Libraries

In [1]:

```python
# Data:
import pandas as pd
import numpy as np
from collections import Counter

# Text preprocessing:
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer, one_hot
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# Layers:
from keras.layers import Input, Embedding, LSTM, Dense, Flatten, concatenate, Dropout,

# Model:
from keras.models import Model

# Metrics:
from sklearn.metrics import roc_auc_score

from time import time
import keras
import matplotlib.pyplot as plt
import seaborn as sns

```

```
Using TensorFlow backend.
```

Load data

In [2]:

```python
data = pd.read_csv('data/final_features.csv')

data.sort_values(by='project_submitted_datetime',inplace=True)

Counter(data['project_is_approved'])
print('ratio excepted: ', round(Counter(data['project_is_approved']).get(1)/data.shape
print('ratio rejected: ', round(Counter(data['project_is_approved']).get(0)/data.shape

```

```
ratio excepted:  85.0 %
ratio rejected:  15.0 %
```

In [3]:

```
for feature in data.iteritems():
    print(feature[0],':','has', str(data[data[feature[0]].isnull().values][feature[0]]

```

```
project_submitted_datetime : has 0 missing values
clean_teacher_prefix : has 0 missing values
clean_school_state : has 0 missing values
clean_grade_categories : has 0 missing values
clean_subject_categories : has 0 missing values
clean_subject_subcategories : has 0 missing values
clean_project_title : has 43 missing values
clean_essay : has 0 missing values
clean_resource_summary : has 0 missing values
resource_summary_contains_numerical_digits : has 0 missing values
std_price : has 0 missing values
std_quantity : has 0 missing values
std_teacher_number_of_previously_posted_projects : has 0 missing values
nrm_price : has 0 missing values
nrm_quantity : has 0 missing values
nrm_teacher_number_of_previously_posted_projects : has 0 missing values
project_is_approved : has 0 missing values
```

In [4]:

```
data.fillna(value={'clean_project_title':''}, inplace=True)

```

In [5]:

```
for feature in data.iteritems():
    print(feature[0],':','has', str(data[data[feature[0]].isnull().values][feature[0]]

```

```
project_submitted_datetime : has 0 missing values
clean_teacher_prefix : has 0 missing values
clean_school_state : has 0 missing values
clean_grade_categories : has 0 missing values
clean_subject_categories : has 0 missing values
clean_subject_subcategories : has 0 missing values
clean_project_title : has 0 missing values
clean_essay : has 0 missing values
clean_resource_summary : has 0 missing values
resource_summary_contains_numerical_digits : has 0 missing values
std_price : has 0 missing values
std_quantity : has 0 missing values
std_teacher_number_of_previously_posted_projects : has 0 missing values
nrm_price : has 0 missing values
nrm_quantity : has 0 missing values
nrm_teacher_number_of_previously_posted_projects : has 0 missing values
project_is_approved : has 0 missing values
```

In [6]:

```python
data['total_text_data'] = data['clean_project_title'] + ' ' + data['clean_essay'] + '

data.drop(labels=['clean_project_title','clean_essay','clean_resource_summary'],axis=1

```

Split data into train,CV and test

In [7]:

```python
# data = data.iloc[0:1000,:]

data_train = data.iloc[0:int(data.shape[0]*0.8),:]
data_train = data_train.iloc[0:int(data_train.shape[0]*0.8),:]
data_cv = data_train.iloc[int(data_train.shape[0]*0.8):,:]
data_test = data.iloc[int(data.shape[0]*0.8):,:]

Y_train = data_train['project_is_approved'].values
data_train.drop(labels=['project_is_approved'],axis=1,inplace=True)
X_train = data_train

Y_cv = data_cv['project_is_approved'].values
data_cv.drop(labels=['project_is_approved'],axis=1,inplace=True)
X_cv = data_cv

Y_test = data_test['project_is_approved'].values
data_test.drop(labels=['project_is_approved'],axis=1,inplace=True)
X_test = data_test

```

```
c:\users\byron\applications\pythonmaster\lib\site-packages\pandas\core\fram
e.py:3697: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  errors=errors)
```

In [8]:

```python
print('X_train shape: ',X_train.shape, 'Y_train shape: ',Y_train.shape)
print('X_cv shape: ',X_cv.shape, 'Y_cv shape: ',Y_cv.shape)
print('X_test shape: ',X_test.shape, 'Y_test shape: ',Y_test.shape)

```

```
X_train shape:  (69918, 14) Y_train shape:  (69918,)
X_cv shape:  (13984, 14) Y_cv shape:  (13984,)
X_test shape:  (21850, 14) Y_test shape:  (21850,)
```

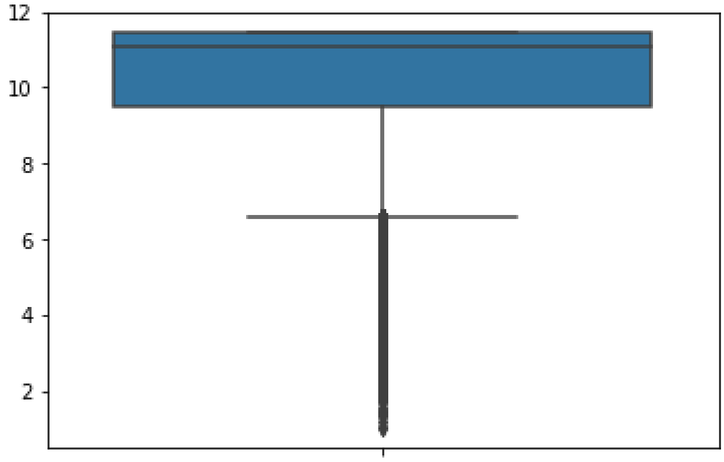
TF-IDF transform data

In [9]:

```
vectorizer = TfidfVectorizer()

total_text_train_tfidf = vectorizer.fit_transform(X_train['total_text_data'])
total_text_cv_tfidf = vectorizer.transform(X_cv['total_text_data'])
total_text_test_tfidf = vectorizer.transform(X_test['total_text_data'])

word_idf_lookup = dict(zip(vectorizer.vocabulary_.keys(),vectorizer.idf_))
idf_word_lookup = { idf : word for word, idf in word_idf_lookup.items() }

```

In [10]:

```
sns.boxplot(y=list(word_idf_lookup.values()))
plt.show()

```



Top words based on IDF value

In [11]:

```
q1 = np.percentile(a=list(word_idf_lookup.values()),q=5)
q3 = np.percentile(a=list(word_idf_lookup.values()),q=95)
word_idf_keep = dict()
for idf, word in idf_word_lookup.items():
    if idf >= q1 and idf <= q3:
        word_idf_keep[word] = idf
word_idf_keep
```

Out[11]:

```
{'tyrerobots': 7.073688343183182,
 'mckissack': 11.4619455276077,
 'positivemany': 10.209182559112332,
 'headstones': 9.757197435369275,
 'committments': 9.957868130831425,
 'belpr': 11.056480419499536,
 'coherence': 10.768798347047754,
 'eread': 9.321879364111428,
 'impracticable': 8.75389532650549,
 'quanjobal': 8.16610866160337,
 'wolverines': 9.670186058379645,
 'cheaptalks': 10.545654795733544,
 'beijing': 7.0552262803434465,
 'roost': 9.516035378552386,
 'aws': 8.077555264261925,
 'islamaphobia': 9.447042507065435,
 'ricebirds': 8.936216883299444,
```

In [12]:

```
def build_reconstructed_sentence(corpus):
    new_corpus = list()
    for sentence in corpus:
        reconstructed_sentence = ''
        for word in sentence.split():
            if word_idf_keep.get(word,0) != 0:
                reconstructed_sentence += word + ' '
        new_corpus.append([reconstructed_sentence.strip()])
    return new_corpus

```

In [13]:

```
X_train['total_text_data'] = np.array(build_reconstructed_sentence(X_train['total_text_
X_cv['total_text_data'] = np.array(build_reconstructed_sentence(X_cv['total_text_data'
X_test['total_text_data'] = np.array(build_reconstructed_sentence(X_test['total_text_da

```

```
c:\users\byron\applications\pythonmaster\lib\site-packages\ipykernel_launche
r.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)

c:\users\byron\applications\pythonmaster\lib\site-packages\ipykernel_launche
r.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  This is separate from the ipykernel package so we can avoid doing imports
 until
```

Turn text data into sequence data - text preprocessing

In [14]:

```
# Use training data:
text = Tokenizer()
text.fit_on_texts(X_train['total_text_data'])
text_sequences_train = text.texts_to_sequences(X_train['total_text_data'])

def max_length(observation_text):
    observation_text_lengths = list()
    for obs in observation_text:
        observation_text_lengths.append(len(obs.split()))
    return np.max(observation_text_lengths)

max_len = int(max_length(X_train['total_text_data'].values))

text_sequences_train = pad_sequences(sequences=text_sequences_train, maxlen=max_len, p

dictionary = text.word_index

frequencies = text.word_counts
frequencies = dict(frequencies)
vocab_size = len(dictionary.keys()) + 1

# Transform cv and test data
text_sequences_cv = text.texts_to_sequences(X_cv['total_text_data'])
text_sequences_cv = pad_sequences(sequences=text_sequences_cv, maxlen=max_len, padding
text_sequences_test = text.texts_to_sequences(X_test['total_text_data'])
text_sequences_test = pad_sequences(sequences=text_sequences_test, maxlen=max_len, pad
```

One hot encode categorical features

In [15]:

```
# Encode teacher prefix:
teacher_vec = CountVectorizer(binary=True)
clean_teacher_prefix_ohe_train = teacher_vec.fit_transform(X_train['clean_teacher_pref
clean_teacher_prefix_ohe_cv = teacher_vec.transform(X_cv['clean_teacher_prefix'])
clean_teacher_prefix_ohe_test = teacher_vec.transform(X_test['clean_teacher_prefix'])

# Encode school state:
school_vec = CountVectorizer(binary=True)
clean_school_state_ohe_train = school_vec.fit_transform(X_train['clean_school_state'])
clean_school_state_ohe_cv = school_vec.transform(X_cv['clean_school_state'])
clean_school_state_ohe_test = school_vec.transform(X_test['clean_school_state'])

# Encode grade categories:
grade_vec = CountVectorizer(binary=True)
clean_grade_categories_ohe_train = grade_vec.fit_transform(X_train['clean_grade_catego
clean_grade_categories_ohe_cv = grade_vec.transform(X_cv['clean_grade_categories'])
clean_grade_categories_ohe_test = grade_vec.transform(X_test['clean_grade_categories']

# Encode subject categories:
cat_vec = CountVectorizer(binary=True)
clean_subject_categories_ohe_train = cat_vec.fit_transform(X_train['clean_subject_cate
clean_subject_categories_ohe_cv = cat_vec.transform(X_cv['clean_subject_categories'])
clean_subject_categories_ohe_test = cat_vec.transform(X_test['clean_subject_categories

# Encode subject subcategories:
subcat_vec = CountVectorizer(binary=True)
clean_subject_subcategories_ohe_train = subcat_vec.fit_transform(X_train['clean_subjec
clean_subject_subcategories_ohe_cv = subcat_vec.transform(X_cv['clean_subject_subcateg
clean_subject_subcategories_ohe_test = subcat_vec.transform(X_test['clean_subject_subc
```

Building the Neural Network

In [16]:

```
input_total_text_data = Input(shape=(max_len,), dtype='int32', name='input_total_text_
embedding_total_text_data = Embedding(input_dim=vocab_size, output_dim=512, embeddings_
lstm = LSTM(units=100, activation='relu', kernel_initializer='he_normal', bias_initial
flatten_1 = Flatten()(lstm)
#############################################################################
input_clean_teacher_prefix = Input(shape=(clean_teacher_prefix_ohe_train.shape[1],), d
embedding_teacher_prefix = Embedding(input_dim=clean_teacher_prefix_ohe_train.shape[1]
flatten_2 = Flatten()(embedding_teacher_prefix)
#############################################################################
input_clean_school_state = Input(shape=(clean_school_state_ohe_train.shape[1],), dtype
embedding_school_state = Embedding(input_dim=clean_school_state_ohe_train.shape[1] + 1
flatten_3 = Flatten()(embedding_school_state)
#############################################################################
input_clean_grade_categories = Input(shape=(clean_grade_categories_ohe_train.shape[1],
embedding_grade_categories = Embedding(input_dim=clean_grade_categories_ohe_train.shap
flatten_4 = Flatten()(embedding_grade_categories)
#############################################################################
input_clean_subject_categories = Input(shape=(clean_subject_categories_ohe_train.shape
embedding_subject_categories = Embedding(input_dim=clean_subject_categories_ohe_train.
flatten_5 = Flatten()(embedding_subject_categories)
#############################################################################
input_clean_subject_subcategories = Input(shape=(clean_subject_subcategories_ohe_train
embedding_subject_subcategories = Embedding(input_dim=clean_subject_subcategories_ohe_
flatten_6 = Flatten()(embedding_subject_subcategories)
#############################################################################
resource_summary_contains_numerical_digits = Input(shape=(1,), dtype='float32', name='
nrm_teacher_number_of_previously_posted_projects = Input(shape=(1,), dtype='float32', 
nrm_price = Input(shape=(1,), dtype='float32', name='nrm_price')
nrm_quantity = Input(shape=(1,), dtype='float32', name='nrm_quantity')
concat = concatenate([resource_summary_contains_numerical_digits,nrm_teacher_number_of_
dense_num = Dense(units=100,activation='relu',kernel_initializer='he_normal',bias_init
#############################################################################
concat_all = concatenate([flatten_1,flatten_2,flatten_3,flatten_4,flatten_5,flatten_6,
#############################################################################
dense_1 = Dense(units=100,activation='relu',kernel_initializer='he_normal',bias_initia
drop_1 = Dropout(rate=0.2)(dense_1)
dense_2 = Dense(units=100,activation='relu',kernel_initializer='he_normal',bias_initia
drop_2 = Dropout(rate=0.2)(dense_2)
dense_3 = Dense(units=100,activation='relu',kernel_initializer='he_normal',bias_initia
output = Dense(units=1,activation='sigmoid',name='output')(dense_3)
```

```
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\tensorflow\python\framework\op_def_library.py:263: colocate_with (fro
m tensorflow.python.framework.ops) is deprecated and will be removed in a fu
ture version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\keras\backend\tensorflow_backend.py:3445: calling dropout (from tenso
rflow.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 [17]:

```
model = Model(inputs=[input_total_text_data,
                      input_clean_teacher_prefix,
                      input_clean_school_state,
                      input_clean_grade_categories,
                      input_clean_subject_categories,
                      input_clean_subject_subcategories,
                      resource_summary_contains_numerical_digits,
                      nrm_teacher_number_of_previously_posted_projects,
                      nrm_price,
                      nrm_quantity
                      ], outputs=[output])
```

In [18]:

```

model.summary()

```

```
____________________________________________________________________________
______________________
Layer (type)                    Output Shape         Param #     Connected t
o
============================================================================
======================
input_total_text_data (InputLay (None, 7)            0
____________________________________________________________________________
______________________
embedding_1 (Embedding)         (None, 7, 512)       184320      input_total
_text_data[0][0]
____________________________________________________________________________
______________________
input_clean_teacher_prefix (Inp (None, 5)            0
____________________________________________________________________________
______________________
input_clean_school_state (Input (None, 51)           0
____________________________________________________________________________
______________________
input_clean_grade_categories (I (None, 4)            0
____________________________________________________________________________
______________________
input_clean_subject_categories  (None, 51)           0
____________________________________________________________________________
______________________
input_clean_subject_subcategori (None, 386)          0
____________________________________________________________________________
______________________
resource_summary_contains_numer (None, 1)            0
____________________________________________________________________________
______________________
nrm_teacher_number_of_previousl (None, 1)            0
____________________________________________________________________________
______________________
nrm_price (InputLayer)          (None, 1)            0
____________________________________________________________________________
______________________
nrm_quantity (InputLayer)       (None, 1)            0
____________________________________________________________________________
______________________
lstm_1 (LSTM)                   (None, 7, 100)       245200      embedding_1
[0][0]
____________________________________________________________________________
______________________
embedding_2 (Embedding)         (None, 5, 512)       3072        input_clean
_teacher_prefix[0][0]
____________________________________________________________________________
______________________
embedding_3 (Embedding)         (None, 51, 512)      26624       input_clean
_school_state[0][0]
____________________________________________________________________________
______________________
```

```
embedding_4 (Embedding)         (None, 4, 512)       2560        input_clean
_grade_categories[0][0
____________________________________________________________________________
______________________
embedding_5 (Embedding)         (None, 51, 512)      26624       input_clean
_subject_categories[0]
____________________________________________________________________________
______________________
embedding_6 (Embedding)         (None, 386, 512)     198144      input_clean
_subject_subcategories
____________________________________________________________________________
______________________
concatenate_1 (Concatenate)     (None, 4)            0           resource_su
mmary_contains_numeric
                                                                 nrm_teacher
_number_of_previously_
                                                                 nrm_price
[0][0]
                                                                 nrm_quantit
y[0][0]
____________________________________________________________________________
______________________
flatten_1 (Flatten)             (None, 700)          0           lstm_1[0]
[0]
____________________________________________________________________________
______________________
flatten_2 (Flatten)             (None, 2560)         0           embedding_2
[0][0]
____________________________________________________________________________
______________________
flatten_3 (Flatten)             (None, 26112)        0           embedding_3
[0][0]
____________________________________________________________________________
______________________
flatten_4 (Flatten)             (None, 2048)         0           embedding_4
[0][0]
____________________________________________________________________________
______________________
flatten_5 (Flatten)             (None, 26112)        0           embedding_5
[0][0]
____________________________________________________________________________
______________________
flatten_6 (Flatten)             (None, 197632)       0           embedding_6
[0][0]
____________________________________________________________________________
______________________
dense_1 (Dense)                 (None, 100)          500         concatenate
_1[0][0]
____________________________________________________________________________
______________________
concatenate_2 (Concatenate)     (None, 255264)       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_2 (Dense)                 (None, 100)          25526500    concatenate
_2[0][0]
____________________________________________________________________________
______________________
dropout_1 (Dropout)             (None, 100)          0           dense_2[0]
[0]
____________________________________________________________________________
______________________
dense_3 (Dense)                 (None, 100)          10100       dropout_1
[0][0]
____________________________________________________________________________
______________________
dropout_2 (Dropout)             (None, 100)          0           dense_3[0]
[0]
____________________________________________________________________________
______________________
dense_4 (Dense)                 (None, 100)          10100       dropout_2
[0][0]
____________________________________________________________________________
______________________
output (Dense)                  (None, 1)            101         dense_4[0]
[0]
============================================================================
======================
Total params: 26,233,845
Trainable params: 26,233,845
Non-trainable params: 0
____________________________________________________________________________
______________________
```

In [19]:

```
tensorboard = keras.callbacks.TensorBoard(log_dir='.logs/{}'.format(time()), histogram_

```

In [20]:

```
model.compile(optimizer=keras.optimizers.Adam(lr=0.0001), loss='binary_crossentropy', 

```

In [21]:

```
class roc_callback(keras.callbacks.Callback):
    def __init__(self,training_data,validation_data):
        self.x = training_data[0]
        self.y = training_data[1]
        self.x_val = validation_data[0]
        self.y_val = validation_data[1]

    def on_train_begin(self, logs={}):
        return

    def on_train_end(self, logs={}):
        return

    def on_epoch_begin(self, epoch, logs={}):
        return

    def on_epoch_end(self, epoch, logs={}):
        y_pred = self.model.predict(self.x)
        roc = roc_auc_score(self.y, y_pred)
        y_pred_val = self.model.predict(self.x_val)
        roc_val = roc_auc_score(self.y_val, y_pred_val)
        print('\rroc-auc: %s - roc-auc_val: %s' % (str(round(roc,4)),str(round(roc_val
        return

    def on_batch_begin(self, batch, logs={}):
        return

    def on_batch_end(self, batch, logs={}):
        return

```

In [22]:

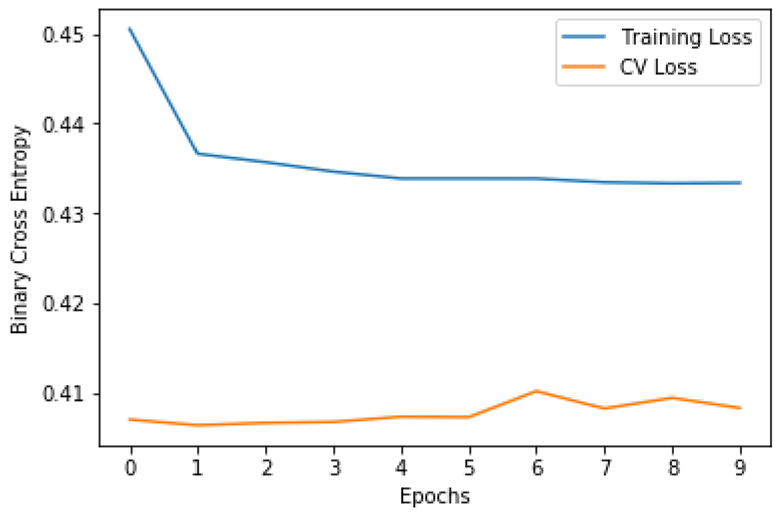
```
batch_size = 100
epochs = 10
history = model.fit(x=[text_sequences_train,
                        clean_teacher_prefix_ohe_train,
                        clean_school_state_ohe_train,
                        clean_grade_categories_ohe_train,
                        clean_subject_categories_ohe_train,
                        clean_subject_subcategories_ohe_train,
                        X_train['resource_summary_contains_numerical_digits'],
                        X_train['nrm_teacher_number_of_previously_posted_projects'],
                        X_train['nrm_price'],
                        X_train['nrm_quantity']
                        ],y=[Y_train],
                        validation_data=([text_sequences_cv,
                        clean_teacher_prefix_ohe_cv,
                        clean_school_state_ohe_cv,
                        clean_grade_categories_ohe_cv,
                        clean_subject_categories_ohe_cv,
                        clean_subject_subcategories_ohe_cv,
                        X_cv['resource_summary_contains_numerical_digits'],
                        X_cv['nrm_teacher_number_of_previously_posted_projects'],
                        X_cv['nrm_price'],
                        X_cv['nrm_quantity']],[Y_cv]),
                        batch_size=batch_size,
                        epochs=epochs,
                        callbacks=[tensorboard,
                        roc_callback(training_data=([text_sequences_train,
                        clean_teacher_prefix_ohe_train,
                        clean_school_state_ohe_train,
                        clean_grade_categories_ohe_train,
                        clean_subject_categories_ohe_train,
                        clean_subject_subcategories_ohe_train,
                        X_train['resource_summary_contains_numerical_digits'],
                        X_train['nrm_teacher_number_of_previously_posted_projects'],
                        X_train['nrm_price'],
                        X_train['nrm_quantity']
                        ], Y_train),
                        validation_data=([text_sequences_cv,
                        clean_teacher_prefix_ohe_cv,
                        clean_school_state_ohe_cv,
                        clean_grade_categories_ohe_cv,
                        clean_subject_categories_ohe_cv,
                        clean_subject_subcategories_ohe_cv,
                        X_cv['resource_summary_contains_numerical_digits'],
                        X_cv['nrm_teacher_number_of_previously_posted_projects'],
                        X_cv['nrm_price'],
                        X_cv['nrm_quantity']], Y_cv))]
                        )
```

```
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.pyt
hon.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 69918 samples, validate on 13984 samples
Epoch 1/10
```

```
69918/69918 [==============================] - 437s 6ms/step - loss: 0.4506
- acc: 0.8402 - val_loss: 0.4070 - val_acc: 0.8593
roc-auc: 0.4492 - roc-auc_val: 0.4448
Epoch 2/10
69918/69918 [==============================] - 436s 6ms/step - loss: 0.4366
- acc: 0.8441 - val_loss: 0.4063 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 3/10
69918/69918 [==============================] - 437s 6ms/step - loss: 0.4357
- acc: 0.8441 - val_loss: 0.4066 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 4/10
69918/69918 [==============================] - 442s 6ms/step - loss: 0.4346
- acc: 0.8441 - val_loss: 0.4067 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 5/10
69918/69918 [==============================] - 442s 6ms/step - loss: 0.4339
- acc: 0.8441 - val_loss: 0.4073 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 6/10
69918/69918 [==============================] - 441s 6ms/step - loss: 0.4339
- acc: 0.8441 - val_loss: 0.4072 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 7/10
69918/69918 [==============================] - 442s 6ms/step - loss: 0.4339
- acc: 0.8441 - val_loss: 0.4101 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 8/10
69918/69918 [==============================] - 440s 6ms/step - loss: 0.4334
- acc: 0.8441 - val_loss: 0.4082 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 9/10
69918/69918 [==============================] - 438s 6ms/step - loss: 0.4333
- acc: 0.8441 - val_loss: 0.4094 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
Epoch 10/10
69918/69918 [==============================] - 439s 6ms/step - loss: 0.4334
- acc: 0.8441 - val_loss: 0.4083 - val_acc: 0.8593
roc-auc: 0.5 - roc-auc_val: 0.5
```

In [23]:

```
plt.plot(np.arange(epochs),history.history['loss'],label='Training Loss')
plt.plot(np.arange(epochs),history.history['val_loss'],label='CV Loss')
plt.xlabel('Epochs')
plt.ylabel('Binary Cross Entropy')
plt.legend()
plt.xticks(np.arange(epochs))
plt.show()
```



In [24]:

```
model.evaluate(x=[text_sequences_test,
                        clean_teacher_prefix_ohe_test,
                        clean_school_state_ohe_test,
                        clean_grade_categories_ohe_test,
                        clean_subject_categories_ohe_test,
                        clean_subject_subcategories_ohe_test,
                        X_test['resource_summary_contains_numerical_digits'],
                        X_test['nrm_teacher_number_of_previously_posted_projects'],
                        X_test['nrm_price'],
                        X_test['nrm_quantity']
                        ],y=[Y_test],
                        batch_size=batch_size
                        )
```

```
21850/21850 [==============================] - 29s 1ms/step
```

Out[24]:

```
[0.4157320438997151, 0.8546910746965037]
```

In [25]:

```
Y_pred = model.predict(x=[text_sequences_test,
                        clean_teacher_prefix_ohe_test,
                        clean_school_state_ohe_test,
                        clean_grade_categories_ohe_test,
                        clean_subject_categories_ohe_test,
                        clean_subject_subcategories_ohe_test,
                        X_test['resource_summary_contains_numerical_digits'],
                        X_test['nrm_teacher_number_of_previously_posted_projects'],
                        X_test['nrm_price'],
                        X_test['nrm_quantity']
                        ],
                      batch_size=batch_size)
roc_auc_score(Y_test, Y_pred)
```

Out[25]:

```
0.5
```