In [1]:

```
%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.utils import class_weight
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
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
import datetime

import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.initializers import RandomNormal
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Input, Embedding, Dense, Flatten, concate
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.callbacks import EarlyStopping
```

Splitting data into Train and Test

In [2]:

```
prepeocessed_data = pd.read_csv('preprocessed_data.csv')
prepeocessed_data.head(2)
```

Out[2]:

	Unnamed: 0	Unnamed: 0.1	id	teacher_id	teachei
0	0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	
1	1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	

2 rows × 21 columns

In [3]:

```
y = prepeocessed_data['project_is_approved'].values
X = prepeocessed_data.drop(['project_is_approved'], axis=1)
X.shape
```

Out[3]:

```
(109248, 20)
```

In [4]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stra
X_test.shape
```

Out[4]:

```
(21850, 20)
```

In [5]:

```
X_train.head(2)
```

Out[5]:

	Unnamed: 0	Unnamed: 0.1	id	teacher_id	te
78894	78894	158986	p205323	c49ee844ee9af3432a8ba3df2ff8d4db	
38598	38598	21691	p020686	92e0259c01b6953d0848a84e564b584b	

In [6]:

```
y_train
```

Out[6]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [7]:

```
X_test.head(2)
```

Out[7]:

	Unnamed: 0	Unnamed: 0.1	id	teacher_id	te
33848	33848	25121	p178007	23a5aaa55a7ced73d4ec6609f0d1e016	
40483	40483	11992	p228886	1135d7f31075de665a8feeb3bdae3d49	

In [8]:

```
y_test
```

Out[8]:

```
array([1, 1, 1, ..., 1, 1, 0], dtype=int64)
```

In [9]:

```
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')
```

```
812it [00:00, 8059.02it/s]

Loading Glove Model

1917495it [03:30, 9111.60it/s]

Done. 1917495  words loaded!
```

In [10]:

```
t = Tokenizer()
t.fit_on_texts(X_train['essay'].values)
vocab_size = len(t.word_index) + 1
# integer encode the documents
X_train_encoded_docs = t.texts_to_sequences(X_train['essay'].values)
X_test_encoded_docs = t.texts_to_sequences(X_test['essay'].values)

# pad documents to a max length of 300 words
max_length = 300
X_train_padded_docs = pad_sequences(X_train_encoded_docs, maxlen=max_length,
X_test_padded_docs = pad_sequences(X_test_encoded_docs, maxlen=max_length, pa

embedding_matrix = np.zeros((vocab_size, max_length))

for word, i in tqdm(t.word_index.items()):
    embedding_vector = model.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

print(X_train_padded_docs.shape)
print(X_test_padded_docs.shape)
```

```
100%|█████████████████████████████████████████████████████████
███████████████| 61662/61662 [00:00<00:00, 426393.12it/s]

(87398, 300)
(21850, 300)
```

In [11]:

```
print(vocab_size)
print(embedding_matrix.shape)
print(embedding_matrix[1])
```

```
61663
(61663, 300)
[-2.4837e-01 -4.5461e-01  3.9227e-02 -2.8422e-01 -3.1852e-02
2.6355e-01
 -4.6323e+00  1.3890e-02 -5.3928e-01 -8.4454e-02  6.1556e-02 -
4.1552e-01
 -1.4599e-01 -5.9321e-01 -2.8738e-02 -3.4991e-02 -2.9698e-01 -
7.9850e-02
  2.7312e-01  2.2040e-01 -8.9859e-02  8.8265e-04 -4.1991e-01 -
1.2536e-01
 -5.4629e-02  3.0550e-02  1.9340e-01 -6.3945e-02  2.7405e-02
5.1193e-02
 -3.8656e-01 -1.1085e-01  1.7259e-01  2.9804e-01 -3.5183e-01
1.3150e-01
 -5.4006e-01 -7.6677e-01 -5.5168e-04  1.3076e-01  2.5101e-02
6.2106e-01
 -2.4797e-01 -3.9790e-01 -3.6116e-01 -5.1967e-01  3.0138e-02 -
5.2436e-02
  6.9281e-02  3.5252e-02 -2.1402e-01  2.4836e-01 -1.5693e-01
1.2829e-01
  3.5425e-01 -1.6080e-01 -5.0720e-03 -3.0656e-01 -2.9514e-01 -
1.3554e-01
 -1.4385e-01 -4.0552e-01  5.7233e-01 -2.7670e-01  3.0519e-01
1.5586e-01
  1.6086e-02 -2.2009e-01  4.8589e-01 -4.1384e-01  2.0546e-01
4.0491e-01
  4.1558e-02 -1.3542e-01  2.2544e-01 -2.3629e-01  1.5193e-01 -
1.0859e-02
 -8.2662e-02 -5.5484e-01 -6.1584e-02 -1.1112e-01 -1.1982e-01 -
3.7064e-01
  1.6501e-01  4.4063e-01 -3.3883e-01 -5.7676e-01  5.0847e-01 -
3.5707e-02
 -5.9233e-02  3.0748e-02 -2.7689e-01 -7.0433e-02  2.7786e-02 -
5.9336e-01
 -2.8220e+00 -1.0052e-01  6.7168e-01 -1.7046e-01 -2.5902e-01
2.7938e-01
  3.9992e-01  3.7480e-02 -2.6409e-01 -2.6378e-01  2.0645e-01
1.7564e-01
 -8.0807e-02 -3.8376e-01  2.6602e-01  3.6214e-01 -9.5112e-02
3.5199e-01
 -8.6994e-01 -1.5747e-01 -2.2550e-01 -6.4948e-02 -2.4845e-01
1.5038e-01
 -3.2951e-01 -2.2285e-01 -2.5509e-02 -2.9725e-01 -3.7715e-01
8.9296e-02
 -3.4399e-02  3.3640e-01  3.5534e-01  3.8253e-01  1.7646e-01
1.3305e-01
```

```
 -3.2743e-01 -4.7115e-01  2.4673e-01 -1.5964e-01  1.8212e-01 -
4.1241e-01
  9.8565e-02  3.8118e-01  3.3043e-01  5.1987e-02 -2.1824e-01
2.2214e-01
 -5.9450e-02 -6.3743e-02  4.3723e-01  1.1068e-01  4.7444e-01
5.6891e-01
  3.1123e-01 -2.0272e-01  8.0078e-02 -4.3905e-01 -1.2246e-01 -
2.5057e-02
 -5.7162e-02  1.4250e-01  9.4468e-02  1.2991e-01  1.0444e-01 -
3.9447e-01
 -2.9337e-01 -2.0466e-01  2.0815e-01 -1.6010e-01 -1.4665e-01
5.4511e-01
  2.9740e-01 -2.2959e-01 -1.7050e-01 -6.2371e-02 -5.0399e-01 -
3.8000e-01
 -3.9528e-01  5.7552e-01 -4.6892e-01 -4.3308e-01  1.5018e-01 -
4.1179e-02
  6.2157e-01  1.9874e-02 -1.1969e-01 -2.5611e-01  2.6602e-01 -
3.7383e-01
  1.2936e-01 -5.0006e-02 -1.1554e-01 -1.7163e-01 -4.2430e-01
1.9844e-01
  5.0611e-01 -1.1093e-01 -1.3939e-01 -5.9377e-01  6.7338e-01
3.8497e-01
  6.2604e-01 -2.0128e-01  3.0058e-01 -1.3946e-01 -1.6186e-01
1.2168e-01
 -1.8410e-02  6.1356e-01 -1.9887e-01  1.9250e-01  8.4372e-03 -
5.0757e-01
  3.5858e-01 -4.9729e-01 -4.4725e-01  2.1423e-02 -2.0769e-01
8.3729e-02
  2.2032e-01  1.4404e-01  1.2590e-03 -4.4309e-01 -1.7242e-01 -
3.5300e-01
 -2.9477e-01  3.2898e-01 -3.1910e+00  3.8910e-01  3.5654e-01
5.2134e-02
  2.0576e-01 -8.8649e-02  1.6398e-01  1.1203e-01  2.8590e-01
2.8940e-01
 -4.4349e-01  9.1036e-01 -3.0902e-01 -1.3985e-01 -3.9499e-01 -
2.7299e-02
 -1.5201e-01  8.4418e-02 -3.7196e-01  4.9827e-02  1.4128e-01 -
1.5126e-01
 -1.6107e-01  4.0226e-03  1.6799e-01 -2.5429e-01 -1.5074e-01 -
5.7409e-01
 -1.5611e-01  6.8407e-02  2.4832e-01  1.6828e-01  7.2764e-02 -
8.6728e-02
  2.1982e-03  1.3593e-01  7.0224e-01 -4.5976e-01 -2.4506e-01 -
3.3874e-01
 -1.0952e-01  2.4698e-01 -5.5919e-01 -3.8866e-01 -1.3372e-01
9.1943e-02
 -1.0543e-01 -3.1319e-01 -2.9952e-01 -2.0611e-01  1.7976e-01
4.5800e-01
 -7.2402e-02  1.6118e-01 -4.1649e-01 -3.0103e-01  2.3234e-01 -
5.0139e-02
  1.0026e-01  3.8974e-01 -6.1342e-02  2.6626e-01 -1.5671e-01
7.5136e-02
 -4.2926e-01 -1.2025e-01  8.2736e-02 -6.2469e-01  4.4267e-02
```

```
6.0673e-01
 -1.2458e-01 -1.5443e-01 -1.6339e-01  5.3097e-02  1.5458e-01 -
3.8053e-01]
```

1.4 Encoding Categorical and Numerical features

In [15]:

```
from sklearn.preprocessing import LabelEncoder
import numpy as np


class LabelEncoderExt(object):
    def __init__(self):
        """
        It differs from LabelEncoder by handling new classes and providing a
        Unknown will be added in fit and transform will take care of new iten
        """
        self.label_encoder = LabelEncoder()
        # self.classes_ = self.label_encoder.classes_

    def fit(self, data_list):
        """
        This will fit the encoder for all the unique values and introduce unk
        :param data_list: A list of string
        :return: self
        """
        self.label_encoder = self.label_encoder.fit(list(data_list) + ['Unknc
        self.classes_ = self.label_encoder.classes_

        return self

    def transform(self, data_list):
        """
        This will transform the data_list to id list where the new values get
        :param data_list:
        :return:
        """
        new_data_list = list(data_list)
        for unique_item in np.unique(data_list):
            if unique_item not in self.label_encoder.classes_:
                new_data_list = ['Unknown' if x==unique_item else x for x in

        return self.label_encoder.transform(new_data_list)
```

1.4.1 encoding categorical features: clean_categories

In [16]:

```
from sklearn.preprocessing import LabelEncoder
vectorizer_cat = LabelEncoderExt()
vectorizer_cat.fit(X_train['clean_categories'].values) # fit has to happen or

X_train_cc_ohe = vectorizer_cat.transform(X_train['clean_categories'].values)
X_test_cc_ohe = vectorizer_cat.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_cc_ohe.shape, y_train.shape)
print(X_test_cc_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.2 encoding categorical features: clean_subcategories

In [19]:

```
vectorizer_subcat = LabelEncoderExt()
vectorizer_subcat.fit(X_train['clean_subcategories'].values) # fit has to hap

X_train_csc_ohe = vectorizer_subcat.transform(X_train['clean_subcategories'].
X_test_csc_ohe = vectorizer_subcat.transform(X_test['clean_subcategories'].va

print("After vectorizations")
print(X_train_csc_ohe.shape, y_train.shape)
print(X_test_csc_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.3 encoding categorical features: school_state

In [20]:

```
vectorizer_school_state = LabelEncoderExt()
vectorizer_school_state.fit(X_train['school_state'].values)

X_train_state_ohe = vectorizer_school_state.transform(X_train['school_state']
X_test_state_ohe = vectorizer_school_state.transform(X_test['school_state'].v

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.4 encoding categorical features: teacher_prefix

In [21]:

```
vectorizer_prefix = LabelEncoderExt()
vectorizer_prefix.fit(X_train['teacher_prefix'].values)

X_train_teacher_ohe = vectorizer_prefix.transform(X_train['teacher_prefix'].v
X_test_teacher_ohe = vectorizer_prefix.transform(X_test['teacher_prefix'].val

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.5 encoding categorical features: project_grade_category

In [22]:

```
vectorizer_grade = LabelEncoderExt()
vectorizer_grade.fit(X_train['project_grade_category'].values)

X_train_grade_ohe = vectorizer_grade.transform(X_train['project_grade_categor
X_test_grade_ohe = vectorizer_grade.transform(X_test['project_grade_category'

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
```

```
After vectorizations
(87398,) (87398,)
(21850,) (21850,)
```

1.4.6 encoding numerical features: price

In [24]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(X_train['price'].values.reshape(-1,1))
X_train_price=scaler.transform(X_train['price'].values.reshape(-1,1))
X_test_price=scaler.transform(X_test['price'].values.reshape(-1,1))


print(X_train_price.shape)
print(X_test_price.shape)
print(X_train_price)
print(X_test_price)
```

```
(87398, 1)
(21850, 1)
[[0.00684614]
 [0.05806564]
 [0.02537021]
 ...
 [0.02702549]
 [0.04971725]
 [0.04152389]]
[[0.00693015]
 [0.1253258 ]
 [0.02181162]
 ...
 [0.07185593]
 [0.02632337]
 [0.01072578]]
```

1.4.7 encoding numerical features: teacher_number_of_previously_posted_projects

In [25]:

```
scaler = MinMaxScaler()
scaler.fit(X_train['teacher_number_of_previously_posted_projects'].values.res
X_train_pre_proj=scaler.transform(X_train['teacher_number_of_previously_poste
X_test_pre_proj=scaler.transform(X_test['teacher_number_of_previously_posted_


print(X_train_pre_proj.shape)
print(X_test_pre_proj.shape)
```

```
(87398, 1)
(21850, 1)
```

1.4.8 encoding numerical features: quantity

In [26]:

```
scaler = MinMaxScaler()
scaler.fit(X_train['quantity'].values.reshape(-1,1))
X_train_quantity=scaler.transform(X_train['quantity'].values.reshape(-1,1))
X_test_quantity=scaler.transform(X_test['quantity'].values.reshape(-1,1))


print(X_train_quantity.shape)
print(X_test_quantity.shape)
```

```
(87398, 1)
(21850, 1)
```

In [27]:

```
X_train_numerals=np.concatenate((X_train_pre_proj,X_train_quantity,X_train_pr
X_test_numerals=np.concatenate((X_test_pre_proj,X_test_quantity,X_test_price)

print(X_train_numerals.shape)
print(X_test_numerals.shape)
```

```
(87398, 3)
(21850, 3)
```

In [28]:

```
print('vocabulary size is ',vocab_size)
```

```
vocabulary size is  61663
```

In [1]:

```
'''
These cells are to load the data if the system crashes during training
'''
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.utils import class_weight
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
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
import datetime

import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.initializers import RandomNormal
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Input, Embedding, Dense, Flatten, concate
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, Reduce

vocab_size = 61663
```

In [29]:

```
#Saving processed data

from sklearn.externals import joblib

joblib.dump(embedding_matrix, 'embedding_matrix.pkl')
joblib.dump(X_train_padded_docs, 'X_train_padded_docs.pkl')
joblib.dump(X_train_cc_ohe, 'X_train_cc_ohe.pkl')
joblib.dump(X_train_csc_ohe, 'X_train_csc_ohe.pkl')
joblib.dump(X_train_teacher_ohe, 'X_train_teacher_ohe.pkl')
joblib.dump(X_train_grade_ohe, 'X_train_grade_ohe.pkl')
joblib.dump(X_train_state_ohe, 'X_train_state_ohe.pkl')
joblib.dump(X_train_numerals, 'X_train_numerals.pkl')
joblib.dump(y_train, 'y_train.pkl')


joblib.dump(X_test_padded_docs, 'X_test_padded_docs.pkl')
joblib.dump(X_test_cc_ohe, 'X_test_cc_ohe.pkl')
joblib.dump(X_test_csc_ohe, 'X_test_csc_ohe.pkl')
joblib.dump(X_test_teacher_ohe, 'X_test_teacher_ohe.pkl')
joblib.dump(X_test_grade_ohe, 'X_test_grade_ohe.pkl')
joblib.dump(X_test_state_ohe, 'X_test_state_ohe.pkl')
joblib.dump(X_test_numerals, 'X_test_numerals.pkl')
joblib.dump(y_test, 'y_test.pkl')
```

Out[29]:

```
['y_test.pkl']
```

In [2]:

```
#Loading processed data

from sklearn.externals import joblib

embedding_matrix = joblib.load('embedding_matrix.pkl')
X_train_padded_docs = joblib.load('X_train_padded_docs.pkl')
X_train_cc_ohe = joblib.load('X_train_cc_ohe.pkl')
X_train_csc_ohe = joblib.load('X_train_csc_ohe.pkl')
X_train_teacher_ohe = joblib.load('X_train_teacher_ohe.pkl')
X_train_grade_ohe = joblib.load('X_train_grade_ohe.pkl')
X_train_state_ohe = joblib.load('X_train_state_ohe.pkl')
X_train_numerals = joblib.load('X_train_numerals.pkl')
y_train = joblib.load('y_train.pkl')


X_test_padded_docs = joblib.load('X_test_padded_docs.pkl')
X_test_cc_ohe = joblib.load('X_test_cc_ohe.pkl')
X_test_csc_ohe = joblib.load('X_test_csc_ohe.pkl')
X_test_teacher_ohe = joblib.load('X_test_teacher_ohe.pkl')
X_test_grade_ohe = joblib.load('X_test_grade_ohe.pkl')
X_test_state_ohe = joblib.load('X_test_state_ohe.pkl')
X_test_numerals = joblib.load('X_test_numerals.pkl')
y_test = joblib.load('y_test.pkl')
```

In [3]:

```
from sklearn.metrics import roc_auc_score
def auc( y_true, y_pred ) :
    score = tf.py_func( lambda y_true, y_pred : roc_auc_score( y_true, y_pred
                        [y_true, y_pred],
                        'float32',
                        stateful=True,
                        name='sklearnAUC' )
    return score
```

Model's architecture

In [4]:

```
# Essay Layers
essay_input = Input(shape=(len(X_train_padded_docs[0]),), name='essay_input'

essay_input_1 = Embedding(input_dim=vocab_size,output_dim=300, input_length=
               weights=[embedding_matrix], trainable=False)(e

essay_input_1 = CuDNNLSTM(units = 64,
               kernel_initializer= 'he_normal',
               return_sequences=True)(essay_input_1)
essay_input_1 = Flatten()(essay_input_1)

# Category Layers
categories_input = Input(shape=(1,), name='categories_input')

categories_input_1= Embedding(input_dim=len(set(X_train_cc_ohe)), output_dim
categories_input_1 = Flatten()(categories_input_1)

# Sub Category Layers
sub_categories_input = Input(shape=(1,), name='sub_categories_input')

sub_categories_input_1 = Embedding(input_dim=len(set(X_train_csc_ohe)), outpu
sub_categories_input_1 = Flatten()(sub_categories_input_1)

# Grade Layers
proj_grade_input = Input(shape=(1,), name='proj_grade_input')

proj_grade_input_1 = Embedding(input_dim=len(set(X_train_grade_ohe)), output_
proj_grade_input_1 = Flatten()(proj_grade_input_1)

# School Layers
school_state_input = Input(shape=(1,), name='school_state_input')

school_state_input_1 = Embedding(input_dim=len(set(X_train_state_ohe)), outpu
school_state_input_1 = Flatten()(school_state_input_1)

# Teacher Prefix Layers
tch_input = Input(shape=(1,), name='tch_input')

tch_input_1= Embedding(input_dim=len(set(X_train_teacher_ohe)), output_dim =
tch_input_1 = Flatten()(tch_input_1)

# Numerical Layers
numeral_input = Input(shape=(X_train_numerals.shape[1],),name='numeral_input

numeral_input_1 = Dense(units = 64,
              activation = 'relu',
              kernel_initializer = 'he_normal')(numeral_input)

# Cconcatinating all the above Layers
x = concatenate([essay_input_1, categories_input_1, sub_categories_input_1,
                     proj_grade_input_1, school_state_input_1,
                     tch_input_1, numeral_input_1])

# Dense Layers
output = Dense(units = 624,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(x)
output = Dropout(0.7)(output)
output = BatchNormalization()(output)

output = Dropout(0.7)(output)
output = BatchNormalization()(output)
output = Dense(units = 512,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(output)



output = Dropout(0.8)(output)
output = BatchNormalization()(output)
output = Dense(units = 512,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(output)
output = Dropout(0.45)(output)
output =Dense(1, activation='sigmoid', name='output')(output)

model_1 = Model(inputs = [essay_input, categories_input, sub_categories_input
                            proj_grade_input, school_state_input, tch_input,r
                  outputs=output)

model_1.compile(loss = 'binary_crossentropy',
                  optimizer = 'adam',
                  metrics = ['accuracy', auc])
```

```
WARNING:tensorflow:From c:\users\addu\appdata\local\program
s\python\python37\lib\site-packages\tensorflow\python\keras
\initializers.py:119: calling RandomUniform.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated an
d will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead o
f passing it to the constructor
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In Tenso
rFlow 2.x, dropout() uses dropout rate instead of keep_pro
b. Please ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In Tenso
rFlow 2.x, dropout() uses dropout rate instead of keep_pro
b. Please ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.8 (>0.5). In Tenso
rFlow 2.x, dropout() uses dropout rate instead of keep_pro
b. Please ensure that this is intended.
WARNING:tensorflow:From c:\users\addu\appdata\local\program
s\python\python37\lib\site-packages\tensorflow\python\ops\i
nit_ops.py:1251: calling VarianceScaling.__init__ (from ten
```

Training the model with train data

In [5]:

```
log_dir="logs\\model_1_log" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S
tensorboard = TensorBoard(log_dir=log_dir)
mcp_save = ModelCheckpoint('.model1_best_weights.hdf5', save_best_only=True,
callbacks_list = [mcp_save,tensorboard]
```

In [6]:

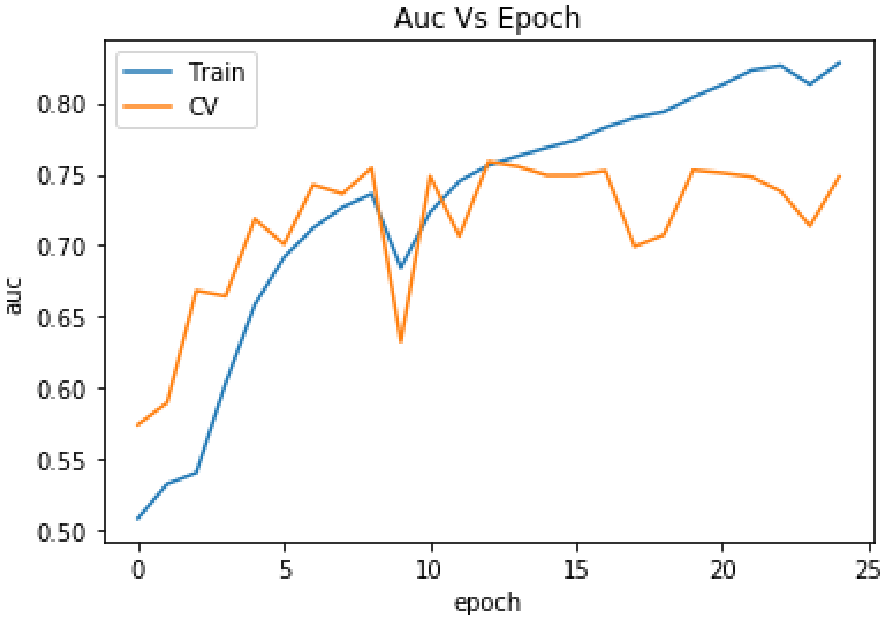
```
history = model_1.fit([X_train_padded_docs, X_train_cc_ohe, X_train_csc_ohe,
                      X_train_grade_ohe, X_train_state_ohe, X_train_teacher_c
                     y_train,
                     batch_size = 1024,
                     epochs = 25,
                     callbacks=callbacks_list,
                     validation_split = 0.2,
                     verbose = 2)
```

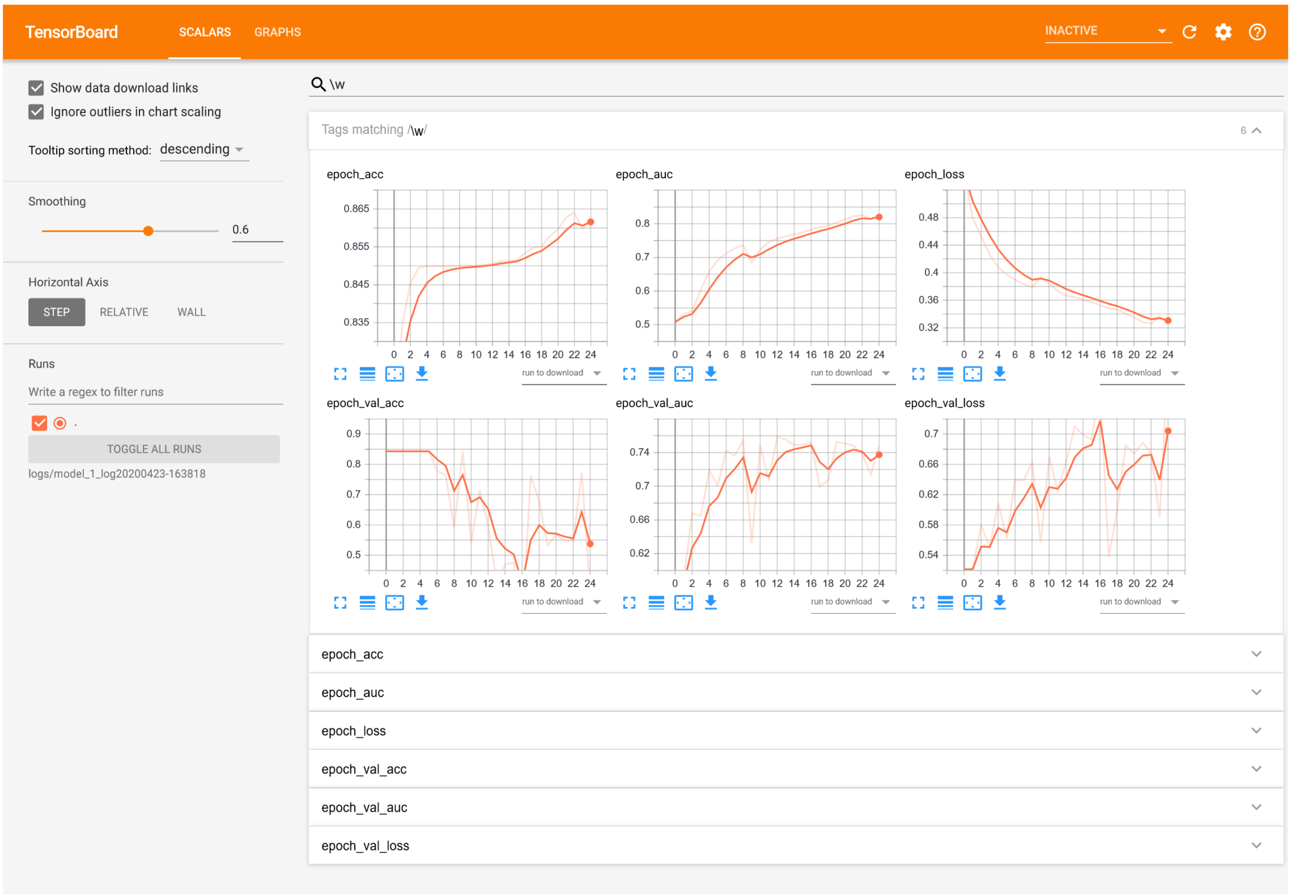
```
 - val_loss: 0.5989 - val_acc: 0.8432 - val_auc: 0.6524

Epoch 11/25
69918/69918 - 4s - loss: 0.3842 - acc: 0.8500 - auc: 0.7233
- val_loss: 0.6702 - val_acc: 0.5412 - val_auc: 0.7487
Epoch 12/25
69918/69918 - 4s - loss: 0.3735 - acc: 0.8503 - auc: 0.7451
- val_loss: 0.6248 - val_acc: 0.7154 - val_auc: 0.7064
Epoch 13/25
69918/69918 - 5s - loss: 0.3670 - acc: 0.8506 - auc: 0.7563
- val_loss: 0.6633 - val_acc: 0.5947 - val_auc: 0.7590
Epoch 14/25
69918/69918 - 4s - loss: 0.3643 - acc: 0.8513 - auc: 0.7627
- val_loss: 0.7102 - val_acc: 0.4108 - val_auc: 0.7555
Epoch 15/25
69918/69918 - 4s - loss: 0.3609 - acc: 0.8512 - auc: 0.7687
- val_loss: 0.6991 - val_acc: 0.4696 - val_auc: 0.7492
Epoch 16/25
69918/69918 - 4s - loss: 0.3574 - acc: 0.8517 - auc: 0.7741
- val_loss: 0.6925 - val_acc: 0.4758 - val_auc: 0.7492
Epoch 17/25
```

Auc Vs Epoch

In [7]:

```
# summarize history for accuracy
plt.plot(history.history['auc'])
plt.plot(history.history['val_auc'])
plt.title('Auc Vs Epoch')
plt.ylabel('auc')
plt.xlabel('epoch')
plt.legend(['Train', 'CV'], loc='right bottom')
plt.show()
```





☑ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: descending

Smoothing 0.6

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

☑ ○ .

TOGGLE ALL RUNS

logs/model_1_log20200423-163818

\w

Tags matching /\w/ 6

epoch_acc

epoch_auc

epoch_loss

epoch_val_acc

epoch_val_auc

epoch_val_loss

epoch_acc

epoch_auc

epoch_loss

epoch_val_acc

epoch_val_auc

epoch_val_loss

Testing the model with test data

In [8]:

```
# Essay Layers
essay_input = Input(shape=(len(X_train_padded_docs[0]),), name='essay_input')

essay_input_1 = Embedding(input_dim=vocab_size,output_dim=300, input_length=
                              weights=[embedding_matrix], trainable=False)(es

essay_input_1 = CuDNNLSTM(units = 64,
                              kernel_initializer= 'he_normal',
                              return_sequences=True)(essay_input_1)
essay_input_1 = Flatten()(essay_input_1)

# Category Layers
categories_input = Input(shape=(1,), name='categories_input')

categories_input_1= Embedding(input_dim=len(set(X_train_cc_ohe)), output_dim
categories_input_1 = Flatten()(categories_input_1)

# Sub Category Layers
sub_categories_input = Input(shape=(1,), name='sub_categories_input')

sub_categories_input_1 = Embedding(input_dim=len(set(X_train_csc_ohe)), outpu
sub_categories_input_1 = Flatten()(sub_categories_input_1)

# Grade Layers
proj_grade_input = Input(shape=(1,), name='proj_grade_input')

proj_grade_input_1 = Embedding(input_dim=len(set(X_train_grade_ohe)), output_
proj_grade_input_1 = Flatten()(proj_grade_input_1)

# School Layers
school_state_input = Input(shape=(1,), name='school_state_input')

school_state_input_1 = Embedding(input_dim=len(set(X_train_state_ohe)), outpu
school_state_input_1 = Flatten()(school_state_input_1)

# Teacher Prefix Layers
tch_input = Input(shape=(1,), name='tch_input')

tch_input_1= Embedding(input_dim=len(set(X_train_teacher_ohe)), output_dim =
tch_input_1 = Flatten()(tch_input_1)

# Numerical Layers
numeral_input = Input(shape=(X_train_numerals.shape[1],),name='numeral_input'

numeral_input_1 = Dense(units = 64,
                            activation = 'relu',
                            kernel_initializer = 'he_normal')(numeral_input)

# Cconcatinating all the above Layers
x = concatenate([essay_input_1, categories_input_1, sub_categories_input_1,
                     proj_grade_input_1, school_state_input_1,
                     tch_input_1, numeral_input_1])

# Dense Layers
output = Dense(units = 624,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(x)
output = Dropout(0.7)(output)
output = BatchNormalization()(output)

output = Dropout(0.7)(output)
output = BatchNormalization()(output)
output = Dense(units = 512,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(output)


output = Dropout(0.8)(output)
output = BatchNormalization()(output)
output = Dense(units = 512,
                  activation = 'relu',
                            kernel_initializer = 'he_normal')(output)
output = Dropout(0.45)(output)
output =Dense(1, activation='sigmoid', name='output')(output)

model_1 = Model(inputs = [essay_input, categories_input, sub_categories_input
                            proj_grade_input, school_state_input, tch_input,
                  outputs=output)

model_1.compile(loss = 'binary_crossentropy',
                  optimizer = 'adam',
                  metrics = ['accuracy', auc])
```

```
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFl
ow 2.x, dropout() uses dropout rate instead of keep_prob. Plea
se ensure that this is intended.
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFl
ow 2.x, dropout() uses dropout rate instead of keep_prob. Plea
se ensure that this is intended.
```

In [9]:

```
#Loading the best weights obtained from traing model on train data
model_1.load_weights(".model1_best_weights.hdf5")
```

In [10]:

```
score=model_1.evaluate([X_test_padded_docs, X_test_cc_ohe, X_test_csc_ohe,
                                 X_test_grade_ohe, X_test_state_ohe, X_test_t
                           batch_size=1024, verbose=0)[2]
```

Auc score of model on unseen Test data

In [11]:

```
print('Test Auc obtained is ',score)
```

```
Test Auc obtained is  0.75348437
```

In [12]:

```
model_1.summary()
```

```
Model: "model_1"
______________________________________________________________
____________________________________
Layer (type)                    Output Shape         Param #
Connected to
==============================================================
====================================
essay_input (InputLayer)        [(None, 300)]        0
______________________________________________________________
____________________________________
embedding_6 (Embedding)         (None, 300, 300)     18498900
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
______________________________________________________________
____________________________________
cu_dnnlstm_1 (CuDNNLSTM)        (None, 300, 64)      93696
embedding_6[0][0]
______________________________________________________________
____________________________________
embedding_7 (Embedding)         (None, 1, 16)        816
categories_input[0][0]
______________________________________________________________
____________________________________
embedding_8 (Embedding)         (None, 1, 64)        25216
sub_categories_input[0][0]
______________________________________________________________
____________________________________
embedding_9 (Embedding)         (None, 1, 64)        256
proj_grade_input[0][0]
______________________________________________________________
____________________________________
embedding_10 (Embedding)        (None, 1, 64)        3264
school_state_input[0][0]
______________________________________________________________
```

```
____________________________________
embedding_11 (Embedding)        (None, 1, 64)        320
tch_input[0][0]
______________________________________________________________
____________________________________
numeral_input (InputLayer)      [(None, 3)]          0
______________________________________________________________
____________________________________
flatten_6 (Flatten)             (None, 19200)        0
cu_dnnlstm_1[0][0]
______________________________________________________________
____________________________________
flatten_7 (Flatten)             (None, 16)           0
embedding_7[0][0]
______________________________________________________________
____________________________________
flatten_8 (Flatten)             (None, 64)           0
embedding_8[0][0]
______________________________________________________________
____________________________________
flatten_9 (Flatten)             (None, 64)           0
embedding_9[0][0]
______________________________________________________________
____________________________________
flatten_10 (Flatten)            (None, 64)           0
embedding_10[0][0]
______________________________________________________________
____________________________________
flatten_11 (Flatten)            (None, 64)           0
embedding_11[0][0]
______________________________________________________________
____________________________________
dense_4 (Dense)                 (None, 64)           256
numeral_input[0][0]
______________________________________________________________
____________________________________
concatenate_1 (Concatenate)     (None, 19536)        0
flatten_6[0][0]

flatten_7[0][0]

flatten_8[0][0]

flatten_9[0][0]

flatten_10[0][0]

flatten_11[0][0]

dense_4[0][0]
______________________________________________________________
____________________________________
dense_5 (Dense)                 (None, 624)          12191088
```

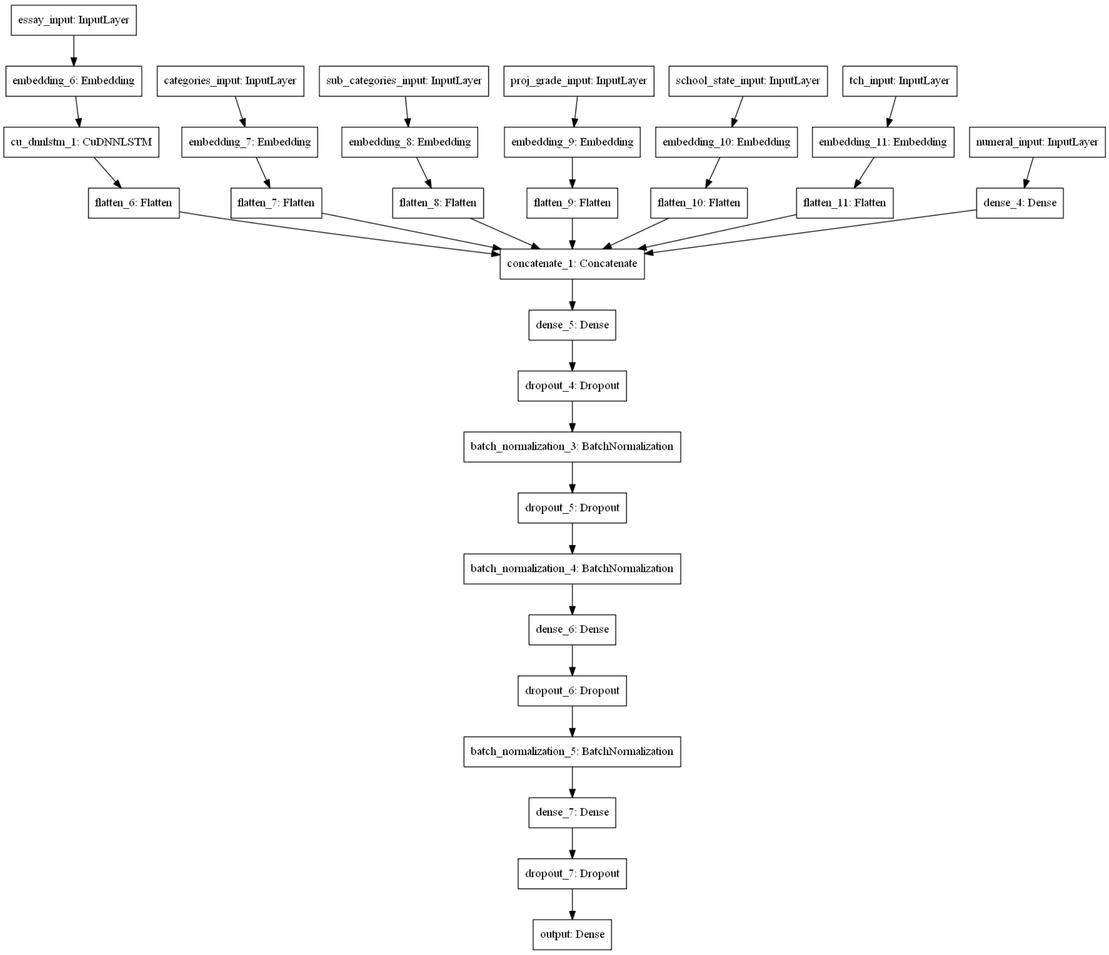
```
concatenate_1[0][0]
______________________________________________________________
____________________________________
dropout_4 (Dropout)             (None, 624)          0
dense_5[0][0]
______________________________________________________________
____________________________________
batch_normalization_3 (BatchNor (None, 624)          2496
dropout_4[0][0]
______________________________________________________________
____________________________________
dropout_5 (Dropout)             (None, 624)          0
batch_normalization_3[0][0]
______________________________________________________________
____________________________________
batch_normalization_4 (BatchNor (None, 624)          2496
dropout_5[0][0]
______________________________________________________________
____________________________________
dense_6 (Dense)                 (None, 512)          320000
batch_normalization_4[0][0]
______________________________________________________________
____________________________________
dropout_6 (Dropout)             (None, 512)          0
dense_6[0][0]
______________________________________________________________
____________________________________
batch_normalization_5 (BatchNor (None, 512)          2048
dropout_6[0][0]
______________________________________________________________
____________________________________
dense_7 (Dense)                 (None, 512)          262656
batch_normalization_5[0][0]
______________________________________________________________
____________________________________
dropout_7 (Dropout)             (None, 512)          0
dense_7[0][0]
______________________________________________________________
____________________________________
output (Dense)                  (None, 1)            513
dropout_7[0][0]
==============================================================
====================================
Total params: 31,404,021
Trainable params: 12,901,601
Non-trainable params: 18,502,420
______________________________________________________________
____________________________________
```

Model's Final Architecture

In [13]:

```
#drawing models
tf.keras.utils.plot_model(
    model_1,
    show_shapes=False,
    show_layer_names=True,
    rankdir='TB'
)
```

Out[13]:



Summary

- Created tensorflow model using text, categorical and numerical layers.
- Performed hyper parameter tuning manually on number of layers, activation functions and optimizers.
- Trained the network using the best obtained hyper parameters.
- Tested the model using Test data and obtained test aucroc score of 0.7534.
- Printed the summary of the model along with it's image.