In [0]:

```
# Credits: https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networ
ks-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
```

```
Using TensorFlow backend.
```

In [1]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%
b&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2
www.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly
ttps%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

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

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

1.2 preprocessing of `project_subject_categories`

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

1.3 preprocessing of `project_subject_subcategories`

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]))
```

1.3 Text preprocessing

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]:

```
project_data.head(2)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate

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 sklearn import preprocessing
from scipy.sparse import hstack
import numpy as np
def feat(xtrain,xtest,xcv,feature):
  from sklearn import preprocessing
  vectorizer = preprocessing.LabelEncoder()
  vectorizer.fit(list(xtrain[feature].values)+['Unknown'])
  categories_one_hot=vectorizer.transform(list(xtrain[feature].values))

  print(len(vectorizer.classes_))
  cattest= list(xtest[feature].values)
  catcv= list(xcv[feature].values)
  for unique_item in np.unique(cattest):
            if unique_item not in vectorizer.classes_:
                cattest = ['Unknown' if x==unique_item else x for x in cattest]
  for unique_item in np.unique(catcv):
            if unique_item not in vectorizer.classes_:
                catcv = ['Unknown' if x==unique_item else x for x in catcv]
  categories_one_hot_test = vectorizer.transform(cattest)
  categories_one_hot_cv = vectorizer.transform(catcv)
  print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
  return categories_one_hot,categories_one_hot_test,categories_one_hot_cv



def price(x,xtest,xcv):
  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_test = price_scalar.transform(xtest['price'].values.reshape(1, -1))
  price_standardized_cv = price_scalar.transform(xcv['price'].values.reshape(1, -1))
  price_standardized=np.transpose(price_standardized)
  price_standardized_test=np.transpose(price_standardized_test)
  price_standardized_cv=np.transpose(price_standardized_cv)
  return price_standardized,price_standardized_test,price_standardized_cv

def proj(x,xtest,xcv):
  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_test =
projects_scalar.transform(xtest['teacher_number_of_previously_posted_projects'].values.reshape(1,
-1))
  projects_standardized_cv =
projects_scalar.transform(xcv['teacher_number_of_previously_posted_projects'].values.reshape(1, -1)
)
  projects_standardized=np.transpose(projects_standardized)
  projects_standardized_test=np.transpose(projects_standardized_test)
  projects_standardized_cv=np.transpose(projects_standardized_cv)
  return projects_standardized,projects_standardized_test,projects_standardized_cv

def qty(x,xtest,xcv):

  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_test = qty_scalar.transform(xtest['quantity'].values.reshape(1, -1))
  qty_standardized_cv = qty_scalar.transform(xcv['quantity'].values.reshape(1, -1))
  qty_standardized=np.transpose(qty_standardized)
  qty_standardized_test=np.transpose(qty_standardized_test)
  qty_standardized_cv=np.transpose(qty_standardized_cv)
  return qty_standardized,qty_standardized_test,qty_standardized_cv


    #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,test_cat,cv_cat=feat(x_train,x_test,x_cv,'clean_categories')
train_subcat,test_subcat,cv_subcat=feat(x_train,x_test,x_cv,'clean_subcategories')
train_state,test_state,cv_state=feat(x_train,x_test,x_cv,'school_state')
train_prefix,test_prefix,cv_prefix=feat(x_train,x_test,x_cv,'teacher_prefix')
train_grade,test_grade,cv_grade=feat(x_train,x_test,x_cv,'project_grade_category')
```

```
51
Shape of matrix after one hot encodig  (69918,)
390
Shape of matrix after one hot encodig  (69918,)
52
Shape of matrix after one hot encodig  (69918,)
7
Shape of matrix after one hot encodig  (69918,)
5
Shape of matrix after one hot encodig  (69918,)
```

In [0]:

```
train_price,test_price,cv_price=price(x_train,x_test,x_cv)
train_proj,test_proj,cv_proj=proj(x_train,x_test,x_cv)
train_qty,test_qty,cv_qty=qty(x_train,x_test,x_cv)
```

In [0]:

```
train_numeral=np.concatenate((train_price,train_proj,train_qty),axis=1)
train_numeral.shape
```

Out[0]:

```
(69918, 3)
```

In [0]:

```
test_numeral=np.concatenate((test_price,test_proj,test_qty),axis=1)
print(test_numeral.shape)
cv_numeral=np.concatenate((cv_price,cv_proj,cv_qty),axis=1)
cv_numeral.shape
```

```
(21850, 3)
```

Out[0]:

```
(17480, 3)
```

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', 'e
ach', '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', "do
esn'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%|          | 159/69918 [00:00<00:43, 1588.24it/s]
```

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

```
100%|██████████| 69918/69918 [00:40<00:00, 1740.16it/s]
  1%|          | 166/21850 [00:00<00:13, 1658.34it/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, 1744.25it/s]
  1%|          | 179/17480 [00:00<00:09, 1779.93it/s]
```

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

```
100%|██████████| 17480/17480 [00:10<00:00, 1744.72it/s]
```

In [0]:

```

```

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]:

```
embedding_matrix.shape
```

Out[0]:

```
(47326, 300)
```

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]:

```
def seq(train_feat):
  train_seq=[]

  train_seq=list(map(lambda el:[el], train_feat))
  return train_seq
train_cat_seq=seq(train_cat)
test_cat_seq=seq(test_cat)
cv_cat_seq=seq(cv_cat)
train_subcat_seq=seq(train_subcat)
test_subcat_seq=seq(test_subcat)
cv_subcat_seq=seq(cv_subcat)
train_prefix_seq=seq(train_prefix)
test_prefix_seq=seq(test_prefix)
cv_prefix_seq=seq(cv_prefix)
train_state_seq=seq(train_state)
test_state_seq=seq(test_state)
cv_state_seq=seq(cv_state)
train_grade_seq=seq(train_grade)
test_grade_seq=seq(test_grade)
cv_grade_seq=seq(cv_grade)
```

In [0]:

```
def padding(xtrain,xtest,xcv):
  max_length=1
  tr_padded=pad_sequences(xtrain,maxlen=max_length,padding='post')
  te_padded=pad_sequences(xtest,maxlen=max_length,padding='post')
  c_padded=pad_sequences(xcv,maxlen=max_length,padding='post')
  print(tr_padded.shape)
  print(te_padded.shape)
  print(c_padded.shape)
  return tr_padded,te_padded,c_padded

train_cat_padded,test_cat_padded,cv_cat_padded=padding(train_cat_seq,test_cat_seq,cv_cat_seq)
train_subcat_padded,test_subcat_padded,cv_subcat_padded=padding(train_subcat_seq,test_subcat_seq,c
v_subcat_seq)
train_grade_padded,test_grade_padded,cv_grade_padded=padding(train_grade_seq,test_grade_seq,cv_grac
e_seq)
train_prefix_padded,test_prefix_padded,cv_prefix_padded=padding(train_prefix_seq,test_prefix_seq,c
v_prefix_seq)
train_state_padded,test_state_padded,cv_state_padded=padding(train_state_seq,test_state_seq,cv_stat
e_seq)
```

```
(69918, 1)
(21850, 1)
(17480, 1)
(69918, 1)
(21850, 1)
(17480, 1)
(69918, 1)
(21850, 1)
(17480, 1)
(69918, 1)
(21850, 1)
(17480, 1)
(69918, 1)
(21850, 1)
(17480, 1)
```

In [0]:

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

```
sub_categories_input = Input(shape=(1,), name='sub_categories_input')
proj_grade_input = Input(shape=(1,), name='proj_grade_input')
school_state_input = Input(shape=(1,), name='school_state_input')
tch_input = Input(shape=(1,), name='tch_input')
numeral_input=Input(shape=(train_numeral.shape[1],),name='numeral_input')
print(essay_input)
print(categories_input)
#print(train_proj.shape[1])
```

```
Tensor("essay_input:0", shape=(None, 300), dtype=float32)
Tensor("categories_input:0", shape=(None, 1), dtype=float32)
```

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(64,kernel_initializer='glorot_normal',kernel_regularizer=regularizers.l2(0.001),
return_sequences=True)(essay1)
essay1= Flatten()(essay1)
cat1=Embedding(input_dim= 51,output_dim=64,input_length=1)(categories_input)
cat1=Flatten()(cat1)
subcat1=Embedding(input_dim= 390,output_dim=64,input_length=1)(sub_categories_input)
subcat1=Flatten()(subcat1)
grade1=Embedding(input_dim= 5,output_dim=64,input_length=1)(proj_grade_input)
grade1=Flatten()(grade1)
state1=Embedding(input_dim=52,output_dim=64,input_length=1)(school_state_input)
state1=Flatten()(state1)
prefix1=Embedding(input_dim=7,output_dim=64,input_length=1)(tch_input)
prefix1=Flatten()(prefix1)
numeral1=Dense(64,activation='relu')(numeral_input)
```

In [0]:

```
final_x= concatenate([essay1,cat1,subcat1,grade1,state1,prefix1,numeral1])
```

In [0]:

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

In [0]:

```
from keras.initializers import he_normal
l1=Dense(32,activation='relu',kernel_initializer=he_normal(),kernel_regularizer=regularizers.l2(0.
001))(final_x)
#l1=BatchNormalization()(l1)
l1=Dropout(0.5)(l1)
l2=Dense(16,activation='relu',kernel_initializer=he_normal(),kernel_regularizer=regularizers.l2(0.
001))(l1)
#l2=BatchNormalization()(l2)
l2=Dropout(0.5)(l2)
l3=Dense(8,activation='relu',kernel_initializer=he_normal(),kernel_regularizer=regularizers.l2(0.00
1))(l2)
output=Dense(2,activation='softmax')(l3)
```

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]:

```
try:

  %tensorflow_version 2.x
except Exception:
  pass

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

In [0]:

```
from keras import callbacks
!rm -rf logs2/image
import tensorflow as tf
logdir = "logs2/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]:

```

```

In [0]:

```
%tensorboard --logdir logs2/image
import tensorflow as tf
from keras.models import Model
from sklearn.metrics import roc_auc_score
from keras.callbacks import TensorBoard
from keras.optimizers import Adam
from time import time
#from tf.keras import metrics
#tensor=TensorBoard(log_dir='logs/{}'.format(time()),write_graph=True)
model=Model(inputs=[essay_input,categories_input,sub_categories_input,proj_grade_input,school_state
_input,tch_input,numeral_input],outputs=output)
model.compile(optimizer=Adam(lr=0.0006),loss='categorical_crossentropy',metrics=[auc])
print(model.summary())
history=model.fit([train_padded,
train_cat,train_subcat,train_grade,train_state,train_prefix,train_numeral],y_train1, batch_size=32
00, epochs=20, verbose=1, validation_data=([cv_padded,cv_cat,cv_subcat,cv_grade,cv_state,cv_prefix,
cv_numeral], y_cv1),class_weight=class_wght,callbacks=[tensorboard_callback])
```

```
WARNING:tensorflow:From <ipython-input-31-fb0e7269b010>:2: py_func (from
tensorflow.python.ops.script_ops) is deprecated and will be removed in a future version.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.

Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
essay_input (InputLayer)        (None, 300)          0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 300, 300)     14197800    essay_input[0][0]
__________________________________________________________________________________________________
categories_input (InputLayer)   (None, 1)            0
__________________________________________________________________________________________________
sub_categories_input (InputLaye (None, 1)            0
__________________________________________________________________________________________________
proj_grade_input (InputLayer)   (None, 1)            0
```

```
__________________________________________________________________________________________________
school_state_input (InputLayer) (None, 1)            0
__________________________________________________________________________________________________
tch_input (InputLayer)          (None, 1)            0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 300, 64)      93440       embedding_1[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 1, 64)        3264        categories_input[0][0]
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 1, 64)        24960       sub_categories_input[0][0]
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, 1, 64)        320         proj_grade_input[0][0]
__________________________________________________________________________________________________
embedding_5 (Embedding)         (None, 1, 64)        3328        school_state_input[0][0]
__________________________________________________________________________________________________
embedding_6 (Embedding)         (None, 1, 64)        448         tch_input[0][0]
__________________________________________________________________________________________________
numeral_input (InputLayer)      (None, 3)            0
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 19200)        0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 64)           0           embedding_2[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 64)           0           embedding_3[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 64)           0           embedding_4[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 64)           0           embedding_5[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 64)           0           embedding_6[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           256         numeral_input[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 19584)        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, 32)           626720      concatenate_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 32)           0           dense_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 16)           528         dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 16)           0           dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 8)            136         dropout_2[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 2)            18          dense_4[0][0]
==================================================================================================
Total params: 14,951,218
Trainable params: 753,418
Non-trainable params: 14,197,800
__________________________________________________________________________________________________
None
Train on 69918 samples, validate on 17480 samples
Epoch 1/20
69918/69918 [==============================] - 350s 5ms/step - loss: 0.8569 - auc: 0.5194 - val_lo
ss: 0.7343 - val_auc: 0.6301
Epoch 2/20
69918/69918 [==============================] - 350s 5ms/step - loss: 0.7316 - auc: 0.5697 - val_lo
ss: 0.6744 - val_auc: 0.6722
Epoch 3/20
69918/69918 [==============================] - 343s 5ms/step - loss: 0.6555 - auc: 0.6102 - val_lo
ss: 0.6234 - val_auc: 0.6984
Epoch 4/20
69918/69918 [==============================] - 342s 5ms/step - loss: 0.6052 - auc: 0.6403 - val_lo
ss: 0.5947 - val_auc: 0.7080
Epoch 5/20
69918/69918 [==============================] - 338s 5ms/step - loss: 0.5681 - auc: 0.6491 - val_lo
ss: 0.5350 - val_auc: 0.7136
Epoch 6/20
69918/69918 [==============================] - 337s 5ms/step - loss: 0.5370 - auc: 0.6721 - val_lo
ss: 0.5231 - val_auc: 0.7200
```

```
Epoch 7/20
69918/69918 [==============================] - 333s 5ms/step - loss: 0.5160 - auc: 0.6835 - val_lo
ss: 0.5048 - val_auc: 0.7234
Epoch 8/20
69918/69918 [==============================] - 333s 5ms/step - loss: 0.5001 - auc: 0.6889 - val_lo
ss: 0.5038 - val_auc: 0.7271
Epoch 9/20
69918/69918 [==============================] - 335s 5ms/step - loss: 0.4873 - auc: 0.6980 - val_lo
ss: 0.4946 - val_auc: 0.7275
Epoch 10/20
69918/69918 [==============================] - 334s 5ms/step - loss: 0.4745 - auc: 0.7103 - val_lo
ss: 0.4845 - val_auc: 0.7279
Epoch 11/20
69918/69918 [==============================] - 333s 5ms/step - loss: 0.4670 - auc: 0.7135 - val_lo
ss: 0.4790 - val_auc: 0.7253
Epoch 12/20
69918/69918 [==============================] - 334s 5ms/step - loss: 0.4587 - auc: 0.7210 - val_lo
ss: 0.4683 - val_auc: 0.7260
Epoch 13/20
69918/69918 [==============================] - 336s 5ms/step - loss: 0.4530 - auc: 0.7259 - val_lo
ss: 0.4682 - val_auc: 0.7239
Epoch 14/20
69918/69918 [==============================] - 334s 5ms/step - loss: 0.4511 - auc: 0.7262 - val_lo
ss: 0.4601 - val_auc: 0.7231
Epoch 15/20
69918/69918 [==============================] - 336s 5ms/step - loss: 0.4426 - auc: 0.7375 - val_lo
ss: 0.4580 - val_auc: 0.7243
Epoch 16/20
69918/69918 [==============================] - 332s 5ms/step - loss: 0.4391 - auc: 0.7410 - val_lo
ss: 0.4535 - val_auc: 0.7150
Epoch 17/20
69918/69918 [==============================] - 336s 5ms/step - loss: 0.4350 - auc: 0.7468 - val_lo
ss: 0.4447 - val_auc: 0.7220
Epoch 18/20
69918/69918 [==============================] - 333s 5ms/step - loss: 0.4321 - auc: 0.7504 - val_lo
ss: 0.4484 - val_auc: 0.7090
Epoch 19/20
69918/69918 [==============================] - 333s 5ms/step - loss: 0.4303 - auc: 0.7530 - val_lo
ss: 0.4526 - val_auc: 0.7203
Epoch 20/20
69918/69918 [==============================] - 336s 5ms/step - loss: 0.4253 - auc: 0.7615 - val_lo
ss: 0.4493 - val_auc: 0.7182
```

In [3]:

```
!pip install svglib
```

```
Collecting svglib
  Downloading
https://files.pythonhosted.org/packages/1f/d0/42227c7bfaba1b0c711006f8668019ae417ef6b31b1bede2247b8
5ad/svglib-1.0.0.tar.gz (899kB)
     |████████████████████████████████| 901kB 3.5MB/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 18.9MB/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 7.4MB/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: webencodings>=0.4 in /usr/local/lib/python3.6/dist-packages (from
tinycss2>=0.6.0->svglib) (0.5.1)
Requirement already satisfied: setuptools>=39.2.0 in /usr/local/lib/python3.6/dist-packages (from
tinycss2>=0.6.0->svglib) (47.1.1)
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=ef5859651bdd9f153376f66838c808c257cecb1bd70fece5cdbd4d6da1117b4e
  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 [4]:

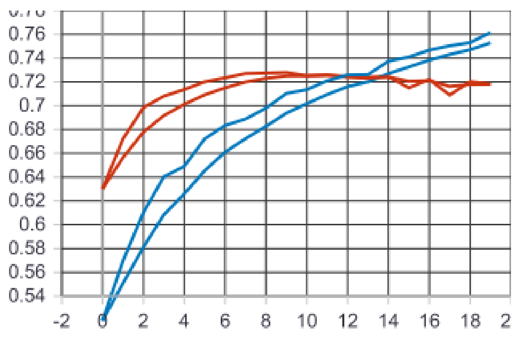
```
from svglib.svglib import svg2rlg
from reportlab.graphics import  renderPM
drawing = svg2rlg("/content/drive/My Drive/epoch_auc_mod1.svg")
renderPM.drawToFile(drawing, "Plot1.png", fmt="PNG")
```

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

In [5]:

```
from IPython.display import Image
Image('Plot1.png')
```

Out[5]:

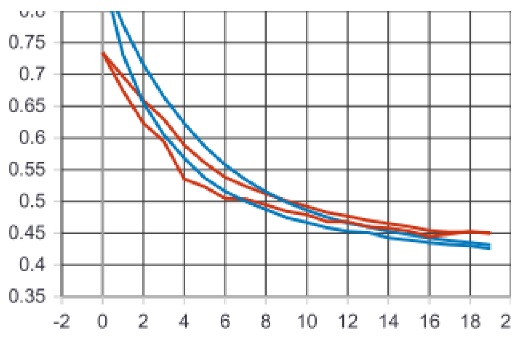


In [6]:

```
drawing = svg2rlg("/content/drive/My Drive/epoch_loss_mod1.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[6]:



In [0]:

```
prob=model.predict([test_padded,
test_cat,test_subcat,test_grade,test_state,test_prefix,test_numeral])
```

In [0]:

```
from sklearn.metrics import roc_auc_score
```

```
scores_auc=(roc_auc_score(y_test1,prob))
print("AUC Score", scores_auc)
```

```
AUC Score 0.7205234868595327
```

In [0]:

```
```