Libraries

In [1]:

```
# 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

# 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
from sklearn.utils import class_weight

```

```
Using TensorFlow backend.
```

Load data

In [2]:

```
data = pd.read_csv('data/final_features.csv')
data.sort_values(by='project_submitted_datetime',inplace=True)
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
data.shape
```

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

Out[2]:

```
(109248, 17)
```

In [3]:

```
# diff = int(round(Counter(data['project_is_approved']).get(1),2) - round(Counter(data
# oversampled = np.random.choice(a=data[data['project_is_approved'] == 0].index.values,
# data = pd.concat(objs=[data,data.loc[oversampled,:]],axis=0)
```

In [4]:

```
# data.sort_values(by='project_submitted_datetime',inplace=True)
# print('ratio excepted: ', round(Counter(data['project_is_approved']).get(1)/data.shap
# print('ratio rejected: ', round(Counter(data['project_is_approved']).get(0)/data.shap
# data.shape
```

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 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 [6]:

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

In [7]:

```
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 [8]:

```
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 [9]:

```
# data = data.iloc[0:10000,:]

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

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

Y_test = data_test['project_is_approved']
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 [10]:

```
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,)
```

Turn text data into sequence data - text preprocessing

In [11]:

```
# 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.mean(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, pa

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

```

Get pretrained word embeddings

In [12]:

```
embeddings_index = dict()
with open(r'glove/glove.6B.100d.txt','r',encoding='utf-8') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.array(values[1:], dtype='float32')
        embeddings_index[word] = coefs
print('Loaded {} word vectors.'.format(len(embeddings_index)))

embedding_matrix = np.zeros((vocab_size, 100))
for word, i in dictionary.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
Loaded 400000 word vectors.
```

One hot encode categorical features

In [13]:

```
# Encode teacher prefix:
teacher_vec = CountVectorizer(binary=True)
clean_teacher_prefix_ohe_train = teacher_vec.fit_transform(X_train['clean_teacher_prefi
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_categor
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_categ
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_subject
clean_subject_subcategories_ohe_cv = subcat_vec.transform(X_cv['clean_subject_subcatego
clean_subject_subcategories_ohe_test = subcat_vec.transform(X_test['clean_subject_subca
```

Building the Neural Network

In [14]:

```
ext_data = Input(shape=(max_len,), dtype='int32', name='input_total_text_data')
initializer='uniform'
al_text_data = Embedding(input_dim=vocab_size, output_dim=100, weights=[embedding_matrix], t
nits=100, activation='tanh', kernel_initializer='glorot_normal', bias_initializer='glorot_no
latten()(lstm)
###################################################################
eacher_prefix = Input(shape=(clean_teacher_prefix_ohe_train.shape[1],), dtype='int32', name=
cher_prefix = Embedding(input_dim=clean_teacher_prefix_ohe_train.shape[1] + 1, output_dim=51
latten()(embedding_teacher_prefix)
###################################################################
chool_state = Input(shape=(clean_school_state_ohe_train.shape[1],), dtype='int32', name='inp
ool_state = Embedding(input_dim=clean_school_state_ohe_train.shape[1] + 1, output_dim=512, e
latten()(embedding_school_state)
###################################################################
rade_categories = Input(shape=(clean_grade_categories_ohe_train.shape[1],), dtype='int32', n
de_categories = Embedding(input_dim=clean_grade_categories_ohe_train.shape[1] + 1, output_di
latten()(embedding_grade_categories)
###################################################################
ubject_categories = Input(shape=(clean_subject_categories_ohe_train.shape[1],), dtype='int32
ject_categories = Embedding(input_dim=clean_subject_categories_ohe_train.shape[1] + 1, outpu
latten()(embedding_subject_categories)
###################################################################
ubject_subcategories = Input(shape=(clean_subject_subcategories_ohe_train.shape[1],), dtype=
ject_subcategories = Embedding(input_dim=clean_subject_subcategories_ohe_train.shape[1] + 1,
latten()(embedding_subject_subcategories)
###################################################################
ary_contains_numerical_digits = Input(shape=(1,), dtype='float32', name='resource_summary_co
umber_of_previously_posted_projects = Input(shape=(1,), dtype='float32', name='nrm_teacher_n
nput(shape=(1,), dtype='float32', name='nrm_price')
= Input(shape=(1,), dtype='float32', name='nrm_quantity')
atenate([resource_summary_contains_numerical_digits,nrm_teacher_number_of_previously_posted_
ense(units=100,activation='relu',kernel_initializer='he_normal')(concat)
###################################################################
concatenate([flatten_1,flatten_2,flatten_3,flatten_4,flatten_5,flatten_6,dense_num])
###################################################################
se(units=100,activation='relu',kernel_initializer='he_normal',bias_initializer='he_normal')(
out(rate=0.2)(dense_1)
se(units=100,activation='relu',kernel_initializer='he_normal',bias_initializer='he_normal')(
out(rate=0.2)(dense_2)
se(units=100,activation='relu',kernel_initializer='he_normal',bias_initializer='he_normal')(
e(units=1,activation='sigmoid',kernel_initializer='glorot_normal',bias_initializer='glorot_n
```

```
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 [15]:

```
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 [16]:

```
1 model.summary()
```

```
__________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected t
o
==================================================================================
==================
input_total_text_data (InputLay (None, 154)          0
__________________________________________________________________________________
__________________
embedding_1 (Embedding)         (None, 154, 100)     5090200     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, 154, 100)     80400       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, 15400)        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, 269964)       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)          26996500    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: 32,444,925
Trainable params: 27,354,725
Non-trainable params: 5,090,200
__________________________________________________________________________________
_____________________
```

In [17]:

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

In [18]:

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

In [19]:

```
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 [20]:

```
class_weights = class_weight.compute_class_weight('balanced',
                                                  np.unique(Y_train),
                                                  Y_train)
```

In [21]:

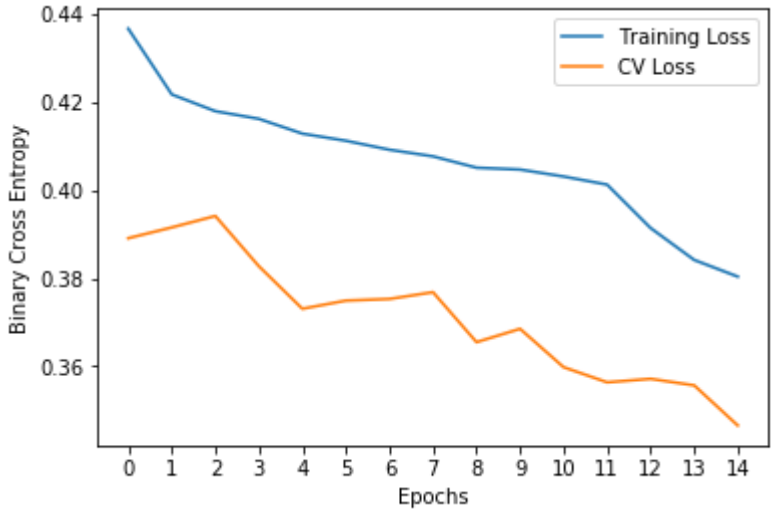
```
batch_size = 100
epochs = 15
history = model.fit(x=[text_sequences_train,
                       clean_teacher_prefix_ohe_train,
                       clean_school_state_ohe_train,
                       clean_grade_categories_ohe_train,
                       clean_subject_categories_ohe_train,
                       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))],
                       class_weight=class_weights
                       )
```

```
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 1/15
69918/69918 [==============================] - 516s 7ms/step - loss: 0.4367
- acc: 0.8426 - val_loss: 0.3891 - val_acc: 0.8593
roc-auc: 0.6699 - roc-auc_val: 0.6673
Epoch 2/15
69918/69918 [==============================] - 503s 7ms/step - loss: 0.4218
- acc: 0.8441 - val_loss: 0.3915 - val_acc: 0.8593
roc-auc: 0.6635 - roc-auc_val: 0.6627
Epoch 3/15
69918/69918 [==============================] - 501s 7ms/step - loss: 0.4180
- acc: 0.8441 - val_loss: 0.3941 - val_acc: 0.8593
roc-auc: 0.6766 - roc-auc_val: 0.6758
Epoch 4/15
69918/69918 [==============================] - 508s 7ms/step - loss: 0.4162
- acc: 0.8441 - val_loss: 0.3827 - val_acc: 0.8593
roc-auc: 0.6974 - roc-auc_val: 0.6963
Epoch 5/15
69918/69918 [==============================] - 509s 7ms/step - loss: 0.4129
- acc: 0.8441 - val_loss: 0.3731 - val_acc: 0.8593
roc-auc: 0.7153 - roc-auc_val: 0.7173
Epoch 6/15
69918/69918 [==============================] - 499s 7ms/step - loss: 0.4112
- acc: 0.8441 - val_loss: 0.3749 - val_acc: 0.8593
roc-auc: 0.7213 - roc-auc_val: 0.7235
Epoch 7/15
69918/69918 [==============================] - 499s 7ms/step - loss: 0.4092
- acc: 0.8441 - val_loss: 0.3753 - val_acc: 0.8593
roc-auc: 0.7292 - roc-auc_val: 0.7312
Epoch 8/15
69918/69918 [==============================] - 501s 7ms/step - loss: 0.4077
- acc: 0.8441 - val_loss: 0.3768 - val_acc: 0.8593
roc-auc: 0.7302 - roc-auc_val: 0.7331
Epoch 9/15
69918/69918 [==============================] - 499s 7ms/step - loss: 0.4051
- acc: 0.8441 - val_loss: 0.3655 - val_acc: 0.8593
roc-auc: 0.7549 - roc-auc_val: 0.76
Epoch 10/15
69918/69918 [==============================] - 499s 7ms/step - loss: 0.4047
- acc: 0.8441 - val_loss: 0.3685 - val_acc: 0.8593
roc-auc: 0.7541 - roc-auc_val: 0.7574
Epoch 11/15
69918/69918 [==============================] - 500s 7ms/step - loss: 0.4031
- acc: 0.8441 - val_loss: 0.3598 - val_acc: 0.8593
roc-auc: 0.7632 - roc-auc_val: 0.7673
Epoch 12/15
69918/69918 [==============================] - 502s 7ms/step - loss: 0.4013
- acc: 0.8441 - val_loss: 0.3563 - val_acc: 0.8593
roc-auc: 0.7697 - roc-auc_val: 0.7734
Epoch 13/15
69918/69918 [==============================] - 515s 7ms/step - loss: 0.3914
- acc: 0.8441 - val_loss: 0.3571 - val_acc: 0.8593
roc-auc: 0.7683 - roc-auc_val: 0.7743
Epoch 14/15
69918/69918 [==============================] - 500s 7ms/step - loss: 0.3842
- acc: 0.8441 - val_loss: 0.3556 - val_acc: 0.8593
roc-auc: 0.7761 - roc-auc_val: 0.7799
Epoch 15/15
69918/69918 [==============================] - 500s 7ms/step - loss: 0.3804
- acc: 0.8441 - val_loss: 0.3465 - val_acc: 0.8593
roc-auc: 0.7876 - roc-auc_val: 0.7904
```

In [22]:

```
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 [23]:

```
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 [==============================] - 37s 2ms/step
```

Out[23]:

```
[0.3726955457532979, 0.8546910746965037]
```

AUC Score on test data

In [24]:

```
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[24]:

```
0.7293475929966586
```