectorizer.transform(X_test['project_grade_category'].values).toarray()
cat_emb_name= 'project_grade_category_Embedding'
no_of_unique_cat  = X_train['project_grade_category'].nunique()
embedding_size = int(min(np.ceil((no_of_unique_cat)/2), 50 ))
input_4 = Input(shape=(X4_tr.shape[1],),name='project_grade_category_input')
print(input_4.shape)
input_4_embedding = Embedding(no_of_unique_cat,
embedding_size,input_length=X4_tr.shape[1],name=cat_emb_name)(input_4)
print(input_4_embedding.shape)
input_4_flatten=Flatten()(input_4_embedding)
print(input_4_flatten.shape)
```

```
(?, 4)
(?, 4, 2)
(?, ?)
```

In [27]:

```
print(X4_tr.shape)
print(X4_cv.shape)
print(X4_test.shape)
```

```
(64000, 4)
(16000, 4)
(20000, 4)
```

In [27]:

```
vectorizer = CountVectorizer()
X5_tr = vectorizer.fit_transform(X_train['clean_categories'].values).toarray()
X5_cv = vectorizer.transform(X_cv['clean_categories'].values).toarray()
X5_test = vectorizer.transform(X_test['clean_categories'].values).toarray()
cat_emb_name= 'clean_categories_Embedding'
no_of_unique_cat  = X_train['clean_categories'].nunique()
embedding_size = int(min(np.ceil((no_of_unique_cat)/2), 50 ))
input_5 = Input(shape=(X5_tr.shape[1],),name='clean_categories_input')
print(input_5.shape)
input_5_embedding = Embedding(no_of_unique_cat, embedding_size,input_length=X5_tr.shape[1],
name=cat_emb_name)(input_5)
print(input_5_embedding.shape)
input_5_flatten=Flatten()(input_5_embedding)
print(input_5_flatten.shape)
```

```
(?, 9)
(?, 9, 26)
(?, ?)
```

In [29]:

```
print(X5_tr.shape)
print(X5_cv.shape)
print(X5_test.shape)
```

```
(64000, 9)
(16000, 9)
(20000, 9)
```

In [28]:

```
vectorizer = CountVectorizer()
X6_tr = vectorizer.fit_transform(X_train['clean_subcategories'].values).toarray()
X6_cv = vectorizer.transform(X_cv['clean_subcategories'].values).toarray()
X6_test = vectorizer.transform(X_test['clean_subcategories'].values).toarray()
cat_emb_name= 'clean_subcategories_Embedding'
no_of_unique_cat  = X_train['clean_subcategories'].nunique()
embedding_size = int(min(np.ceil((no_of_unique_cat)/2), 50 ))
input_6 = Input(shape=(X6_tr.shape[1],),name='clean_subcategories_input')
print(input_6.shape)
input_6_embedding = Embedding(no_of_unique_cat, embedding_size,input_length=X6_tr.shape[1],
name=cat_emb_name)(input_6)
print(input_6_embedding.shape)
input_6_flatten=Flatten()(input_6_embedding)
print(input_6_flatten.shape)
```

```
(?, 30)
(?, 30, 50)
(?, ?)
```

In [31]:

```
print(X6_tr.shape)
print(X6_cv.shape)
print(X6_test.shape)
```

```
(64000, 30)
(16000, 30)
(20000, 30)
```

In [29]:

```
X7_tr = preprocessing.normalize(X_train[['teacher_number_of_previously_posted_projects', 'price']]
)
X7_cv = preprocessing.normalize(X_cv[['teacher_number_of_previously_posted_projects', 'price']])
X7_test = preprocessing.normalize(X_test[['teacher_number_of_previously_posted_projects', 'price']
])
input_7 = Input(shape=(len(X_train[numericals].columns),),name='numerical_input')
print(input_7.shape)
input_7_dense = Dense(128)(input_7)
print(input_7_dense.shape)
```

```
(?, 2)
(?, 128)
```

In [33]:

```
print(X7_tr.shape)
print(X7_cv.shape)
print(X7_test.shape)
```

```
(64000, 2)
(16000, 2)
(20000, 2)
```

In [0]:

```
from keras.regularizers import l2
#At the end we concatenate altogther and add other Dense layers
output_1 = Concatenate()
([input_1_flatten,input_2_flatten,input_3_flatten,input_4_flatten,input_5_flatten,input_6_flatten,
input_7_dense])
output_1 = Dense(256,activation='relu',kernel_initializer='he_normal')(output_1)
output_1= Dropout(0.2)(output_1)
output_1 = Dense(128,activation='relu',kernel_initializer='he_normal')(output_1)
output_1= Dropout(0.2)(output_1)
output_1 = Dense(64,activation='relu',kernel_initializer='he_normal')(output_1)
output_1= Dropout(0.3)(output_1)
output_1 = Dense(2, activation='softmax')(output_1)
```

In [0]:

```
#https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-ro
c-and-auc-in-keras
import tensorflow as tf
def auroc(y_true, y_pred):
    return tf.py_func(roc_auc_score, (y_true, y_pred), tf.double)
```

In [142]:

```
from keras import optimizers
model = Model(inputs=[input_1,input_2,input_3,input_4,input_5,input_6,input_7], outputs=output_1)
model.compile(loss='binary_crossentropy', optimizer=optimizers.SGD(lr=0.01, decay=1e-3,momentum=0.9
) ,metrics=[auroc])
model.summary()
```

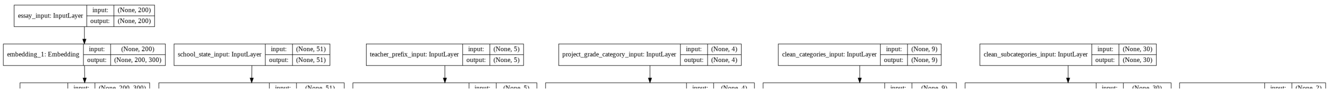
```
Model: "model_21"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to                     
==================================================================================================
essay_input (InputLayer)        (None, 200)          0                                            
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 200, 300)     13699800    essay_input[0][0]                
```

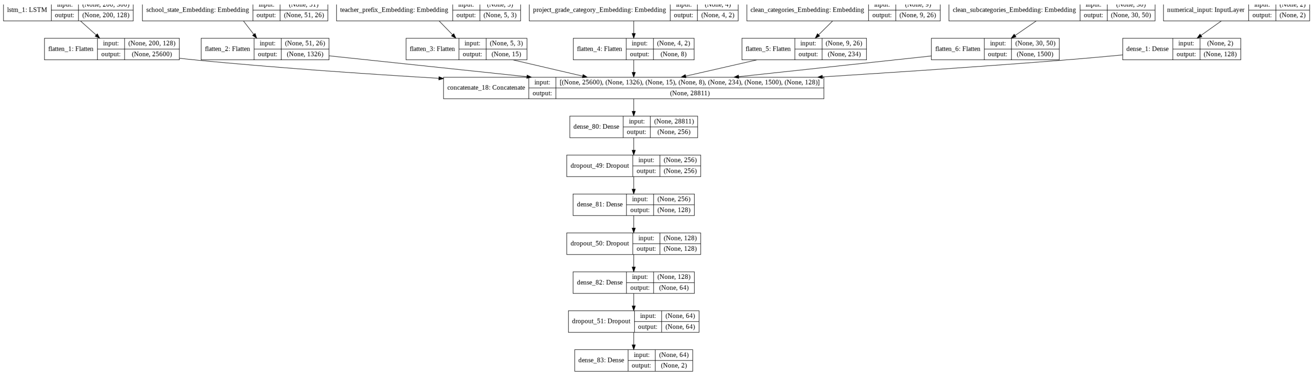
```
__________________________________________________________________________________________________
school_state_input (InputLayer) (None, 51)           0
__________________________________________________________________________________________________
teacher_prefix_input (InputLaye (None, 5)            0
__________________________________________________________________________________________________
project_grade_category_input (I (None, 4)            0
__________________________________________________________________________________________________
clean_categories_input (InputLa (None, 9)            0
__________________________________________________________________________________________________
clean_subcategories_input (Inpu (None, 30)           0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 200, 128)     219648      embedding_1[0][0]
__________________________________________________________________________________________________
school_state_Embedding (Embeddi (None, 51, 26)       1326        school_state_input[0][0]
__________________________________________________________________________________________________
teacher_prefix_Embedding (Embed (None, 5, 3)         15          teacher_prefix_input[0][0]
__________________________________________________________________________________________________
project_grade_category_Embeddin (None, 4, 2)         8           project_grade_category_input[0][0
__________________________________________________________________________________________________
clean_categories_Embedding (Emb (None, 9, 26)        1326        clean_categories_input[0][0]
__________________________________________________________________________________________________
clean_subcategories_Embedding ( (None, 30, 50)       19200       clean_subcategories_input[0][0]
__________________________________________________________________________________________________
numerical_input (InputLayer)    (None, 2)            0
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 25600)        0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 1326)         0           school_state_Embedding[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 15)           0           teacher_prefix_Embedding[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 8)            0           project_grade_category_Embedding[
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 234)          0           clean_categories_Embedding[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 1500)         0           clean_subcategories_Embedding[0][
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          384         numerical_input[0][0]
__________________________________________________________________________________________________
concatenate_18 (Concatenate)    (None, 28811)        0           flatten_1[0][0]
                                                                 flatten_2[0][0]
                                                                 flatten_3[0][0]
                                                                 flatten_4[0][0]
                                                                 flatten_5[0][0]
                                                                 flatten_6[0][0]
                                                                 dense_1[0][0]
__________________________________________________________________________________________________
dense_80 (Dense)                (None, 256)          7375872     concatenate_18[0][0]
__________________________________________________________________________________________________
dropout_49 (Dropout)            (None, 256)          0           dense_80[0][0]
__________________________________________________________________________________________________
dense_81 (Dense)                (None, 128)          32896       dropout_49[0][0]
__________________________________________________________________________________________________
dropout_50 (Dropout)            (None, 128)          0           dense_81[0][0]
__________________________________________________________________________________________________
dense_82 (Dense)                (None, 64)           8256        dropout_50[0][0]
__________________________________________________________________________________________________
dropout_51 (Dropout)            (None, 64)           0           dense_82[0][0]
__________________________________________________________________________________________________
dense_83 (Dense)                (None, 2)            130         dropout_51[0][0]
==================================================================================================
Total params: 21,358,861
Trainable params: 7,659,061
Non-trainable params: 13,699,800
__________________________________________________________________________________________________
```

In [143]:

```
from keras.utils import plot_model
plot_model(model, to_file='model_1.png', show_shapes=True, show_layer_names=True)
```

Out[143]:




lstm_1: LSTM
output: (None, 200, 128)
school_state_Embedding: Embedding
output: (None, 51, 26)
teacher_prefix_Embedding: Embedding
output: (None, 5, 3)
project_grade_category_Embedding: Embedding
output: (None, 4, 2)
clean_categories_Embedding: Embedding
output: (None, 9, 26)
clean_subcategories_Embedding: Embedding
output: (None, 30, 50)
numerical_input: InputLayer
output: (None, 2)
flatten_1: Flatten input: (None, 200, 128) output: (None, 25600)
flatten_2: Flatten input: (None, 51, 26) output: (None, 1326)
flatten_3: Flatten input: (None, 5, 3) output: (None, 15)
flatten_4: Flatten input: (None, 4, 2) output: (None, 8)
flatten_5: Flatten input: (None, 9, 26) output: (None, 234)
flatten_6: Flatten input: (None, 30, 50) output: (None, 1500)
dense_1: Dense input: (None, 2) output: (None, 128)
concatenate_18: Concatenate input: [(None, 25600), (None, 1326), (None, 15), (None, 8), (None, 234), (None, 1500), (None, 128)] output: (None, 28811)
dense_80: Dense input: (None, 28811) output: (None, 256)
dropout_49: Dropout input: (None, 256) output: (None, 256)
dense_81: Dense input: (None, 256) output: (None, 128)
dropout_50: Dropout input: (None, 128) output: (None, 128)
dense_82: Dense input: (None, 128) output: (None, 64)
dropout_51: Dropout input: (None, 64) output: (None, 64)
dense_83: Dense input: (None, 64) output: (None, 2)

In [144]:

```
history = model.fit(x=[X1_tr,X2_tr,X3_tr,X4_tr, X5_tr, X6_tr,X7_tr], y=y_train, validation_data=([X1_cv,X2_cv,X3_cv,X4_cv, X5_cv, X6_cv,X7_cv],y_cv),epochs=10,batch_size=300,verbose=2)
```

```
Train on 64000 samples, validate on 16000 samples
Epoch 1/10
 - 100s - loss: 0.3608 - auroc: 0.7872 - val_loss: 0.3691 - val_auroc: 0.7487
Epoch 2/10
 - 93s - loss: 0.3380 - auroc: 0.7965 - val_loss: 0.3683 - val_auroc: 0.7504
Epoch 3/10
 - 91s - loss: 0.3344 - auroc: 0.7991 - val_loss: 0.3689 - val_auroc: 0.7497
Epoch 4/10
 - 93s - loss: 0.3311 - auroc: 0.8028 - val_loss: 0.3729 - val_auroc: 0.7505
Epoch 5/10
 - 93s - loss: 0.3274 - auroc: 0.8054 - val_loss: 0.3686 - val_auroc: 0.7496
Epoch 6/10
 - 92s - loss: 0.3230 - auroc: 0.8095 - val_loss: 0.3722 - val_auroc: 0.7488
Epoch 7/10
 - 95s - loss: 0.3189 - auroc: 0.8114 - val_loss: 0.3690 - val_auroc: 0.7478
Epoch 8/10
 - 95s - loss: 0.3145 - auroc: 0.8131 - val_loss: 0.3732 - val_auroc: 0.7468
Epoch 9/10
 - 93s - loss: 0.3085 - auroc: 0.8172 - val_loss: 0.3751 - val_auroc: 0.7466
Epoch 10/10
 - 93s - loss: 0.3027 - auroc: 0.8201 - val_loss: 0.3809 - val_auroc: 0.7449
```

In [0]:

```
score = model.evaluate(x=[X1_test,X2_test,X3_test,X4_test, X5_test, X6_test, X7_test], y=y_test, verbose=2,batch_size=500)
```

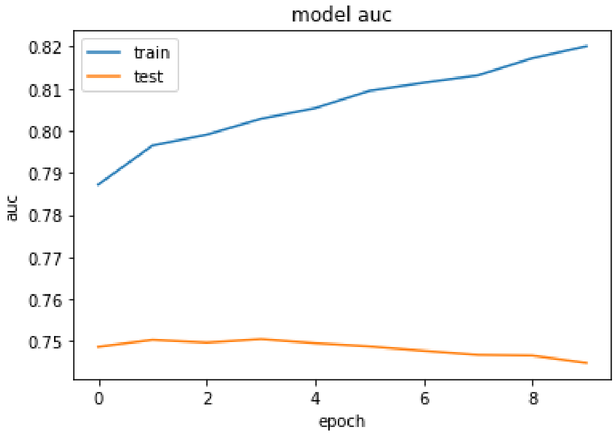
In [146]:

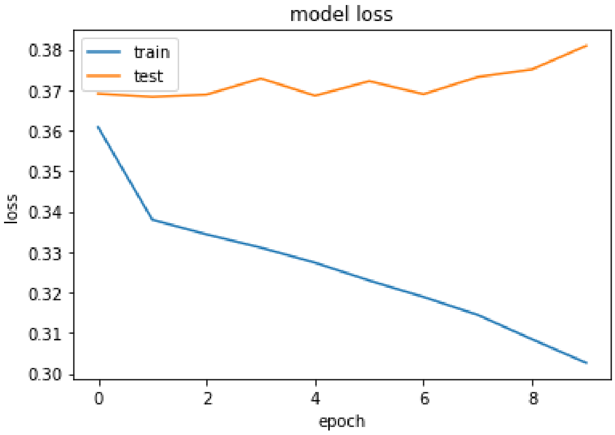
```
print("Test Loss:", score[0])
print("Test AUC:", score[1])
```

```
Test Loss: 0.38501315861940383
Test AUC: 0.7441678294901292
```

In [147]:

```
plt.plot(history.history['auroc'])
plt.plot(history.history['val_auroc'])

plt.title('model auc')
plt.ylabel('auc')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```





In [148]:

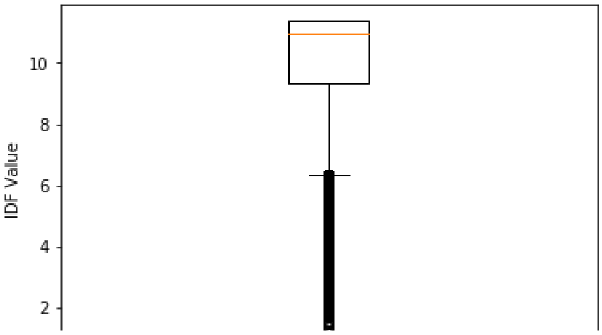
```
# serialize weights to HDF5
model.save_weights("model_1.h5")
print("Saved model to disk")
```

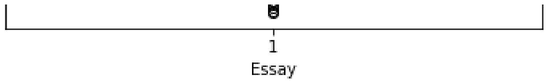
```
Saved model to disk
```

Model 2

In [149]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
vectorizer = TfidfVectorizer()
vectorizer.fit(X_train['essay'].values)
plt.boxplot(list(vectorizer.idf_))
plt.xlabel('Essay')
plt.ylabel('IDF Value')
plt.show()
```





In [150]:

```
tenth_percentile=np.quantile((vectorizer.idf_),0.10)
ninty_percentile=np.quantile((vectorizer.idf_),0.90)
print(tenth_percentile)
print(ninty_percentile)
dictionary = dict(zip(vectorizer.get_feature_names(), list(vectorizer.idf_)))
filterred_words=[]
for k,v in dictionary.items():
    if v > tenth_percentile and v < ninty_percentile:
        filterred_words.append(k)

len(filterred_words)
```

```
7.375306104990596
11.373506806659794
```

Out[150]:

```
23474
```

In [0]:

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(filterred_words)

X8_tr = np.array(tokenizer.texts_to_sequences(X_train['essay'].values))
X8_cv = np.array(tokenizer.texts_to_sequences(X_cv['essay'].values))
X8_test = np.array(tokenizer.texts_to_sequences(X_test['essay'].values))
```

In [0]:

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

maxlen = 200

X8_tr = pad_sequences(X8_tr, padding='post', maxlen=maxlen)
X8_cv = pad_sequences(X8_cv, padding='post', maxlen=maxlen)
X8_test = pad_sequences(X8_test, padding='post', maxlen=maxlen)
```

In [42]:

```
print(X8_tr.shape)
print(X8_cv.shape)
print(X8_test.shape)
```

```
(64000, 200)
(16000, 200)
(20000, 200)
```

In [0]:

```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [0]:

```
embeddings_dictionary = dict()
for word in glove_words:
    vector_dimensions = model[word]
    embeddings_dictionary [word] = vector_dimensions
```

In [0]:

```
embedding_matrix = np.zeros((vocab_size, 300))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

In [46]:

```
embedding_matrix.shape
```

Out[46]:

```
(23475, 300)
```

In [156]:

```
input_8 = Input(shape=(maxlen,),name='essay_tfidf_input')
print(input_8.shape)
input_8_embedding = Embedding(vocab_size, 300, weights=[embedding_matrix], trainable=False )(input_
8)
print(input_8_embedding.shape)
input_8_lstm = LSTM(128,return_sequences=True)(input_8_embedding)
print(input_8_lstm.shape)
input_8_flatten=Flatten()(input_8_lstm)
print(input_8_flatten.shape)
```

```
(?, 200)
(?, 200, 300)
(?, ?, 128)
(?, ?)
```

In [0]:

```
#At the end we concatenate altogther and add other Dense layers
output_2 = Concatenate()
([input_8_flatten,input_2_flatten,input_3_flatten,input_4_flatten,input_5_flatten,input_6_flatten,
input_7_dense])
output_2 = Dense(256, kernel_initializer="he_uniform",activation='relu',kernel_regularizer=l2(0.001
))(output_2)
output_2= Dropout(0.3)(output_2)
output_2 = Dense(128, kernel_initializer="he_uniform",activation='relu',kernel_regularizer=l2(0.001
))(output_2)
output_2= Dropout(0.5)(output_2)
output_2 = Dense(64, kernel_initializer="he_uniform", activation='relu',kernel_regularizer=l2(0.001
))(output_2)
output_2 = Dense(2, activation='softmax')(output_2)
```

In [158]:

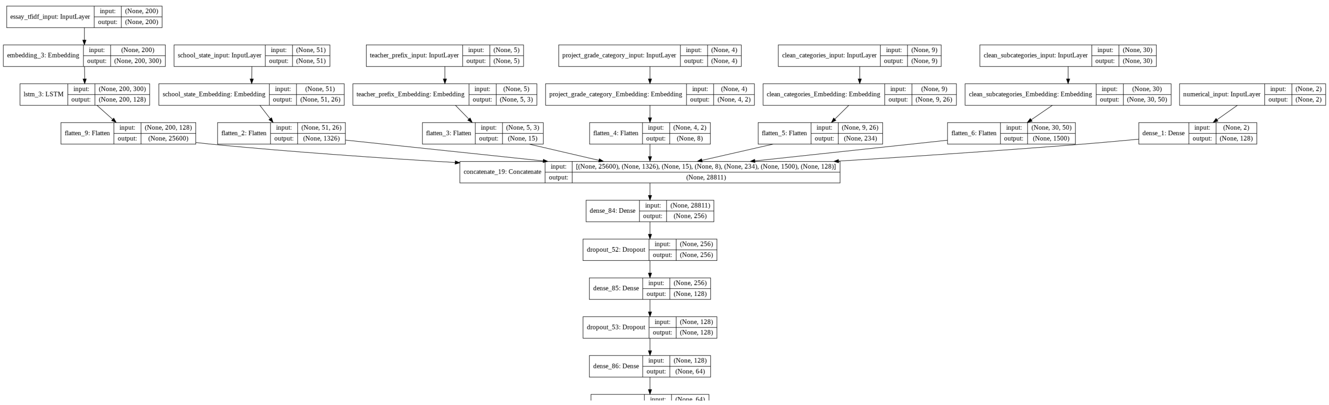
```
model_2 = Model(inputs=[input_8,input_2,input_3,input_4,input_5,input_6,input_7], outputs=output_2
)
model_2.compile(loss='binary_crossentropy', optimizer=optimizers.SGD(lr=0.01, decay=1e-3,momentum=0
.9),metrics=[auroc])
model_2.summary()
```

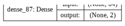
```
Model: "model_22"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
essay_tfidf_input (InputLayer)  (None, 200)          0
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 200, 300)     7042500     essay_tfidf_input[0][0]
__________________________________________________________________________________________________
school_state_input (InputLayer) (None, 51)           0
__________________________________________________________________________________________________
teacher_prefix_input (InputLaye (None, 5)            0
__________________________________________________________________________________________________
project_grade_category_input (I (None, 4)            0
__________________________________________________________________________________________________
clean_categories_input (InputLa (None, 9)            0
__________________________________________________________________________________________________
clean_subcategories_input (Inpu (None, 30)           0                                            
__________________________________________________________________________________________________
lstm_3 (LSTM)                   (None, 200, 128)     219648      embedding_3[0][0]                
__________________________________________________________________________________________________
school_state_Embedding (Embeddi (None, 51, 26)       1326        school_state_input[0][0]         
__________________________________________________________________________________________________
teacher_prefix_Embedding (Embed (None, 5, 3)         15          teacher_prefix_input[0][0]       
__________________________________________________________________________________________________
project_grade_category_Embeddin (None, 4, 2)         8           project_grade_category_input[0][0
__________________________________________________________________________________________________
clean_categories_Embedding (Emb (None, 9, 26)        1326        clean_categories_input[0][0]     
__________________________________________________________________________________________________
clean_subcategories_Embedding ( (None, 30, 50)       19200       clean_subcategories_input[0][0]  
__________________________________________________________________________________________________
numerical_input (InputLayer)    (None, 2)            0                                            
__________________________________________________________________________________________________
flatten_9 (Flatten)             (None, 25600)        0           lstm_3[0][0]                     
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 1326)         0           school_state_Embedding[0][0]     
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 15)           0           teacher_prefix_Embedding[0][0]   
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 8)            0           project_grade_category_Embedding[
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 234)          0           clean_categories_Embedding[0][0] 
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 1500)         0           clean_subcategories_Embedding[0][
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          384         numerical_input[0][0]            
__________________________________________________________________________________________________
concatenate_19 (Concatenate)    (None, 28811)        0           flatten_9[0][0]                  
                                                                 flatten_2[0][0]                  
                                                                 flatten_3[0][0]                  
                                                                 flatten_4[0][0]                  
                                                                 flatten_5[0][0]                  
                                                                 flatten_6[0][0]                  
                                                                 dense_1[0][0]                    
__________________________________________________________________________________________________
dense_84 (Dense)                (None, 256)          7375872     concatenate_19[0][0]             
__________________________________________________________________________________________________
dropout_52 (Dropout)            (None, 256)          0           dense_84[0][0]                   
__________________________________________________________________________________________________
dense_85 (Dense)                (None, 128)          32896       dropout_52[0][0]                 
__________________________________________________________________________________________________
dropout_53 (Dropout)            (None, 128)          0           dense_85[0][0]                   
__________________________________________________________________________________________________
dense_86 (Dense)                (None, 64)           8256        dropout_53[0][0]                 
__________________________________________________________________________________________________
dense_87 (Dense)                (None, 2)            130         dense_86[0][0]                   
==================================================================================================
Total params: 14,701,561
Trainable params: 7,659,061
Non-trainable params: 7,042,500
__________________________________________________________________________________________________
```

In [159]:

```
plot_model(model_2, to_file='model_2.png', show_shapes=True, show_layer_names=True)
```

Out[159]:


essay_tfidf_input: InputLayer | input: (None, 200) | output: (None, 200)
embedding_3: Embedding | input: (None, 200) | output: (None, 200, 300)
school_state_input: InputLayer | input: (None, 51) | output: (None, 51)
teacher_prefix_input: InputLayer | input: (None, 5) | output: (None, 5)
project_grade_category_input: InputLayer | input: (None, 4) | output: (None, 4)
clean_categories_input: InputLayer | input: (None, 9) | output: (None, 9)
clean_subcategories_input: InputLayer | input: (None, 30) | output: (None, 30)
lstm_3: LSTM | input: (None, 200, 300) | output: (None, 200, 128)
school_state_Embedding: Embedding | input: (None, 51) | output: (None, 51, 26)
teacher_prefix_Embedding: Embedding | input: (None, 5) | output: (None, 5, 3)
project_grade_category_Embedding: Embedding | input: (None, 4) | output: (None, 4, 2)
clean_categories_Embedding: Embedding | input: (None, 9) | output: (None, 9, 26)
clean_subcategories_Embedding: Embedding | input: (None, 30) | output: (None, 30, 50)
numerical_input: InputLayer | input: (None, 2) | output: (None, 2)
flatten_9: Flatten | input: (None, 200, 128) | output: (None, 25600)
flatten_2: Flatten | input: (None, 51, 26) | output: (None, 1326)
flatten_3: Flatten | input: (None, 5, 3) | output: (None, 15)
flatten_4: Flatten | input: (None, 4, 2) | output: (None, 8)
flatten_5: Flatten | input: (None, 9, 26) | output: (None, 234)
flatten_6: Flatten | input: (None, 30, 50) | output: (None, 1500)
dense_1: Dense | input: (None, 2) | output: (None, 128)
concatenate_19: Concatenate | input: [(None, 25600), (None, 1326), (None, 15), (None, 8), (None, 234), (None, 1500), (None, 128)] | output: (None, 28811)
dense_84: Dense | input: (None, 28811) | output: (None, 256)
dropout_52: Dropout | input: (None, 256) | output: (None, 256)
dense_85: Dense | input: (None, 256) | output: (None, 128)
dropout_53: Dropout | input: (None, 128) | output: (None, 128)
dense_86: Dense | input: (None, 128) | output: (None, 64)
input: (None, 64)



In [160]:

```
history = model_2.fit(x=[X8_tr,X2_tr,X3_tr,X4_tr, X5_tr, X6_tr,X7_tr], y=y_train, validation_data=(
[X8_cv,X2_cv,X3_cv,X4_cv, X5_cv, X6_cv,X7_cv],y_cv),epochs=10,batch_size=500,verbose=2)
```

```
Train on 64000 samples, validate on 16000 samples
Epoch 1/10
 - 68s - loss: 1.3157 - auroc: 0.5545 - val_loss: 1.2714 - val_auroc: 0.6000
Epoch 2/10
 - 62s - loss: 1.2558 - auroc: 0.5857 - val_loss: 1.2340 - val_auroc: 0.6114
Epoch 3/10
 - 62s - loss: 1.2197 - auroc: 0.6003 - val_loss: 1.2017 - val_auroc: 0.6154
Epoch 4/10
 - 62s - loss: 1.1893 - auroc: 0.6084 - val_loss: 1.1735 - val_auroc: 0.6174
Epoch 5/10
 - 63s - loss: 1.1628 - auroc: 0.6127 - val_loss: 1.1490 - val_auroc: 0.6195
Epoch 6/10
 - 62s - loss: 1.1384 - auroc: 0.6184 - val_loss: 1.1267 - val_auroc: 0.6210
Epoch 7/10
 - 63s - loss: 1.1171 - auroc: 0.6215 - val_loss: 1.1063 - val_auroc: 0.6218
Epoch 8/10
 - 63s - loss: 1.0979 - auroc: 0.6232 - val_loss: 1.0880 - val_auroc: 0.6230
Epoch 9/10
 - 62s - loss: 1.0797 - auroc: 0.6278 - val_loss: 1.0717 - val_auroc: 0.6234
Epoch 10/10
 - 63s - loss: 1.0633 - auroc: 0.6322 - val_loss: 1.0566 - val_auroc: 0.6228
```

In [0]:

```
score = model_2.evaluate(x=[X8_test,X2_test,X3_test,X4_test, X5_test, X6_test, X7_test], y=y_test,
verbose=2,batch_size=500)
```

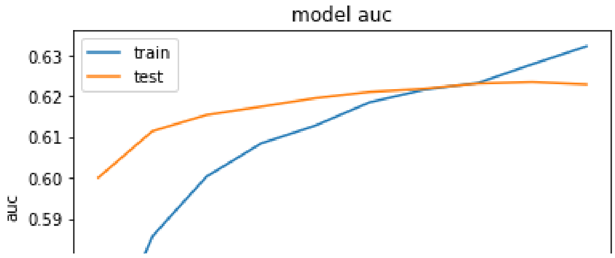
In [162]:

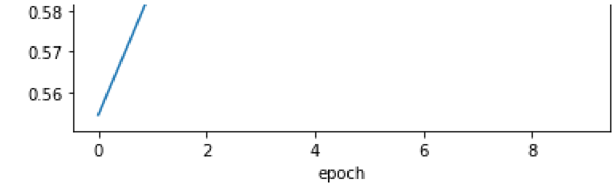
```
print("Test Loss:", score[0])
print("Test AUC:", score[1])
```

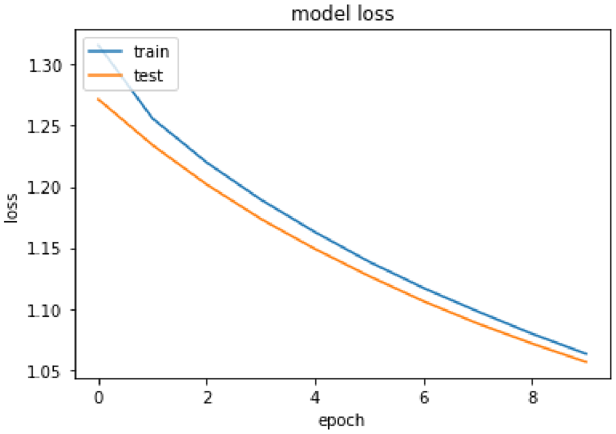
```
Test Loss: 1.0564408168196677
Test AUC: 0.6289043083580907
```

In [163]:

```
plt.plot(history.history['auroc'])
plt.plot(history.history['val_auroc'])

plt.title('model auc')
plt.ylabel('auc')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```







In [164]:

```
# serialize weights to HDF5
model_2.save_weights("model_2.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

Model 3

In [165]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X9_tr = scalar.fit_transform(X_train['price'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
X9_cv = scalar.transform(X_cv['price'].values.reshape(-1,1))
X9_test = scalar.transform(X_test['price'].values.reshape(-1,1))
print(X9_tr.shape)
print(X9_cv.shape)
print(X9_test.shape)
```

```
(64000, 1)
(16000, 1)
(20000, 1)
```

In [166]:

```
scalar = StandardScaler()
X10_tr = scalar.fit_transform(X_train['teacher_number_of_previously_posted_projects'].values.reshap
e(-1, 1))
X10_cv = scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1
))
X10_test = scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-
1, 1))
print(X10_tr.shape)
print(X10_cv.shape)
print(X10_test.shape)
```

```
(64000, 1)
(16000, 1)
(20000, 1)
```

In [167]:

In [167]:

```
numeric_tr = np.hstack((X2_tr,X3_tr,X4_tr,X5_tr,X6_tr,X9_tr,X10_tr))
numeric_cv = np.hstack((X2_cv,X3_cv,X4_cv,X5_cv,X6_cv,X9_cv,X10_cv))
numeric_test = np.hstack((X2_test,X3_test,X4_test,X5_test,X6_test,X9_test,X10_test))
print(numeric_tr.shape)
print(numeric_cv.shape)
print(numeric_test.shape)
```

```
(64000, 101)
(16000, 101)
(20000, 101)
```

In [168]:

```
numeric_tr=np.expand_dims(numeric_tr,axis=2)
numeric_cv=np.expand_dims(numeric_cv,axis=2)
numeric_test=np.expand_dims(numeric_test,axis=2)
print(numeric_tr.shape)
print(numeric_cv.shape)
print(numeric_test.shape)
```

```
(64000, 101, 1)
(16000, 101, 1)
(20000, 101, 1)
```

In [169]:

```
input_9 = Input(shape=(numeric_tr.shape[1],numeric_tr.shape[2],),name='combined_input')
print(input_9.shape)
```

```
(?, 101, 1)
```

In [0]:

```
#At the end we concatenate altogther and add other Dense layers

#output_3=tf.reshape(output_3,[-1,output_3.shape[1],output_3.shape[1]])
#print(output_3.shape)
output_3 = Conv1D(128, 5, strides=1,activation="relu",kernel_initializer='he_uniform',padding='same
')(input_9)
output_3 = MaxPooling1D(pool_size=5)(output_3)
output_3= Dropout(0.2)(output_3)
output_3 = Conv1D(64, 5, activation="relu")(output_3)
output_3 = MaxPooling1D(pool_size=5)(output_3)
output_3= Dropout(0.4)(output_3)
output_3 = Flatten()(output_3)

output_4 = Concatenate()([input_1_flatten,output_3])
output_4 = Dense(256, kernel_initializer="he_uniform",activation='relu')(output_4)
output_4= Dropout(0.2)(output_4)
output_4 = Dense(128, kernel_initializer="he_uniform",activation='relu')(output_4)
output_4= Dropout(0.3)(output_4)
output_4 = Dense(64, kernel_initializer="he_uniform", activation='relu')(output_4)
output_4= Dropout(0.4)(output_4)
output_4 = Dense(2, activation='softmax')(output_4)
```

In [171]:

```
from keras import optimizers
model_3 = Model(inputs=[input_1,input_2,input_3,input_4,input_5,input_6,input_9], outputs=output_4
)
model_3.compile(loss='binary_crossentropy', optimizer=optimizers.SGD(lr=0.01, decay=1e-6, momentum=
0.9) ,metrics=[auroc])
model_3.summary()
```

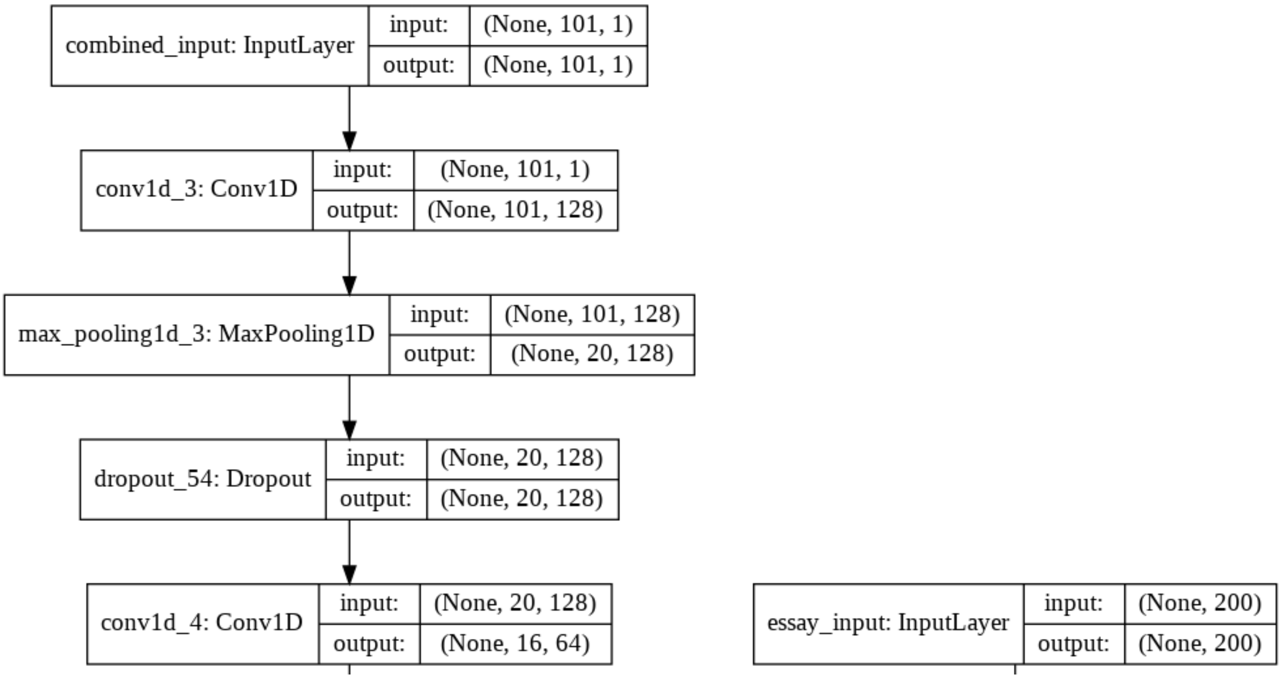
```
Model: "model_23"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
combined_input (InputLayer)     (None, 101, 1)       0
```

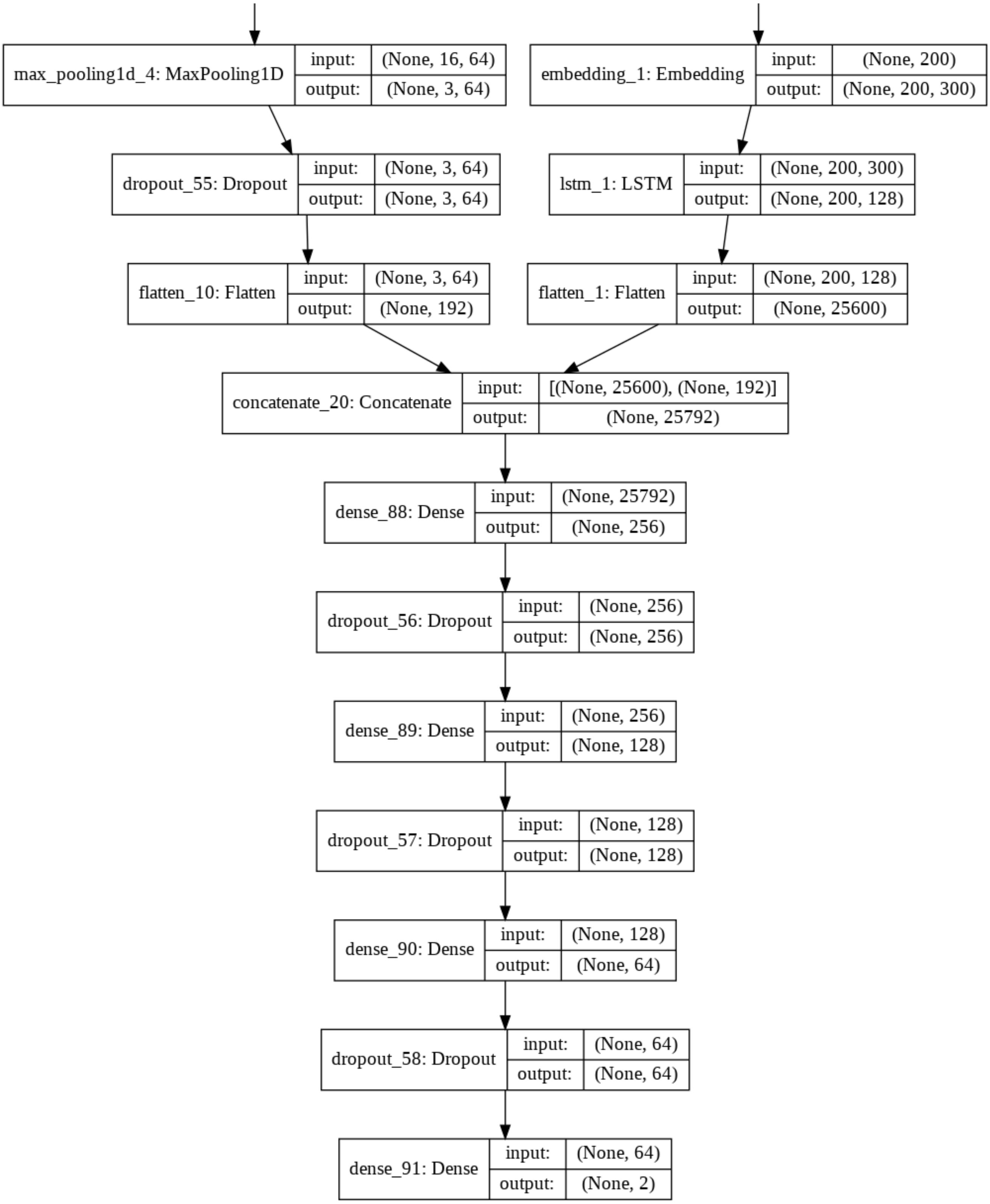
```
__________________________________________________________________________________________________
conv1d_3 (Conv1D)               (None, 101, 128)     768         combined_input[0][0]
__________________________________________________________________________________________________
max_pooling1d_3 (MaxPooling1D)  (None, 20, 128)      0           conv1d_3[0][0]
__________________________________________________________________________________________________
dropout_54 (Dropout)            (None, 20, 128)      0           max_pooling1d_3[0][0]
__________________________________________________________________________________________________
essay_input (InputLayer)        (None, 200)          0
__________________________________________________________________________________________________
conv1d_4 (Conv1D)               (None, 16, 64)       41024       dropout_54[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 200, 300)     13699800    essay_input[0][0]
__________________________________________________________________________________________________
max_pooling1d_4 (MaxPooling1D)  (None, 3, 64)        0           conv1d_4[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 200, 128)     219648      embedding_1[0][0]
__________________________________________________________________________________________________
dropout_55 (Dropout)            (None, 3, 64)        0           max_pooling1d_4[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 25600)        0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_10 (Flatten)            (None, 192)          0           dropout_55[0][0]
__________________________________________________________________________________________________
concatenate_20 (Concatenate)    (None, 25792)        0           flatten_1[0][0]
                                                                 flatten_10[0][0]
__________________________________________________________________________________________________
dense_88 (Dense)                (None, 256)          6603008     concatenate_20[0][0]
__________________________________________________________________________________________________
dropout_56 (Dropout)            (None, 256)          0           dense_88[0][0]
__________________________________________________________________________________________________
dense_89 (Dense)                (None, 128)          32896       dropout_56[0][0]
__________________________________________________________________________________________________
dropout_57 (Dropout)            (None, 128)          0           dense_89[0][0]
__________________________________________________________________________________________________
dense_90 (Dense)                (None, 64)           8256        dropout_57[0][0]
__________________________________________________________________________________________________
dropout_58 (Dropout)            (None, 64)           0           dense_90[0][0]
__________________________________________________________________________________________________
dense_91 (Dense)                (None, 2)            130         dropout_58[0][0]
==================================================================================================
Total params: 20,605,530
Trainable params: 6,905,730
Non-trainable params: 13,699,800
__________________________________________________________________________________________________
```

In [172]:

```
plot_model(model_3, to_file='model_3.png', show_shapes=True, show_layer_names=True)
```

Out[172]:





In [173]:

```
history = model_3.fit(x=[X8_tr,X2_tr,X3_tr,X4_tr, X5_tr, X6_tr,numeric_tr], y=y_train, validation_d
ata=([X8_cv,X2_cv,X3_cv,X4_cv, X5_cv, X6_cv,numeric_cv],y_cv),epochs=10,batch_size=500,verbose=2)
```

```
Train on 64000 samples, validate on 16000 samples
Epoch 1/10
 - 71s - loss: 0.4506 - auroc: 0.5017 - val_loss: 0.4343 - val_auroc: 0.5238
Epoch 2/10
 - 64s - loss: 0.4305 - auroc: 0.5098 - val_loss: 0.4311 - val_auroc: 0.5359
Epoch 3/10
 - 62s - loss: 0.4275 - auroc: 0.5159 - val_loss: 0.4264 - val_auroc: 0.5296
Epoch 4/10
 - 63s - loss: 0.4255 - auroc: 0.5255 - val_loss: 0.4246 - val_auroc: 0.5280
Epoch 5/10
 - 63s - loss: 0.4249 - auroc: 0.5268 - val_loss: 0.4248 - val_auroc: 0.5323
```

```
Epoch 6/10
 - 62s - loss: 0.4242 - auroc: 0.5316 - val_loss: 0.4251 - val_auroc: 0.5379
Epoch 7/10
 - 62s - loss: 0.4232 - auroc: 0.5445 - val_loss: 0.4238 - val_auroc: 0.5442
Epoch 8/10
 - 62s - loss: 0.4230 - auroc: 0.5458 - val_loss: 0.4236 - val_auroc: 0.5453
Epoch 9/10
 - 62s - loss: 0.4229 - auroc: 0.5496 - val_loss: 0.4233 - val_auroc: 0.5477
Epoch 10/10
 - 63s - loss: 0.4227 - auroc: 0.5522 - val_loss: 0.4235 - val_auroc: 0.5384
```

In [0]:

```
score = model_3.evaluate(x=[X8_test,X2_test,X3_test,X4_test, X5_test, X6_test, numeric_test], y=y_test, verbose=2,batch_size=500)
```

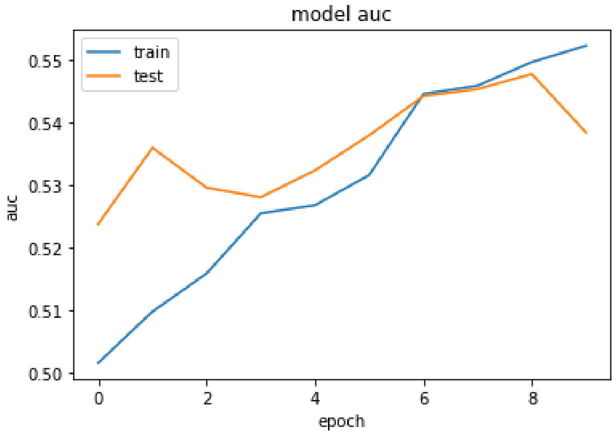
In [175]:

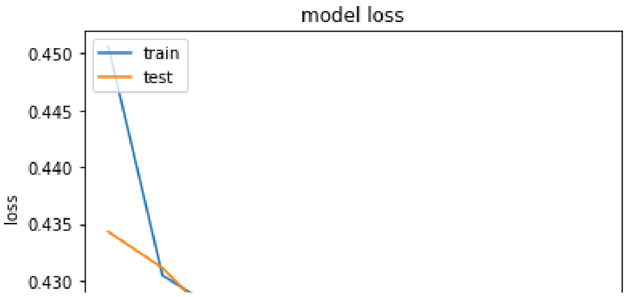
```
print("Test Loss:", score[0])
print("Test AUC:", score[1])
```

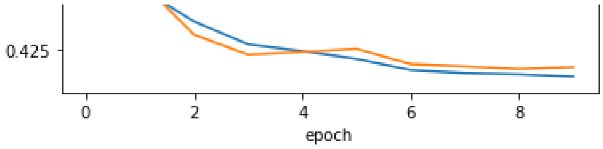
```
Test Loss: 0.4226397812366486
Test AUC: 0.5524177426764195
```

In [176]:

```
plt.plot(history.history['auroc'])
plt.plot(history.history['val_auroc'])

plt.title('model auc')
plt.ylabel('auc')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```







In [177]:

```
# serialize weights to HDF5
model_3.save_weights("model_3.h5")
print("Saved model to disk")
```

```
Saved model to disk
```