In [0]:

```
# Credits: https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/
# LSTM for sequence classification in the IMDB dataset
import numpy
#from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
import pandas as pd
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
#import pandas as pd
#import numpy as np
import nltk
import string
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.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

from collections import Counter
```

In [0]:

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

In [0]:

```
project_data = pd.read_csv('drive/My Drive/train_data.csv',error_bad_lines=False,engine='python')
resource_data = pd.read_csv('drive/My Drive/resources.csv')
```

In [0]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [0]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[0]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [0]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

In [0]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
```

```
sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

In [0]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [0]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [0]:

```
from sklearn.model_selection import train_test_split
data=project_data
data.head()
y=data.project_is_approved
x=data
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,test_size=0.2,stratify=y)
x_train,x_cv,y_train,y_cv=train_test_split(x_train,y_train,train_size=0.8,test_size=0.2,stratify=y
_train)


print("shape of train data ")
print(x_train.shape)
print(y_train.shape)
print("shape of test data ")
print(x_test.shape)
print(y_test.shape)
print("shape of crossvalidation data ")
print(x_cv.shape)
print(y_cv.shape)
```

```
shape of train data
(69918, 20)
(69918,)
shape of test data
(21850, 20)
(21850,)
shape of crossvalidation data
(17480, 20)
(17480,)
```

In [0]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import Normalizer
from scipy.sparse import hstack
import numpy as np
def cat(x):
  vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
  vectorizer.fit(x['clean_categories'].values)
  print(vectorizer.get_feature_names())


  categories_one_hot = vectorizer.transform(x['clean_categories'].values)
  print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
  return categories_one_hot


def  subcat(x):
  vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
binary=True)
  vectorizer.fit(x['clean_subcategories'].values)
  print(vectorizer.get_feature_names())



  subcategories_one_hot = vectorizer.transform(x['clean_subcategories'].values)
  print("Shape of matrix after one hot encodig ",subcategories_one_hot.shape)
  return subcategories_one_hot

def state(x):
  vectorizer = CountVectorizer(lowercase=False, binary=True)
  vectorizer.fit(x['school_state'].values)
  print(vectorizer.get_feature_names())



  state_one_hot = vectorizer.transform(x['school_state'].values)
  print("Shape of matrix after one hot encodig ",state_one_hot.shape)
  return state_one_hot
def prefix(x):
  x = x.replace(np.nan, '', regex=True)
  vectorizer = CountVectorizer( vocabulary=['Mrs.', 'Ms.','Mr.','Teacher','Dr.'],lowercase=False)
  vectorizer.fit(x['teacher_prefix'].values)
  print(vectorizer.get_feature_names())



  prefix_one_hot = vectorizer.transform(x['teacher_prefix'].values)
  print("Shape of matrix after one hot encodig ",prefix_one_hot.shape)
  return prefix_one_hot

def grade(x):

  vocab=[]
  for i in x['project_grade_category'].values:
      vocab.append(i)
  v_set=set(vocab)
  vocab=list(v_set)
  vectorizer = CountVectorizer(vocabulary=vocab,lowercase=False)
  vectorizer.fit(x['project_grade_category'].values)




  grade_one_hot = vectorizer.transform(x['project_grade_category'].values)
  print("Shape of matrix after one hot encodig ",grade_one_hot.shape)
  return grade_one_hot

def price(x):
  price_scalar =  Normalizer(copy=False,norm='l2')
  price_scalar.fit(x['price'].values.reshape(1,-1)) # finding the mean and standard deviation of th
is data
```

```
# Now standardize the data with above maen and variance.

  price_standardized = price_scalar.transform(x['price'].values.reshape(1, -1))
  price_standardized=np.transpose(price_standardized)
  return price_standardized

def proj(x):
  projects_scalar = Normalizer(copy=False,norm='l2')
  projects_scalar.fit(x['teacher_number_of_previously_posted_projects'].values.reshape(1,-1)) # fin
ding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
  projects_standardized =
projects_scalar.transform(x['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))
  projects_standardized =np.transpose(projects_standardized)
  return projects_standardized
def qty(x):

  qty_scalar= Normalizer(copy=False,norm='l2')
  qty_scalar.fit(x['quantity'].values.reshape(1,-1)) # finding the mean and standard deviation of t
his data


    # Now standardize the data with above maen and variance.
  qty_standardized = qty_scalar.transform(x['quantity'].values.reshape(1, -1))
  qty_standardized=np.transpose(qty_standardized)
  return qty_standardized

    #X1 = hstack((categories_one_hot,
subcategories_one_hot,state_one_hot,prefix_one_hot,grade_one_hot,
price_standardized,projects_standardized,qty_standardized))
    #print(X1.shape)
    #return(X1)
```

In [0]:

```
train_cat=cat(x_train)
train_subcat=subcat(x_train)
train_state=state(x_train)
train_prefix=prefix(x_train)
train_grade=grade(x_train)
train_price=price(x_train)
train_proj=proj(x_train)
train_qty=qty(x_train)
test_cat=cat(x_test)
test_subcat=subcat(x_test)
test_state=state(x_test)
test_prefix=prefix(x_test)
test_grade=grade(x_test)
test_price=price(x_test)
test_proj=proj(x_test)
test_qty=qty(x_test)
cv_cat=cat(x_cv)
cv_subcat=subcat(x_cv)
cv_state=state(x_cv)
cv_prefix=prefix(x_cv)
cv_grade=grade(x_cv)
cv_price=price(x_cv)
cv_proj=proj(x_cv)
cv_qty=qty(x_cv)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (69918, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (69918, 30)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
```

```
Shape of matrix after one hot encodig  (69918, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (69918, 5)
Shape of matrix after one hot encodig  (69918, 4)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (21850, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (21850, 30)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encodig  (21850, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (21850, 5)
Shape of matrix after one hot encodig  (21850, 4)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (17480, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (17480, 30)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encodig  (17480, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (17480, 5)
Shape of matrix after one hot encodig  (17480, 4)
```

In [0]:

```
train_cat.shape
```

Out[0]:

```
(69918, 9)
```

In [0]:

```
train_other_text= hstack((train_cat,train_subcat,train_state,train_prefix,train_grade,train_price,
train_proj,train_qty))
test_other_text=hstack((test_cat,test_subcat,test_state,test_prefix,test_grade,test_price,test_proj
,test_qty))
cv_other_text=hstack((cv_cat,cv_subcat,cv_state,cv_prefix,cv_grade,cv_price,cv_proj,cv_qty))
```

In [0]:

```
from sklearn.preprocessing import normalize
train_other_text_norm = normalize(train_other_text, norm='l1', axis=1)
test_other_text_norm = normalize(test_other_text, norm='l1', axis=1)
cv_other_text_norm = normalize(cv_other_text, norm='l1', axis=1)
```

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
```

```
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [0]:

```
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:

```
def preprocessing(x):
    import nltk
    nltk.download('stopwords')
    from tqdm import tqdm
    preprocessed_essays = []
    # tqdm is for printing the status bar
    for sentence in tqdm(x.values):
        sent = decontracted(sentence)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\"', ' ')
        sent = sent.replace('\\n', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent=' '.join(e.lower() for e in sent.split() if e.lower() not in stopwords)
        preprocessed_essays.append(sent.strip())
    return preprocessed_essays
```

In [0]:

```
train_essay=[]

test_essay=[]
cv_essay=[]



train_essay=preprocessing(x_train['essay'])
test_essay=preprocessing(x_test['essay'])
cv_essay=preprocessing(x_cv['essay'])
```

```
  0%|          | 174/69918 [00:00<00:40, 1739.85it/s]
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
100%|██████████| 69918/69918 [00:39<00:00, 1783.27it/s]
  1%|          | 170/21850 [00:00<00:12, 1696.49it/s]
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
100%|██████████| 21850/21850 [00:12<00:00, 1779.80it/s]
  1%|          | 177/17480 [00:00<00:09, 1763.43it/s]
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
100%|██████████| 17480/17480 [00:09<00:00, 1771.97it/s]
```

In [0]:

```
embeddings_index = dict()

f = open('drive/My Drive/glove_words/glove.42B.300d.txt')
for line in f:
 values = line.split()
 word = values[0]
 coefs = np.asarray(values[1:], dtype='float32')
 embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
# create a weight matrix for words in training docs
```

```
Loaded 1229093 word vectors.
```

In [0]:

```
from keras.preprocessing.text import Tokenizer
t=Tokenizer()
t.fit_on_texts(train_essay)
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_train = t.texts_to_sequences(train_essay)
print(encoded_train)
encoded_test=t.texts_to_sequences(test_essay)
encoded_cv=t.texts_to_sequences(cv_essay)
# pad documents to a max length of 4 words
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

In [0]:

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

In [0]:

```
from keras_preprocessing.sequence import pad_sequences
max_length=300
train_padded=pad_sequences(encoded_train,maxlen=max_length,padding='post')
test_padded=pad_sequences(encoded_test,maxlen=max_length,padding='post')
cv_padded=pad_sequences(encoded_cv,maxlen=max_length,padding='post')
print(train_padded.shape)
print(test_padded.shape)
print(cv_padded.shape)
```

```
(69918, 300)
(21850, 300)
(17480, 300)
```

In [0]:

```
from keras.layers import Input,Embedding,Flatten,LSTM,Dense,concatenate,Dropout,BatchNormalization
,Conv1D,MaxPooling1D
essay_input = Input(shape=(len(train_padded[0]),), name='essay_input')
otherthan_text_input=Input(shape=(train_other_text_norm.shape[1],),name='otherthan_text_input')
```

In [0]:

```
from tensorflow.keras import regularizers
essay1=Embedding(input_dim=vocab_size,output_dim=300,input_length=train_padded.shape[1],weights=[e
mbedding_matrix],trainable=False)(essay_input)
essay1 =
LSTM(32,return_sequences=True,kernel_initializer='glorot_normal',kernel_regularizer=regularizers.l
2(0.001))(essay1)
essay1= Flatten()(essay1)
otherthan_text=Embedding(input_dim=train_other_text_norm.shape[1],output_dim=32,input_length=train_
other_text_norm.shape[1])(otherthan_text_input)
otherthan_text=Conv1D(64,2,activation='relu',padding='same')(otherthan_text)
otherthan_text=MaxPooling1D(2,padding='same')(otherthan_text)
otherthan_text=Flatten()(otherthan_text)
```

In [0]:

```
final_x= concatenate([essay1,otherthan_text])
```

In [0]:

```
from keras import utils
num_classes=2
y_train1 = utils.to_categorical(y_train, num_classes)
y_test1 = utils.to_categorical(y_test, num_classes)
y_cv1 = utils.to_categorical(y_cv, num_classes)
```

In [0]:

```
from sklearn.utils import compute_class_weight
class_label = project_data['project_is_approved']
class_wght = compute_class_weight("balanced", classes= np.unique(class_label),y=class_label)
```

In [0]:

```
import tensorflow as tf
def auc(y_true,y_pred):
```

```
  return tf.compat.v1.py_func(roc_auc_score,(y_true,y_pred),tf.double)
```

In [0]:

```
l1=Dense(24,activation='relu')(final_x)
l1=BatchNormalization()(l1)
l1=Dropout(0.5)(l1)
l2=Dense(16,activation='relu')(l1)
l2=BatchNormalization()(l2)
l2=Dropout(0.5)(l2)
l3=Dense(8,activation='relu')(l2)
output=Dense(2,activation='softmax',name='output')(l3)
```

In [0]:

```
try:

  %tensorflow_version 2.x
except Exception:
  pass

# Load the TensorBoard notebook extension.
%load_ext tensorboard
```

In [0]:

```
from keras import callbacks
!rm -rf logs/image

logdir = "logs1/image/"
# Define the basic TensorBoard callback.
tensorboard_callback = callbacks.TensorBoard(log_dir=logdir)
file_writer_cm = tf.summary.create_file_writer(logdir + '/cm')
```

In [0]:

```
%tensorboard --logdir logs1/image
import tensorflow as tf
from keras.models import Model
from sklearn.metrics import roc_auc_score
from keras.callbacks import TensorBoard
from time import time
from sklearn.utils import class_weight
from keras.optimizers import Adam
#from tf.keras import metrics
tensor=TensorBoard(log_dir='logs/{}'.format(time()),write_graph=True)
model=Model(inputs=[essay_input,otherthan_text_input],outputs=output)
model.compile(optimizer=Adam(lr=0.0005),loss='categorical_crossentropy',metrics=[auc])
weights = class_weight.compute_class_weight('balanced',np.unique(y_train),y_train)
print(model.summary())
history=model.fit([train_padded,train_other_text_norm.todense()],y_train1, batch_size=3200, epochs
=20, verbose=1, shuffle='batch' , validation_data=([cv_padded,cv_other_text_norm.todense()], y_cv1
),class_weight=class_wght,callbacks=[tensorboard_callback])
```

```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
otherthan_text_input (InputLaye (None, 102)          0
__________________________________________________________________________________________________
essay_input (InputLayer)        (None, 300)          0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 102, 32)      3264        otherthan_text_input[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 300, 300)     14197800    essay_input[0][0]
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 102, 64)      4160        embedding_2[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 300, 32)      42624       embedding_1[0][0]
__________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)  (None, 51, 64)       0           conv1d_1[0][0]
__________________________________________________________________________________________________
```

```
flatten_1 (Flatten)             (None, 9600)         0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 3264)         0           max_pooling1d_1[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 12864)        0           flatten_1[0][0]
                                                                 flatten_2[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 24)           308760      concatenate_1[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 24)           96          dense_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 24)           0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 16)           400         dropout_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 16)           64          dense_2[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 16)           0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 8)            136         dropout_2[0][0]
__________________________________________________________________________________________________
output (Dense)                  (None, 2)            18          dense_3[0][0]
==================================================================================================
Total params: 14,557,322
Trainable params: 359,442
Non-trainable params: 14,197,880
__________________________________________________________________________________________________
None
Train on 69918 samples, validate on 17480 samples
Epoch 1/20
69918/69918 [==============================] - 123s 2ms/step - loss: 0.5435 - auc: 0.6310 - val_lo
ss: 0.6957 - val_auc: 0.5421
Epoch 2/20
69918/69918 [==============================] - 126s 2ms/step - loss: 0.5186 - auc: 0.6440 - val_lo
ss: 0.6291 - val_auc: 0.4753
Epoch 3/20
69918/69918 [==============================] - 124s 2ms/step - loss: 0.4955 - auc: 0.6477 - val_lo
ss: 0.5726 - val_auc: 0.4032
Epoch 4/20
69918/69918 [==============================] - 127s 2ms/step - loss: 0.4762 - auc: 0.6486 - val_lo
ss: 0.5371 - val_auc: 0.5080
Epoch 5/20
69918/69918 [==============================] - 123s 2ms/step - loss: 0.4638 - auc: 0.6582 - val_lo
ss: 0.5238 - val_auc: 0.2924
Epoch 6/20
69918/69918 [==============================] - 126s 2ms/step - loss: 0.4507 - auc: 0.6629 - val_lo
ss: 0.5313 - val_auc: 0.3268
Epoch 7/20
69918/69918 [==============================] - 126s 2ms/step - loss: 0.4420 - auc: 0.6605 - val_lo
ss: 0.4584 - val_auc: 0.6725
Epoch 8/20
69918/69918 [==============================] - 123s 2ms/step - loss: 0.4351 - auc: 0.6714 - val_lo
ss: 0.4556 - val_auc: 0.6496
Epoch 9/20
69918/69918 [==============================] - 124s 2ms/step - loss: 0.4310 - auc: 0.6747 - val_lo
ss: 0.4088 - val_auc: 0.7036
Epoch 10/20
69918/69918 [==============================] - 128s 2ms/step - loss: 0.4240 - auc: 0.6785 - val_lo
ss: 0.4148 - val_auc: 0.6975
Epoch 11/20
69918/69918 [==============================] - 125s 2ms/step - loss: 0.4204 - auc: 0.6886 - val_lo
ss: 0.4225 - val_auc: 0.7048
Epoch 12/20
69918/69918 [==============================] - 123s 2ms/step - loss: 0.4138 - auc: 0.6944 - val_lo
ss: 0.4278 - val_auc: 0.6768
Epoch 13/20
69918/69918 [==============================] - 127s 2ms/step - loss: 0.4128 - auc: 0.6947 - val_lo
ss: 0.4406 - val_auc: 0.7064
Epoch 14/20
69918/69918 [==============================] - 126s 2ms/step - loss: 0.4079 - auc: 0.7049 - val_lo
ss: 0.4369 - val_auc: 0.6363
Epoch 15/20
69918/69918 [==============================] - 129s 2ms/step - loss: 0.4080 - auc: 0.7047 - val_lo
ss: 0.4307 - val_auc: 0.7014
Epoch 16/20
69918/69918 [==============================] - 126s 2ms/step - loss: 0.4057 - auc: 0.7117 - val_lo
ss: 0.5151 - val_auc: 0.7096
```

```
Epoch 17/20
69918/69918 [==============================] - 123s 2ms/step - loss: 0.4039 - auc: 0.7174 - val_lo
ss: 0.4056 - val_auc: 0.7105
Epoch 18/20
69918/69918 [==============================] - 127s 2ms/step - loss: 0.4021 - auc: 0.7223 - val_lo
ss: 0.3961 - val_auc: 0.7148
Epoch 19/20
69918/69918 [==============================] - 125s 2ms/step - loss: 0.3969 - auc: 0.7309 - val_lo
ss: 0.4021 - val_auc: 0.7107
Epoch 20/20
69918/69918 [==============================] - 127s 2ms/step - loss: 0.3931 - auc: 0.7372 - val_lo
ss: 0.4022 - val_auc: 0.7160
```

In [2]:

```
!pip install svglib
```

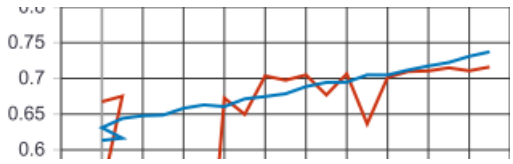
```
Collecting svglib
  Downloading
https://files.pythonhosted.org/packages/1f/d0/42227c7bfaba1b0c711006f8668019ae417ef6b31b1bede2247b8
5ad/svglib-1.0.0.tar.gz (899kB)
     |████████████████████████████████| 901kB 2.5MB/s
Collecting reportlab
  Downloading
https://files.pythonhosted.org/packages/63/a2/81b959f0d25660dc466bc0fe675c65e331f3264b4e39254a3b277
cec/reportlab-3.5.42-cp36-cp36m-manylinux2010_x86_64.whl (2.6MB)
     |████████████████████████████████| 2.6MB 7.4MB/s
Requirement already satisfied: lxml in /usr/local/lib/python3.6/dist-packages (from svglib)
(4.2.6)
Collecting tinycss2>=0.6.0
  Downloading
https://files.pythonhosted.org/packages/94/2c/4e501f9c351343c8ba10d70b5a7ca97cdab2690af043a6e52ada6
b6b/tinycss2-1.0.2-py3-none-any.whl (61kB)
     |████████████████████████████████| 71kB 8.2MB/s
Collecting cssselect2>=0.2.0
  Downloading
https://files.pythonhosted.org/packages/72/bb/9ad85eacc5f273b08bd5203a1d587479a93f27df9056e4e5f6327
d0e/cssselect2-0.3.0-py3-none-any.whl
Requirement already satisfied: pillow>=4.0.0 in /usr/local/lib/python3.6/dist-packages (from
reportlab->svglib) (7.0.0)
Requirement already satisfied: setuptools>=39.2.0 in /usr/local/lib/python3.6/dist-packages (from
tinycss2>=0.6.0->svglib) (46.4.0)
Requirement already satisfied: webencodings>=0.4 in /usr/local/lib/python3.6/dist-packages (from
tinycss2>=0.6.0->svglib) (0.5.1)
Building wheels for collected packages: svglib
  Building wheel for svglib (setup.py) ... done
  Created wheel for svglib: filename=svglib-1.0.0-cp36-none-any.whl size=26843
sha256=c8e337e04d0e03fa1af9ab9d6fdd54c146c128da7a1952dcd24f4728a765da1f
  Stored in directory:
/root/.cache/pip/wheels/e5/8e/78/7c1c7a612f8a87139b1b087b68c2c941976c2f24e1c0259cbb
Successfully built svglib
Installing collected packages: reportlab, tinycss2, cssselect2, svglib
Successfully installed cssselect2-0.3.0 reportlab-3.5.42 svglib-1.0.0 tinycss2-1.0.2
```

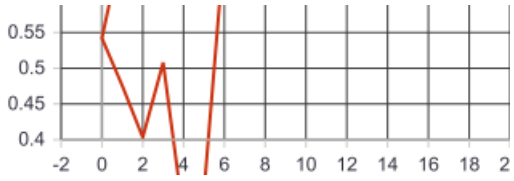
In [3]:

```
from svglib.svglib import svg2rlg
from reportlab.graphics import  renderPM
drawing = svg2rlg("drive/My Drive/epoch_auc_mod3.svg")
renderPM.drawToFile(drawing, "Plot1.png", fmt="PNG")
from IPython.display import Image
Image('Plot1.png')
```

```
Unable to find a clipping path with id clip_0
Unable to find a clipping path with id clip_1
```

Out[3]:



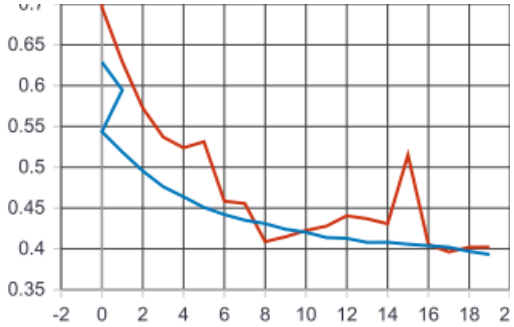


In [4]:

```
drawing = svg2rlg("drive/My Drive/epoch_loss_mod3.svg")
renderPM.drawToFile(drawing, "Plot2.png", fmt="PNG")
Image('Plot2.png')
```

```
Unable to find a clipping path with id clip_0
Unable to find a clipping path with id clip_1
```

Out[4]:



In [0]:

```
prob=model.predict([test_padded, test_other_text_norm.todense()])
#fpr, tpr, thresholds = roc_curve(y_test, prob)
scores_auc=(roc_auc_score(y_test1,prob))
print("AUC Score", scores_auc)
```

```
AUC Score 0.7103363705679722
```