In [1]:
```
!curl --header 'Host: doc-0s-14-docs.googleusercontent.com' --user-agent 'Mozi
lla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0' --
header 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/we
bp,*/*;q=0.8' --header 'Accept-Language: en-US,en;q=0.5' --referer 'https://dr
ive.google.com/drive/u/0/folders/1CJnItndeSSJu7aragQoXWZS9-0apN6pp' --cookie
'AUTH_iiuqqtdlb7b4kmj4id73smj9p4bkh7lp_nonce=2p5o99bi7v53k; _ga=GA1.2.17131332
74.1589641238' --header 'Upgrade-Insecure-Requests: 1' 'https://doc-0s-14-doc
s.googleusercontent.com/docs/securesc/p82isa6abc7tr2v14rq0r7o21rmk0tog/mtj3ppf
ltagf4k93dtdcfuok08fcuana/1592312550000/00484516897554883881/01701982909666782
580/1GpATd_pM4mcnWWIs28-s1lgqdAg2Wdv-?e=download&authuser=0&nonce=2p5o99bi7v53
k&user=01701982909666782580&hash=ffe7t5e2qqvah9jjco5lkprtsi5b226a' --output 'p
reprocessed_data.csv'
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
nt
                                 Dload  Upload   Total   Spent    Left  Speed
100  118M    0  118M    0     0  82.8M      0 --:--:--  0:00:01 --:--:-- 82.7
M
```

In [2]:
```
!curl --header 'Host: doc-0g-14-docs.googleusercontent.com' --user-agent 'Mozi
lla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0' --
header 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/we
bp,*/*;q=0.8' --header 'Accept-Language: en-US,en;q=0.5' --referer 'https://dr
ive.google.com/drive/u/0/folders/1CJnItndeSSJu7aragQoXWZS9-0apN6pp' --cookie
'AUTH_iiuqqtdlb7b4kmj4id73smj9p4bkh7lp=01701982909666782580|1592312550000|c9rk
fi4qpmgl40ac354q0ghidg0vp6n6; _ga=GA1.2.1713133274.1589641238' --header 'Upgra
de-Insecure-Requests: 1' 'https://doc-0g-14-docs.googleusercontent.com/docs/se
curesc/p82isa6abc7tr2v14rq0r7o21rmk0tog/2simpt93fi963lff5kc9oinro0a9ofkn/15923
12550000/00484516897554883881/01701982909666782580/1pGd5tLwA30M7wkbJKdXHaae9tY
VDICJ_?e=download&authuser=0' --output 'glove_vectors'
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
nt
                                 Dload  Upload   Total   Spent    Left  Speed
100  121M    0  121M    0     0  61.6M      0 --:--:--  0:00:01 --:--:-- 61.6
M
```

DonorsChoose

Assignment-14: Apply LSTM on Donors Choose dataset

abhipise2704@gmail.com_14

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

In [0]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

warnings.filterwarnings("ignore", category=DeprecationWarning)

warnings.warn("this will not show", DeprecationWarning)

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

In [4]:
```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers import Input, Embedding, LSTM, Dropout, BatchNormalization,
Dense, concatenate, Flatten, Conv1D, MaxPool1D, LeakyReLU, ELU, SpatialDropout
1D, MaxPooling1D, GlobalAveragePooling1D, GlobalMaxPooling1D
from keras.preprocessing.text import Tokenizer, one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Model, load_model
from keras import regularizers
from keras.optimizers import *
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard, Reduc
eLROnPlateau
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
#!pip install tensorboardcolab
from sklearn.preprocessing import StandardScaler
from keras.utils import np_utils
```

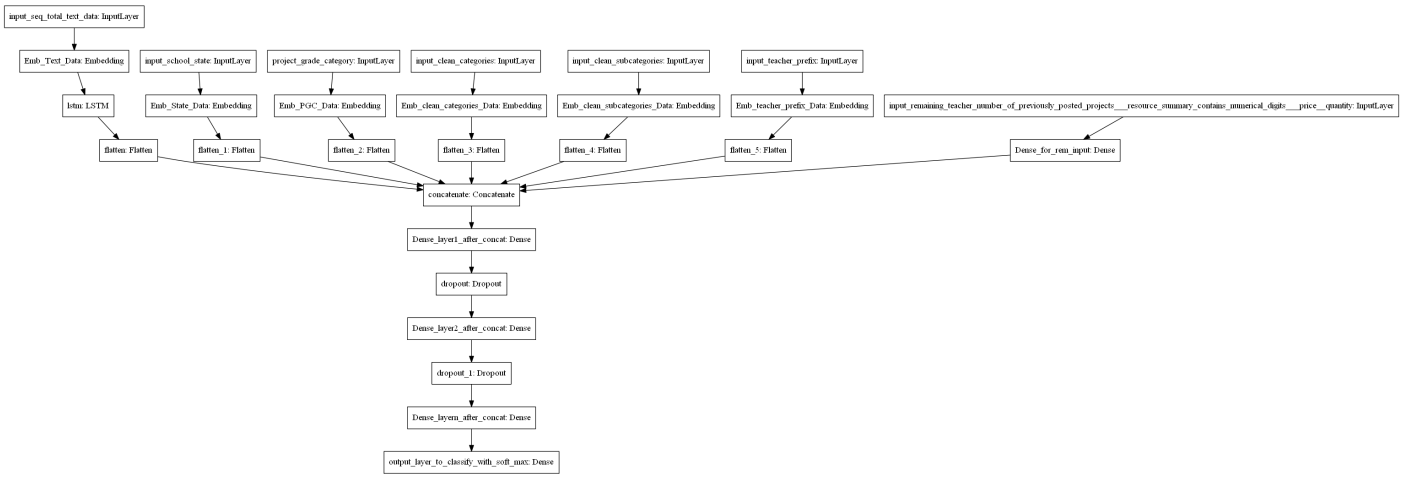
```
Using TensorFlow backend.
```

Assignment : 14

1. Download the preprocessed DonorsChoose data from here Dataset (https://drive.google.com/file/d/1GU3LIJJ3zS1xLXXe-sdItSJHtI5txjVO/view?usp=sharing)
2. Split the data into train, cv, and test
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' (https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics) as a metric. check this (https://datascience.stackexchange.com/a/20192) for using auc as a metric. you need to print the AUC value for each epoch. Note: you should NOT use the tf.metric.auc
5. You are free to choose any number of layers/hiddden units but you have to use same type of architectures shown below.
6. You can use any one of the optimizers and choice of Learning rate and momentum, resources: cs231n class notes (http://cs231n.github.io/neural-networks-3/), cs231n class video (https://www.youtube.com/watch?v=hd_KFJ5ktUc).
7. You should Save the best model weights.
8. For all the model's use TensorBoard (https://www.youtube.com/watch?v=2U6Jl7oqRkM) and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots and include those images in .ipynb notebook and PDF.
9. Use Categorical Cross Entropy as Loss to minimize.
10. try to get AUC more than 0.8 for atleast one model

Model-1

Build and Train deep neural network as shown below



ref: https://i.imgur.com/w395Yk9.png (https://i.imgur.com/w395Yk9.png)

- **Input_seq_total_text_data** --- You have to give Total text data columns. After this use the Embedding layer to get word vectors. Use given predefined glove word vectors, don't train any word vectors. After this use LSTM and get the LSTM output and Flatten that output.
- **Input_school_state** --- Give 'school_state' column as input to embedding layer and Train the Keras Embedding layer.
- **Project_grade_category** --- Give 'project_grade_category' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_categories** --- Give 'input_clean_categories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_clean_subcategories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_teacher_prefix' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_remaining_teacher_number_of_previously_posted_projects._resource_summary_contains_nun** ---concatenate remaining columns and add a Dense layer after that.

- For LSTM, you can choose your sequence padding methods on your own or you can train your LSTM without padding, there is no restriction on that.

Below is an example of embedding layer for a categorical columns. In below code all are dummy values, we gave only for referance.

In [0]:
```
# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work
input_layer = Input(shape=(n,))
embedding = Embedding(no_1, no_2, input_length=n)(input_layer)
flatten = Flatten()(embedding)
```

1. Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/)
2. Please go through this link https://keras.io/getting-started/functional-api-guide/ (https://keras.io/getting-started/functional-api-guide/) and check the 'Multi-input and multi-output models' then you will get to know how to give multiple inputs.

1.1 Dividing into X(data) and y (label)

In [5]:
```
data = pd.read_csv('/content/preprocessed_data.csv')#,nrows=1000)
# Dataset is now stored in a Pandas Dataframe
data.shape
```

Out[5]:
```
(109248, 9)
```

In [6]:
```
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)
X.head(1)
```

Out[6]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_pro
0	ca	mrs	grades_prek_2	

In [7]:
```
X.shape
```

Out[7]:
```
(109248, 8)
```

1.2 Splitting data into Train and cross validation(or test): Stratified Sampling

In [0]:
```
# 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=0.33, stra
tify=y)
#X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=
0.33, stratify=y_train)
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

1.3 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/ (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)

1.3.1 Teacher Prefix

In [0]:
```
from sklearn.preprocessing import LabelEncoder
```

In [0]:
```
# Teacher Prefix
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
no_of_unique_prefix  = X_train["teacher_prefix"].nunique()
embed_size_prefix = int(min(np.ceil((no_of_unique_prefix)/2), 50 ))

input_teacher_prefix = Input(shape=(1,),name="teacher_prefix")
embed_teacher_prefix_out = Embedding(no_of_unique_prefix,
                                     embed_size_prefix,
                                     trainable=True,
                                     name="embedded_teacher_prefix")(input_teacher_prefix)
flat_teacher_prefix_out = Flatten()(embed_teacher_prefix_out)

#LabelEncoder : https://stackoverflow.com/a/41774086
lab_enc = LabelEncoder()
X_train_teacher_prefix_label = lab_enc.fit_transform(X_train["teacher_prefix"])

X_test_teacher_prefix_label = lab_enc.transform(X_test["teacher_prefix"])
```

1.3.2 School Sate

In [0]:
```
# School State
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
no_of_unique_state  = X_train["school_state"].nunique()
embed_size_state= int(min(np.ceil((no_of_unique_state)/2), 50 ))

input_school_state = Input(shape=(1,),name="school_prefix")
embed_school_state_out = Embedding(no_of_unique_state,
                                   embed_size_state,
                                   name="embedded_school_state",
                                   trainable=True)(input_school_state)
flat_school_state_out = Flatten()(embed_school_state_out)



lab_enc = LabelEncoder()
X_train_school_state_label = lab_enc.fit_transform(X_train["school_state"])
X_test_school_state_label = lab_enc.transform(X_test["school_state"])
```

1.3.3 Project Grade Cotegory

In [0]:
```
#project_grade_category
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
no_of_unique_grade  = X_train["project_grade_category"].nunique()
embed_size_grade = int(min(np.ceil((no_of_unique_grade)/2), 50 ))

input_project_grade_category = Input(shape=(1,), name="project_grade_category")
embed_project_grade_category_out = Embedding(no_of_unique_grade,
                                             embed_size_grade,
                                             name="embedded_project_grade_category",
                                             trainable=True)(input_project_grade_category)
flat_project_grade_category_out = Flatten()(embed_project_grade_category_out)


lab_enc = LabelEncoder()
X_train_project_grade_category_label = lab_enc.fit_transform(X_train["project_grade_category"])
X_test_project_grade_category_label = lab_enc.transform(X_test["project_grade_category"])
```

1.3.4 Project Categories

In [0]:
```
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
#project_subject_categories
no_of_unique_subcat  = X_train["clean_categories"].nunique()
embed_size_subcat = int(min(np.ceil((no_of_unique_subcat)/2), 50 ))

input_categories= Input(shape=(1,),name="clean_categories")
embed_categories_out = Embedding(no_of_unique_subcat,
                                 embed_size_subcat,
                                 name="embedded_clean_categories",
                                 trainable=True)(input_categories)
flat_categories_out = Flatten()(embed_categories_out)


lab_enc = LabelEncoder()
X_train_clean_categories_label = lab_enc.fit_transform(X_train["clean_categories"])

X_test["clean_categories"] = X_test["clean_categories"].map(lambda s: ' ' if s not in lab_enc.classes_ else s)
lab_enc.classes_ = np.append(lab_enc.classes_, ' ')

X_test_clean_categories_label= lab_enc.transform(X_test["clean_categories"])
```

1.3.5 Project SubCategories

In [0]:
```
#project_subject_subcategories
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
no_of_unique_subcat_1  = X_train["clean_subcategories"].nunique()
embed_size_subcat_1 = int(min(np.ceil((no_of_unique_subcat_1)/2), 50 ))

input_subcategories = Input(shape=(1,),name="clean_subcategories")
embed_subcategories_out = Embedding(no_of_unique_subcat_1,
                                    embed_size_subcat_1,
                                    name="embedded_subcategories",
                                    trainable=True)(input_subcategories)
flat_subcategories_out = Flatten()(embed_subcategories_out)

lab_enc = LabelEncoder()
X_train_clean_subcategories_label = lab_enc.fit_transform(X_train["clean_subcategories"])
X_test["clean_subcategories"] = X_test["clean_subcategories"].map(lambda s: ' ' if s not in lab_enc.classes_ else s)
lab_enc.classes_ = np.append(lab_enc.classes_, ' ')
X_test_clean_subcategories_label= lab_enc.transform(X_test["clean_subcategories"])
```

In [211]:
```
X_train.columns
```

Out[211]:
```
Index(['school_state', 'teacher_prefix', 'project_grade_category',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'clean_subcategories', 'essay', 'price'],
      dtype='object')
```

1.4 Vectroizring Numerical Feature

1.4.1 Price and Teacher_number_of_previously_posted_projects

In [0]:
```
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
price_train=X_train['price'].values.reshape(-1,1)
price_test=X_test['price'].values.reshape(-1,1)


teacher_number_of_previously_posted_projects_train=X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)
teacher_number_of_previously_posted_projects_test=X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)

numeric_features_price_teacher_train=np.concatenate((price_train,
                                teacher_number_of_previously_posted_projects_train),
                                axis=1)


numeric_features_price_teacher_test=np.concatenate((price_test,
                                teacher_number_of_previously_posted_projects_test),
                                axis=1)
```

In [0]:
```
scalar=StandardScaler()

numeric_features_price_teacher_train_scaled=scalar.fit_transform(numeric_features_price_teacher_train)
numeric_features_price_teacher_test_scaled=scalar.transform(numeric_features_price_teacher_test)
```

In [0]:
```
numerical_out = Input(shape=(2,),name="numerical_features")
numerical_dense_out = Dense(100,
                            activation="relu",
                            kernel_initializer="he_normal",
                            kernel_regularizer=regularizers.l2(0.001))(numerical_out)
```

1.5 Tokenizning Text data Essay Vectorization of Essays

In [0]:
```
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
#tokenizer keras : https://stackoverflow.com/a/51956230
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train["essay"].tolist())
seq_train = tokenizer.texts_to_sequences(X_train["essay"].values)
seq_test = tokenizer.texts_to_sequences(X_test["essay"].values)
```

In [0]:
```
from keras.preprocessing.sequence import pad_sequences
```

In [0]:
```
#this is done to make the input to the fisrt layer same length
padded_train = pad_sequences(seq_train,maxlen=300,padding='post', truncating=
'post')
padded_test = pad_sequences(seq_test, maxlen=300,padding='post', truncating='p
ost')
```

In [0]:
```
import pickle
with open('glove_vectors', 'rb') as f:
    glove = pickle.load(f)
    glove_words =  set(glove.keys())
```

In [0]:
```
#https://learn-neural-networks.com/world-embedding-by-keras/

vocab_size = len(tokenizer.word_index) + 1

 #if integer data is encoded with values
# from 0 to 10, then the size of the dictionary will be 11 words.
# create a weight matrix for words in training docs
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-ke
ras/

#max_vocabulary_length = len(tokenizer.word_index)

embedding_matrix = np.zeros((vocab_size, 300))

for word, i in tokenizer.word_index.items():

    if word in glove_words:
      embedding_vector = glove[word]
      embedding_matrix[i] = embedding_vector
```

1.6 DEEP LEARINNG MODEL ON VECTORED FEATURES

1.6.1 MODEL_1 FUNCTIONAL API

In [0]:
```
#flatten 1 of embbeded text
essay = Input(shape=(300,))
X = Embedding(output_dim=300,
              input_dim=vocab_size,
              input_length=len(padded_train[0]) ,
              weights=[embedding_matrix])(essay)
lstm_essay = LSTM(100,
                  recurrent_dropout=0.5,
                  kernel_regularizer=regularizers.l2(0.001),
                  return_sequences=True)(X)
flatten_1_essay = Flatten()(lstm_essay)
```

In [0]:
```
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
concat_out = concatenate([flatten_1_essay,
                          flat_teacher_prefix_out,
                          flat_school_state_out ,
                          flat_project_grade_category_out,
                          flat_categories_out,
                          flat_subcategories_out,
                          numerical_dense_out],axis=-1)

x = Dense(128,activation="relu", kernel_initializer="he_normal",
                       kernel_regularizer=regularizers.l2(0.001))(concat_out)

x1 = Dropout(0.5)(x)

x2 = Dense(256,activation="relu",kernel_initializer="he_normal",
                       kernel_regularizer=regularizers.l2(0.001))(x1)

x3 = Dropout(0.5)(x2)

x4 = Dense(64,activation="relu", kernel_initializer="he_normal",
                      kernel_regularizer=regularizers.l2(0.001))(x3)

x5 = BatchNormalization()(x4)

output = Dense(2, activation='softmax', name='output')(x5)
model_1 = Model(inputs=[essay,
                        input_teacher_prefix,
                        input_school_state,
                        input_project_grade_category,
                        input_categories,
                        input_subcategories,
                        numerical_out],outputs=[output])
```

1.6.2 MODEL_1 ARCHITECTURE

In [26]:
```
# summarize the model
from tensorflow.keras.utils import plot_model

plot_model(model_1, 'model_1.png', show_shapes=True)
```

Out[26]:

1.6.3 MODEL_1 SUMMARY

In [27]:
```
print(model_1.summary())
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 300)          0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 300, 300)     14445000    input_1[0]
[0]
__________________________________________________________________________________________________
teacher_prefix (InputLayer)     (None, 1)            0
__________________________________________________________________________________________________
school_prefix (InputLayer)      (None, 1)            0
__________________________________________________________________________________________________
project_grade_category (InputLa (None, 1)            0
__________________________________________________________________________________________________
clean_categories (InputLayer)   (None, 1)            0
__________________________________________________________________________________________________
clean_subcategories (InputLayer (None, 1)            0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 300, 100)     160400      embedding_1
[0][0]
__________________________________________________________________________________________________
embedded_teacher_prefix (Embedd (None, 1, 3)         15          teacher_pref
ix[0][0]
__________________________________________________________________________________________________
embedded_school_state (Embeddin (None, 1, 26)        1326        school_prefi
x[0][0]
__________________________________________________________________________________________________
embedded_project_grade_category (None, 1, 2)         8           project_grad
e_category[0][0]
__________________________________________________________________________________________________
embedded_clean_categories (Embe (None, 1, 26)        1326        clean_catego
ries[0][0]
__________________________________________________________________________________________________
embedded_subcategories (Embeddi (None, 1, 50)        19500       clean_subcat
egories[0][0]
__________________________________________________________________________________________________
numerical_features (InputLayer) (None, 2)            0
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 30000)        0           lstm_1[0][0]
__________________________________________________________________________________________________
```

```
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 3)            0           embedded_tea
cher_prefix[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 26)           0           embedded_sch
ool_state[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 2)            0           embedded_pro
ject_grade_category[0
__________________________________________________________________________________________________
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 26)           0           embedded_cle
an_categories[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 50)           0           embedded_sub
categories[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 100)          300         numerical_fe
atures[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 30207)        0           flatten_6[0]
[0]
                                                                 flatten_1[0]
[0]
                                                                 flatten_2[0]
[0]
                                                                 flatten_3[0]
[0]
                                                                 flatten_4[0]
[0]
                                                                 flatten_5[0]
[0]
                                                                 dense_1[0]
[0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          3866624     concatenate_
1[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           dense_2[0]
[0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 256)          33024       dropout_1[0]
[0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 256)          0           dense_3[0]
[0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
```

```
dense_4 (Dense)                 (None, 64)           16448       dropout_2[0]
[0]
__________________________________________________________________________________
_____________________
batch_normalization_1 (BatchNor (None, 64)           256         dense_4[0]
[0]
__________________________________________________________________________________
_____________________
output (Dense)                  (None, 2)            130         batch_normal
ization_1[0][0]
==================================================================================
=====================
Total params: 18,544,357
Trainable params: 18,544,229
Non-trainable params: 128
__________________________________________________________________________________
_____________________
None
```

1.6.4 MODEL_1 COMPLILATION AND TESTING

In [0]:
```
# https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-roc-and-auc-in-keras
#https://github.com/sahildigikar15/LSTM-on-Donors-Choose-Dataset/blob/master/LSTM_donors_choose.ipynb

def auc1(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score(y_true, y_pred)

def auroc(y_true, y_pred):
    return tf.py_func(auc1, (y_true, y_pred), tf.double)
```

In [0]:
```
X_train_final_1 = [padded_train,
                   X_train_teacher_prefix_label,
                   X_train_school_state_label,
                   X_train_project_grade_category_label,
                   X_train_clean_categories_label,
                   X_train_clean_subcategories_label,
                   numeric_features_price_teacher_train_scaled]


X_test_final_1 = [padded_test,
                   X_test_teacher_prefix_label,
                   X_test_school_state_label,
                   X_test_project_grade_category_label,
                   X_test_clean_categories_label,
                   X_test_clean_subcategories_label,
                   numeric_features_price_teacher_test_scaled]
```

In [0]:
```
Y_train = np_utils.to_categorical(y_train, 2)
Y_test = np_utils.to_categorical(y_test, 2)
```

In [0]:
```
#y_train[1000:1500]
```

In [0]:
```
#Y_train[1000:1500]
```

In [0]:
```
from keras.callbacks import EarlyStopping,TensorBoard

checkpoint_1 = ModelCheckpoint("model_1.h5",
                               monitor="val_auroc",
                               mode="max",
                               save_best_only = True,
                               verbose=1)
early_stopping = EarlyStopping(monitor='val_auroc',
                           min_delta=0,
                           patience=2,
                           verbose=2, mode='auto')

NAME = 'model_1'
tensorboard_callback =  TensorBoard(log_dir="logs",histogram_freq=1)
callbacks_1 = [tensorboard_callback,checkpoint_1,early_stopping]
```

In [0]:
```
from sklearn.metrics import roc_auc_score
```

In [0]:
```
model_1.compile(optimizer=Adam(lr=0.001),
                loss='categorical_crossentropy',
                metrics=[auroc])
```

In [0]:
```
history_1 = model_1.fit(X_train_final_1,
                        y_train,batch_size=512,
                        epochs=15,
                        validation_data=(X_test_final_1,Y_test),
                        verbose=10,callbacks=callbacks_1)
```

```
Train on 73196 samples, validate on 36052 samples
Epoch 1/15

Epoch 00001: val_auroc improved from -inf to 0.70070, saving model to model_
1.h5
Epoch 2/15

Epoch 00002: val_auroc improved from 0.70070 to 0.74721, saving model to mode
l_1.h5
Epoch 3/15

Epoch 00003: val_auroc improved from 0.74721 to 0.75380, saving model to mode
l_1.h5
Epoch 00003: early stopping
```

In [0]:
```
print(history_1.history)
```

```
{'val_loss': [0.915327158088519, 0.6953587072030244, 0.5973227125193982], 'va
l_auroc': [0.7006973624229431, 0.7472118735313416, 0.753804087638855], 'los
s': [1.4246434963715586, 0.7821747613113443, 0.6174498673631648], 'auroc':
[0.54676956, 0.7137192, 0.76374865]}
```

In [0]:
```
model_1_perfo=pd.DataFrame(history_1.history)
```

In [0]:
```
model_1_perfo
```

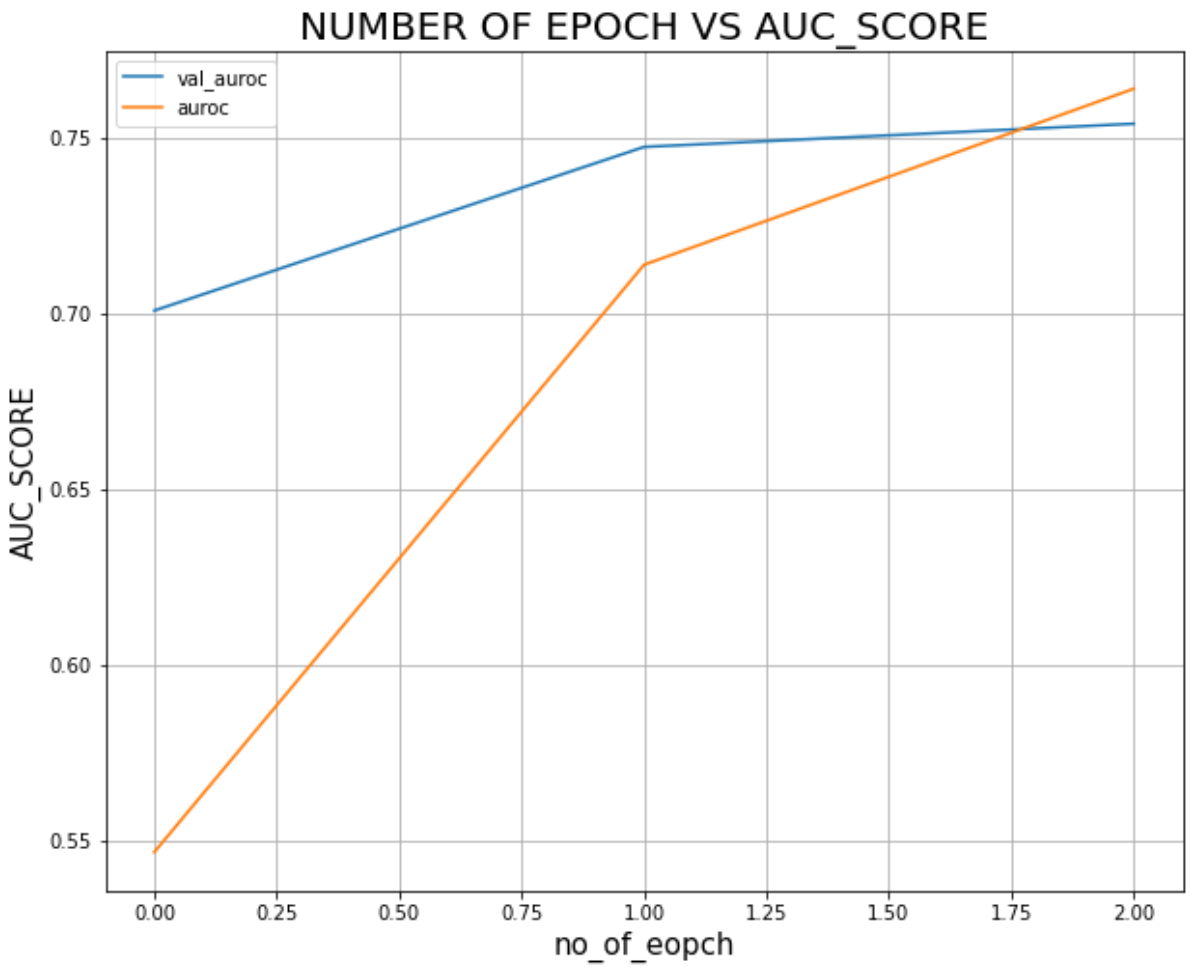
Out[0]:

	val_loss	val_auroc	loss	auroc
0	0.915327	0.700697	1.424643	0.546770
1	0.695359	0.747212	0.782175	0.713719
2	0.597323	0.753804	0.617450	0.763749

In [0]:
```
model_1_auc=model_1_perfo.drop(['val_loss', 'loss'], axis=1)
```

In [0]:
```
plt.figure()
ax=model_1_auc.plot(figsize=(10, 8))
ax.set_ylabel('AUC_SCORE',fontsize=15)
ax.set_xlabel('no_of_eopch',fontsize=15)
ax.set_title('NUMBER OF EPOCH VS AUC_SCORE',fontsize=20)
plt.grid()
```

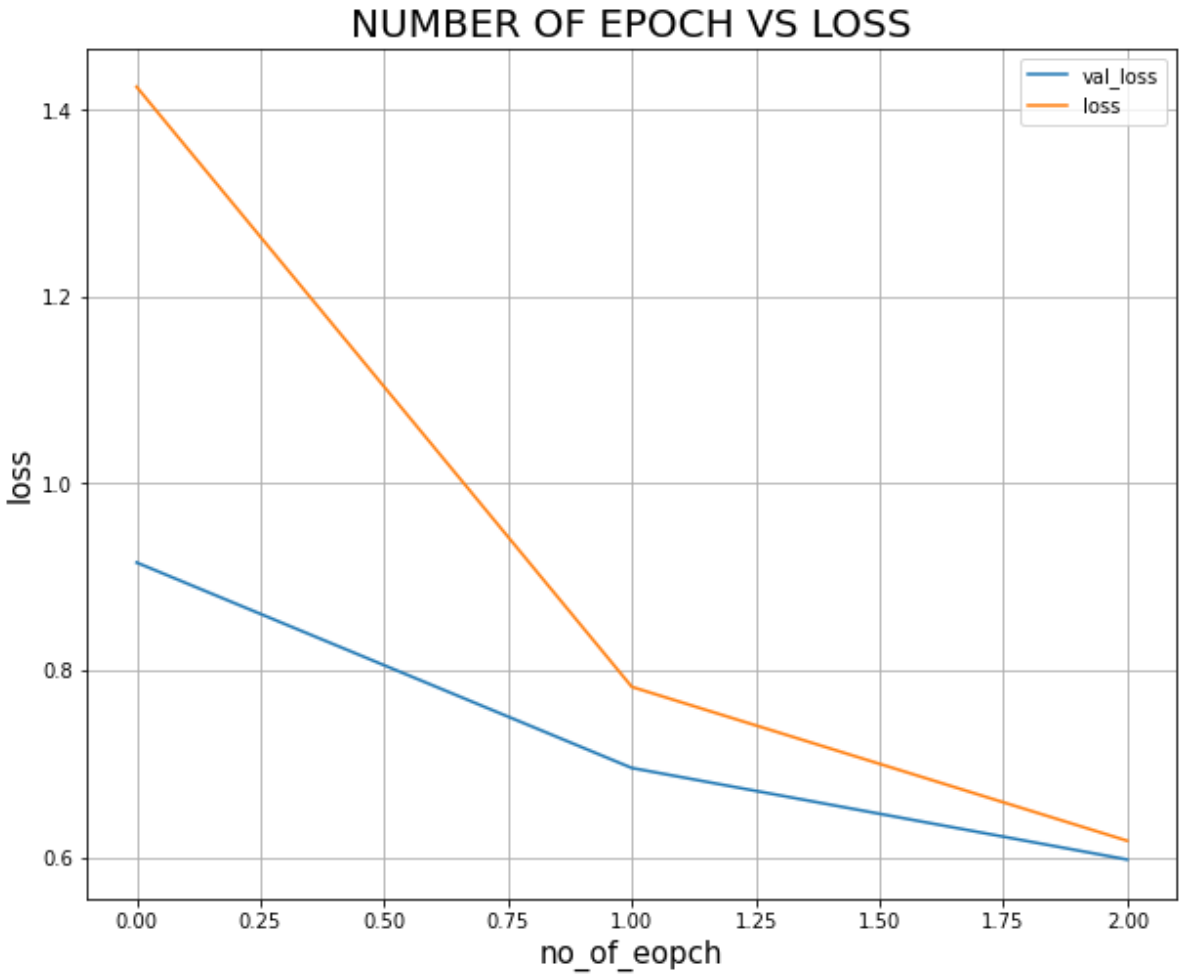
```
<Figure size 432x288 with 0 Axes>
```



In [0]:
```
model_1_loss=model_1_perfo.drop(['val_auroc', 'auroc'], axis=1)
```

In [0]:
```
plt.figure()
ax=model_1_loss.plot(figsize=(10, 8))
ax.set_ylabel('loss',fontsize=15)
ax.set_xlabel('no_of_eopch',fontsize=15)
ax.set_title('NUMBER OF EPOCH VS LOSS',fontsize=20)
plt.grid()
```

```
<Figure size 432x288 with 0 Axes>
```



1.6.5 MODEL 1 TRAIN AND TEST AUC

In [0]:
```
y_train_pred = model_1.predict(X_train_final_1)
print("Train AUC:",roc_auc_score(Y_train,y_train_pred))

y_test_pred = model_1.predict(X_test_final_1)
print("Test AUC:",roc_auc_score(Y_test,y_test_pred))
```

```
Train AUC: 0.8042405606100359
Test AUC: 0.7547182056302763
```

Model-2

Use the same model as above but for 'input_seq_total_text_data' give only some words in the sentance not all the words. Filter the words as below.

1. Train the TF-IDF on the Train data feature 'essay'

2. Get the idf value for each word we have in the train data.

3. Remove the low idf value and high idf value words from our data. Do some analysis on the Idf values and based on those values choose the low and high threshold value. Because very frequent words and very very rare words don't give much information. (you can plot a box plots and take only the idf scores within IQR range and corresponding words)

4. Train the LSTM after removing the Low and High idf value words. (In model-1 Train on total data but in Model-2 train on data after removing some words based on IDF values)

2.1 idf values of Essay text using tfidf vectorizer

In [0]:
```
vectorizer = TfidfVectorizer()
tfidf = vectorizer.fit_transform(X_train["essay"].values)
```

In [170]:
```
print("="*70)
print("words 5000 to 5005:\n",vectorizer.get_feature_names()[5000:5005])
print("-"*70)
print("idf values of 5000 to 5005:\n",vectorizer.idf_[5000:5005])
print("="*70)
```

```
======================================================================
words 5000 to 5005:
 ['bellingham', 'bellow', 'bellringer', 'bellringers', 'bells']
----------------------------------------------------------------------
idf values of 5000 to 5005:
 [11.50776253 11.10229743 11.50776253 11.10229743  7.75825846]
======================================================================
```

In [0]:
```
data = {'word': vectorizer.get_feature_names() , 'idf_value': vectorizer.idf_}
tfidf_df = pd.DataFrame(data=data)
```

In [172]:
```
tfidf_df.head(10)
```

Out[172]:

	word	idf_value
0	00	7.170472
1	000	5.918643
2	001	11.102297
3	002	11.507763
4	003	11.507763
5	005nannan	11.507763
6	00am	10.121468
7	00pm	9.716003
8	01	11.102297
9	01075rm	11.507763

In [0]:
```
tfidf_df = tfidf_df.sort_values(by ='idf_value' )
```

In [174]:
```
tfidf_df.head(5)
```

Out[174]:

	word	idf_value
41110	students	1.007652
28552	nannan	1.045202
37437	school	1.159333
28455	my	1.246566
24690	learning	1.364471

In [175]:
```
tfidf_df.tail(2)
```

Out[175]:

	word	idf_value
34160	qha	11.507763
48119	zzzzzzz	11.507763

In [176]:
```
tfidf_df[tfidf_df['word']=='eukaryotic']
```

Out[176]:

	word	idf_value
15364	eukaryotic	11.507763

In [177]:
```
tfidf_df[tfidf_df['word']=='00pm']
```

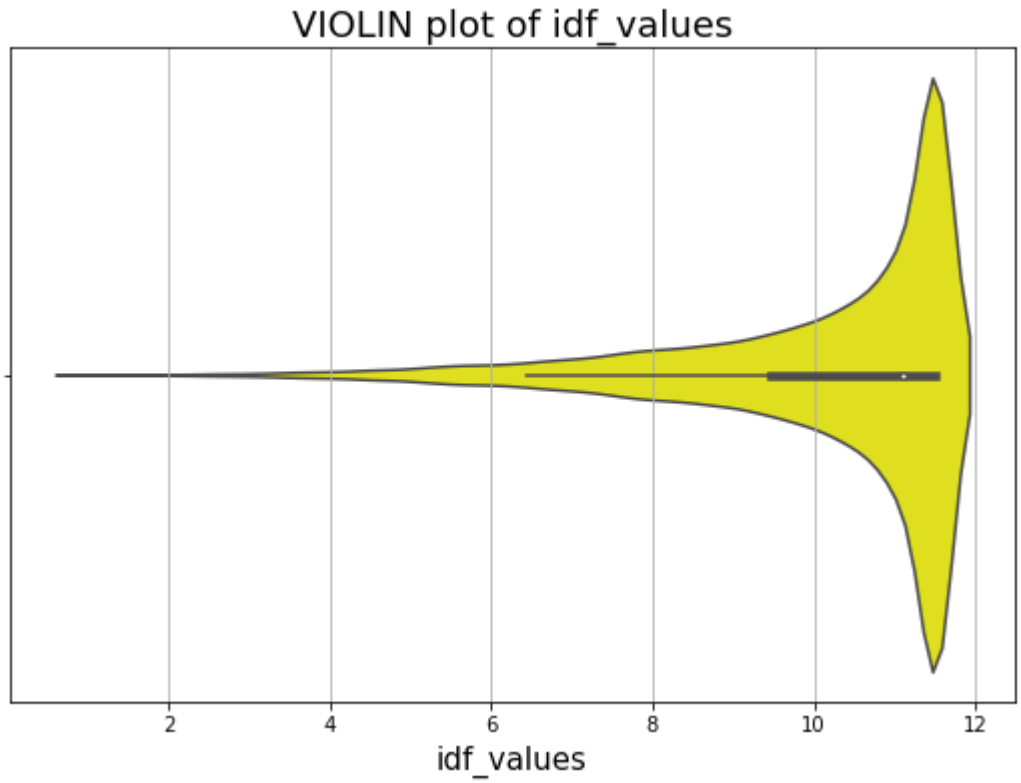
Out[177]:

	word	idf_value
7	00pm	9.716003

2.1.1 Violin plot of idf_values

In [178]:
```
plt.figure(figsize=(9,6))
plt.grid()
sns.violinplot(x = "idf_value",data=tfidf_df,orient="h",color='yellow')
plt.xlabel("idf_values",fontsize=15)
plt.title("VIOLIN plot of idf_values",fontsize=18)
```

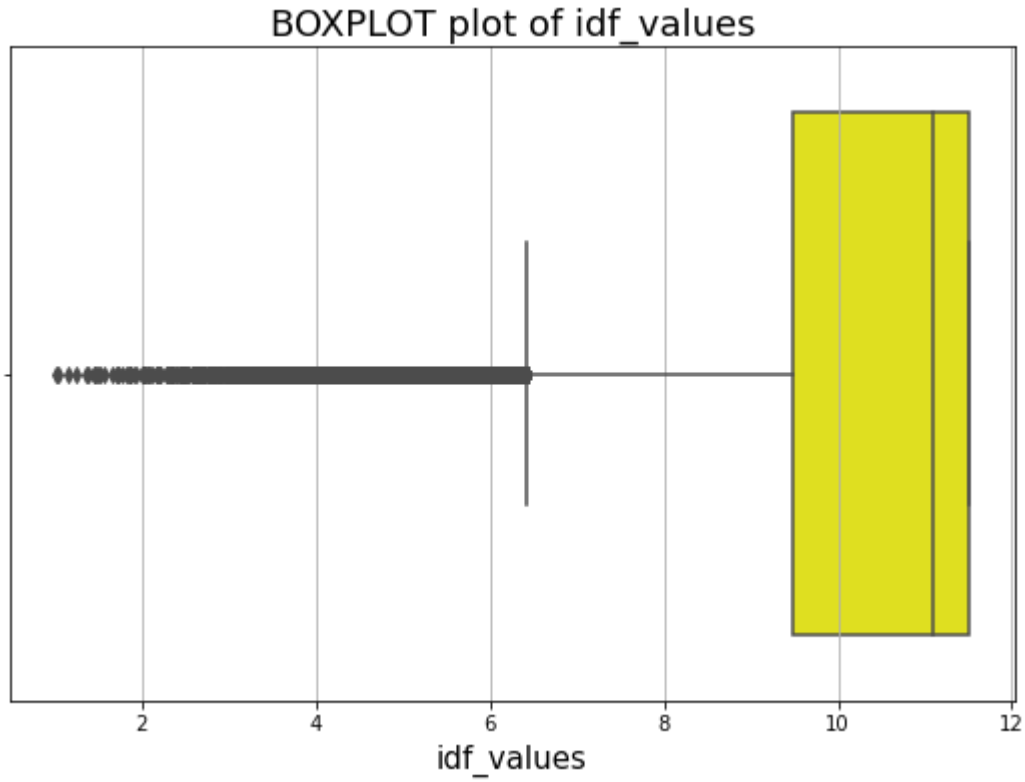
Out[178]:
```
Text(0.5, 1.0, 'VIOLIN plot of idf_values')
```



2.1.2 Box plot of idf_values

In [179]:
```
plt.figure(figsize=(9,6))
plt.grid()
sns.boxplot(x = "idf_value",data=tfidf_df,orient="h",color='yellow')
plt.xlabel("idf_values",fontsize=15)
plt.title("BOXPLOT plot of idf_values",fontsize=18)
```

Out[179]:
```
Text(0.5, 1.0, 'BOXPLOT plot of idf_values')
```



In [180]:
```
tfidf_df.shape
```

Out[180]:
```
(48120, 2)
```

In [0]:
```
#tfidf_df['idf_value'].plot(kind="bar")
```

In [182]:
```
np.arange(0, 100, 5)
```

Out[182]:
```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80,
       85, 90, 95])
```

In [183]:
```
print(np.percentile(tfidf_df['idf_value'],np.arange(0, 100, 5)))
```

```
[ 1.00765247  6.20445763  7.4735219   8.26908408  8.94281318  9.47672488
  9.89832462 10.25499957 10.5914718  10.81461535 11.10229743 11.10229743
 11.10229743 11.50776253 11.50776253 11.50776253 11.50776253 11.50776253
 11.50776253 11.50776253]
```

In [0]:
```
dict_for_percentiles={'idf_values_q':[ 1.00762494,  6.20346384 , 7.4735219 ,
8.26908408 ,
                                       8.94281318  ,9.49285951,9.89832462 ,10.2
5499957,
                                       10.5914718 , 10.81461535, 11.10229743, 1
1.10229743,
                                       11.10229743, 11.50776253, 11.50776253 ,1
1.50776253 ,
                                       11.50776253, 11.50776253,11.50776253 ,1
1.50776253],
 'percentiles':[ 0,  5, 10, 15, 20,
                25, 30, 35, 40, 45,
                50, 55, 60, 65, 70,
                75, 80,85, 90, 95]}
```

In [0]:
```
df_quantiles=pd.DataFrame(data=dict_for_percentiles)
```

In [186]:
```
df_quantiles
```

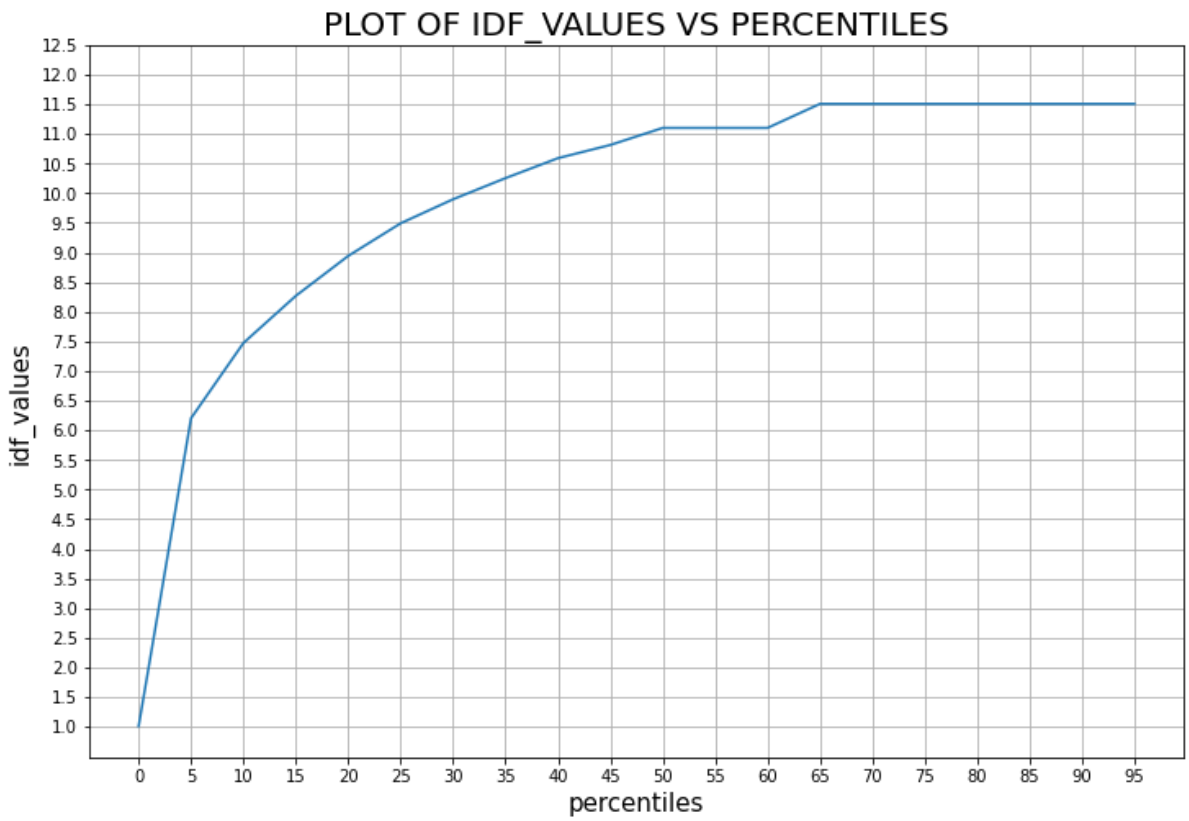
Out[186]:

	idf_values_q	percentiles
0	1.007625	0
1	6.203464	5
2	7.473522	10
3	8.269084	15
4	8.942813	20
5	9.492860	25
6	9.898325	30
7	10.255000	35
8	10.591472	40
9	10.814615	45
10	11.102297	50
11	11.102297	55
12	11.102297	60
13	11.507763	65
14	11.507763	70
15	11.507763	75
16	11.507763	80
17	11.507763	85
18	11.507763	90
19	11.507763	95

2.1.3 PLOT OF IDF_VALUES VS PERCENTILES

In [187]:
```
plt.figure(figsize=(12,8))
plt.grid()

plt.plot(df_quantiles['percentiles'], df_quantiles['idf_values_q'],)
plt.ylabel('idf_values',fontsize=15)
plt.xlabel('percentiles',fontsize=15)
plt.xticks(df_quantiles['percentiles'])
plt.yticks(np.arange(1,13,0.5))

plt.title('PLOT OF IDF_VALUES VS PERCENTILES',fontsize=20)
```

Out[187]: Text(0.5, 1.0, 'PLOT OF IDF_VALUES VS PERCENTILES')



we'll consider 4th poercentiles to 45th percetnilies values by observing graph

In [188]:
```
print("4th percentiles idf values:",np.percentile(tfidf_df['idf_value'],4))
print("45th percentiles idf values:",np.percentile(tfidf_df['idf_value'],45))
```

```
4th percentiles idf values: 5.821669971750325
50th percentiles idf values: 10.814615354383104
```

In [189]:
```
print("shape of tfidf_df: ",tfidf_df.shape)

tfidf_best = tfidf_df[(tfidf_df['idf_value']>=5.816525443279854) & (tfidf_df[
'idf_value'] <=10.814615354383104)]

print("shape of tfidf_best :",tfidf_best.shape)
```

```
shape of tfidf_df:  (48120, 2)
shape of tfidf_best : (22062, 2)
```

In [190]:
```
tfidf_best.head()
```

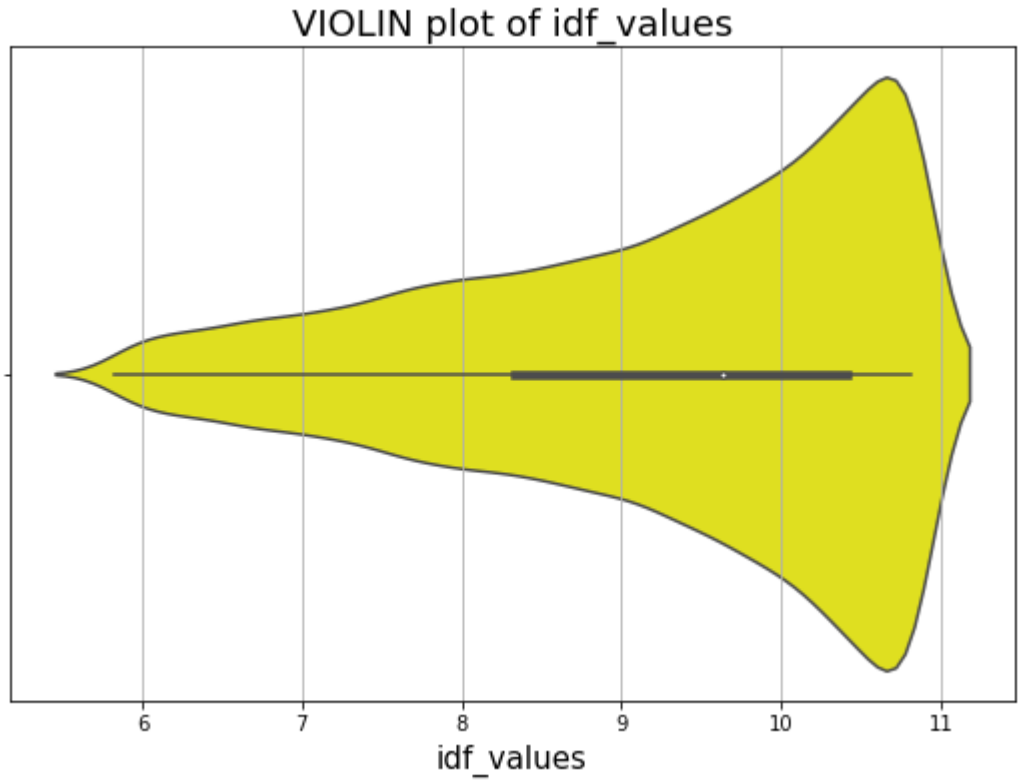
Out[190]:

	word	idf_value
38682	significantly	5.819094
36439	risks	5.819094
33477	processing	5.819094
12938	dive	5.822484
40062	sponges	5.824183

2.1.4 Violin Plot after rejecting low and high idf_values

In [191]:
```
plt.figure(figsize=(9,6))
plt.grid()
sns.violinplot(x = "idf_value",data=tfidf_best,orient="h",color='yellow')
plt.xlabel("idf_values",fontsize=15)
plt.title("VIOLIN plot of idf_values",fontsize=18)
```

Out[191]:
```
Text(0.5, 1.0, 'VIOLIN plot of idf_values')
```



2.2 Tokening selected essays using selected words based on idf_values

In [0]:
```
array_best_words=tfidf_best['word'].values
```

In [193]:
```
array_best_words
```

Out[193]:
```
array(['significantly', 'risks', 'processing', ..., 'begets',
       'infomercial', 'infrastructures'], dtype=object)
```

In [0]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(array_best_words)
X_train_tfidf = tokenizer.texts_to_sequences(X_train["essay"])
X_test_tfidf = tokenizer.texts_to_sequences(X_test["essay"])
```

In [195]:
```
len(tokenizer.word_index)
```

Out[195]:
```
22062
```

In [0]:
```
vocab_size = len(tokenizer.word_index) + 1
```

In [0]:
```
X_train_tfidf_padded = pad_sequences(X_train_tfidf,
                                     maxlen=800,
                                     padding='post',
                                     truncating='post')

X_test_tfidf_padded = pad_sequences(X_test_tfidf,
                                    maxlen=800,
                                    padding='post',
                                    truncating='post')
```

In [0]:
```
#https://learn-neural-networks.com/world-embedding-by-keras/

vocab_size = len(tokenizer.word_index) + 1
embedding_matrix = np.zeros((vocab_size, 300))

for word, i in tokenizer.word_index.items():

    if word in glove_words:
      embedding_vector = glove[word]
      embedding_matrix[i] = embedding_vector
```

In [199]:
```
tf.keras.backend.clear_session
```

Out[199]:
```
<function tensorflow.python.keras.backend.clear_session>
```

2.3 LSTM embedding and lstm layer on essyas text

In [0]:
```
input_text_essay_tfidf = Input(shape=(800,),name="input_text_essay_tfidf")

embedding_layer = Embedding(vocab_size,
                            300,
                            weights=[embedding_matrix],
                            input_length=800,
                            trainable=False)(input_text_essay_tfidf)

lstm_out = LSTM(128,
                recurrent_dropout=0.5,
                kernel_regularizer=regularizers.l2(0.001),
                return_sequences=True)(embedding_layer)
flattten_essay_tfidf= Flatten()(lstm_out)
```

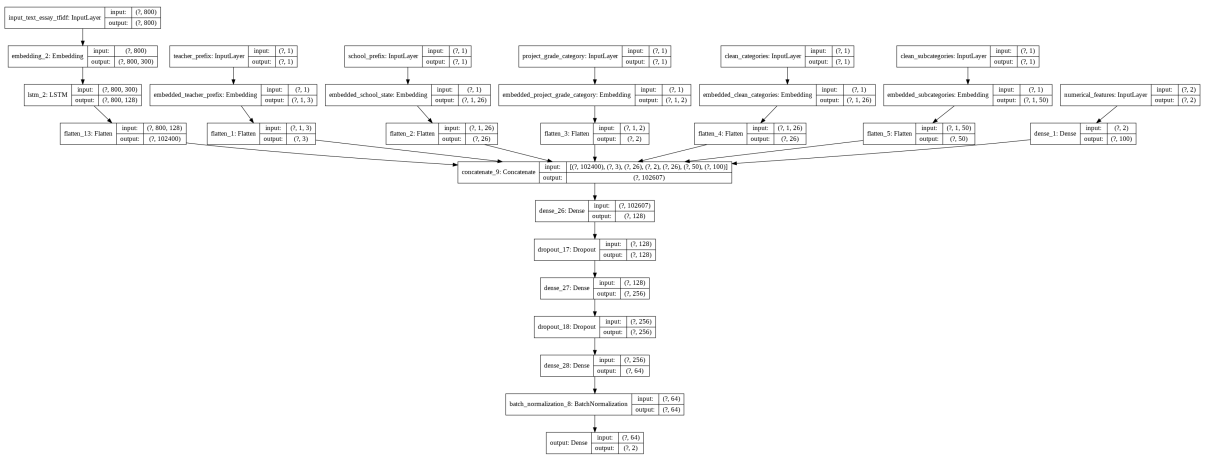
2.4 Model_2 Deep learing model

In [0]:
```
concat_out_2 = concatenate([flattten_essay_tfidf,
                            flat_teacher_prefix_out,
                            flat_school_state_out ,
                            flat_project_grade_category_out,
                            flat_categories_out,
                            flat_subcategories_out,
                            numerical_dense_out],axis=-1)

x_model_2 = Dense(128,activation="relu", kernel_initializer="he_normal",
                        kernel_regularizer=regularizers.l2(0.001))(concat_out_2)

x1_model_2 = Dropout(0.5)(x_model_2)

x2_model_2 = Dense(256,activation="relu",kernel_initializer="he_normal",
                        kernel_regularizer=regularizers.l2(0.001))(x1_model_2)

x3_model_2 = Dropout(0.5)(x2_model_2)

x4_model_2 = Dense(64,activation="relu", kernel_initializer="he_normal",
                       kernel_regularizer=regularizers.l2(0.001))(x3_model_2)

x5_model_2 = BatchNormalization()(x4_model_2)

output_2 = Dense(2, activation='softmax', name='output')(x5_model_2)
model_2 = Model(inputs=[input_text_essay_tfidf,
                        input_teacher_prefix,
                        input_school_state,
                        input_project_grade_category,
                        input_categories,
                        input_subcategories,
                        numerical_out],outputs=[output_2])
```

2.5 Model Architecture and Summary

In [204]:
```
# summarize the model
from tensorflow.keras.utils import plot_model

plot_model(model_2, 'model_2.png', show_shapes=True)
```

Out[204]:



In [205]:
```
print(model_2.summary())
```

```
Model: "model_7"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_text_essay_tfidf (InputLa (None, 800)          0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 800, 300)     6618900     input_text_e
ssay_tfidf[0][0]
__________________________________________________________________________________________________
teacher_prefix (InputLayer)     (None, 1)            0
__________________________________________________________________________________________________
school_prefix (InputLayer)      (None, 1)            0
__________________________________________________________________________________________________
project_grade_category (InputLa (None, 1)            0
__________________________________________________________________________________________________
clean_categories (InputLayer)   (None, 1)            0
__________________________________________________________________________________________________
clean_subcategories (InputLayer (None, 1)            0
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 800, 128)     219648      embedding_2
[0][0]
__________________________________________________________________________________________________
embedded_teacher_prefix (Embedd (None, 1, 3)         15          teacher_pref
ix[0][0]
__________________________________________________________________________________________________
embedded_school_state (Embeddin (None, 1, 26)        1326        school_prefi
x[0][0]
__________________________________________________________________________________________________
embedded_project_grade_category (None, 1, 2)         8           project_grad
e_category[0][0]
__________________________________________________________________________________________________
embedded_clean_categories (Embe (None, 1, 26)        1326        clean_catego
ries[0][0]
__________________________________________________________________________________________________
embedded_subcategories (Embeddi (None, 1, 50)        19700       clean_subcat
egories[0][0]
__________________________________________________________________________________________________
numerical_features (InputLayer) (None, 2)            0
__________________________________________________________________________________________________
flatten_13 (Flatten)            (None, 102400)       0           lstm_2[0][0]
__________________________________________________________________________________________________
```

```
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 3)            0           embedded_tea
cher_prefix[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 26)           0           embedded_sch
ool_state[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 2)            0           embedded_pro
ject_grade_category[0
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 26)           0           embedded_cle
an_categories[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 50)           0           embedded_sub
categories[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 100)          300         numerical_fe
atures[0][0]
__________________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, 102607)       0           flatten_13
[0][0]
                                                                 flatten_1[0]
[0]
                                                                 flatten_2[0]
[0]
                                                                 flatten_3[0]
[0]
                                                                 flatten_4[0]
[0]
                                                                 flatten_5[0]
[0]
                                                                 dense_1[0]
[0]
__________________________________________________________________________________________________
dense_26 (Dense)                (None, 128)          13133824    concatenate_
9[0][0]
__________________________________________________________________________________________________
dropout_17 (Dropout)            (None, 128)          0           dense_26[0]
[0]
__________________________________________________________________________________________________
dense_27 (Dense)                (None, 256)          33024       dropout_17
[0][0]
__________________________________________________________________________________________________
dropout_18 (Dropout)            (None, 256)          0           dense_27[0]
[0]
__________________________________________________________________________________________________
```

```
dense_28 (Dense)                (None, 64)           16448       dropout_18
[0][0]
__________________________________________________________________________
_____________________
batch_normalization_8 (BatchNor (None, 64)           256         dense_28[0]
[0]
__________________________________________________________________________
_____________________
output (Dense)                  (None, 2)            130         batch_normal
ization_8[0][0]
==========================================================================
======================
Total params: 20,044,905
Trainable params: 13,425,877
Non-trainable params: 6,619,028
__________________________________________________________________________
_____________________
None
```

2.6 Model_2 Compliation and Testing

In [0]:
```
X_train_final_2 = [X_train_tfidf_padded,
                   X_train_teacher_prefix_label,
                   X_train_school_state_label,
                   X_train_project_grade_category_label,
                   X_train_clean_categories_label,
                   X_train_clean_subcategories_label,
                   numeric_features_price_teacher_train_scaled]


X_test_final_2 = [X_test_tfidf_padded,
                   X_test_teacher_prefix_label,
                   X_test_school_state_label,
                   X_test_project_grade_category_label,
                   X_test_clean_categories_label,
                   X_test_clean_subcategories_label,
                   numeric_features_price_teacher_test_scaled]
```

In [0]:
```
# https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-roc-and-auc-in-keras
#https://github.com/sahildigikar15/LSTM-on-Donors-Choose-Dataset/blob/master/LSTM_donors_choose.ipynb
def auc1(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score(y_true, y_pred)

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

In [218]:
```
from keras.callbacks import EarlyStopping,TensorBoard
import warnings
warnings.filterwarnings('ignore')

checkpoint_1 = ModelCheckpoint("model_2.h5",
                               monitor="val_auroc",
                               mode="max",
                               save_best_only = True,
                               verbose=10)
early_stopping = EarlyStopping(monitor='val_auroc',
                           min_delta=0,
                           patience=10,
                           verbose=10, mode='auto')

NAME = 'model_2'
# tensor-board in colab
# Refer: https://www.tensorflow.org/tensorboard/get_started
import os
import datetime

! rm -rf ./logs/
logdir = os.path.join("logs2", datetime.datetime.now().strftime("%Y%m%d-%H%M%
S"))
print(logdir)

tensorboard_callback =  TensorBoard(log_dir=logdir,histogram_freq=1)
callbacks_1 = [tensorboard_callback,checkpoint_1,early_stopping]

model_2.compile(optimizer=Adam(lr=0.001),
                loss='categorical_crossentropy',
                metrics=[auroc])
```

```
logs2/20200616-143937
```

In [219]:
```
%load_ext tensorboard
%tensorboard --logdir $logdir
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [220]:
```
history_2 = model_2.fit(X_train_final_2,
                        Y_train,batch_size=512,
                        epochs=15,
                        validation_data=(X_test_final_2,Y_test),
                        verbose=10,callbacks=callbacks_1)
```

```
Train on 73196 samples, validate on 36052 samples
Epoch 1/15

Epoch 00001: val_auroc improved from -inf to 0.64716, saving model to model_
2.h5
Epoch 2/15

Epoch 00002: val_auroc improved from 0.64716 to 0.66040, saving model to mode
l_2.h5
Epoch 3/15

Epoch 00003: val_auroc improved from 0.66040 to 0.66744, saving model to mode
l_2.h5
Epoch 4/15

Epoch 00004: val_auroc did not improve from 0.66744
Epoch 5/15

Epoch 00005: val_auroc improved from 0.66744 to 0.67076, saving model to mode
l_2.h5
Epoch 6/15

Epoch 00006: val_auroc did not improve from 0.67076
Epoch 7/15

Epoch 00007: val_auroc improved from 0.67076 to 0.67220, saving model to mode
l_2.h5
Epoch 8/15

Epoch 00008: val_auroc did not improve from 0.67220
Epoch 9/15

Epoch 00009: val_auroc improved from 0.67220 to 0.67321, saving model to mode
l_2.h5
Epoch 10/15

Epoch 00010: val_auroc did not improve from 0.67321
Epoch 11/15

Epoch 00011: val_auroc did not improve from 0.67321
Epoch 00011: early stopping
```

In [221]:
```
print(history_2.history)
```

```
{'val_loss': [0.9064567181599281, 0.6945099279021603, 0.6005317293339798, 0.5
479587422390169, 0.5070740595008588, 0.48795664096849417, 0.4690746718501589
4, 0.4589037619234388, 0.4471896375361762, 0.4481935477092768, 0.437567940770
64726], 'val_auroc': [0.6471607089042664, 0.6604022979736328, 0.6674440503120
422, 0.6636821627616882, 0.6707605719566345, 0.6656519770622253, 0.6722006201
74408, 0.6720241904258728, 0.6732096076011658, 0.6672064065933228, 0.66899132
72857666], 'loss': [1.35221007001342, 0.7901845625648277, 0.6453928637422495,
0.5750804884465407, 0.5277135420255344, 0.4999368268483662, 0.479040620800797
23, 0.46303043257207116, 0.45317486150517894, 0.44415762559287086, 0.43858991
12171681], 'auroc': [0.55929095, 0.6315364, 0.65248054, 0.65609926, 0.663503,
0.6622636, 0.664473, 0.6673788, 0.669908, 0.6708307, 0.6720362]}
```

In [0]:
```
model_2_perfo=pd.DataFrame(history_2.history)
```

In [223]:
```
model_2_perfo
```

Out[223]:

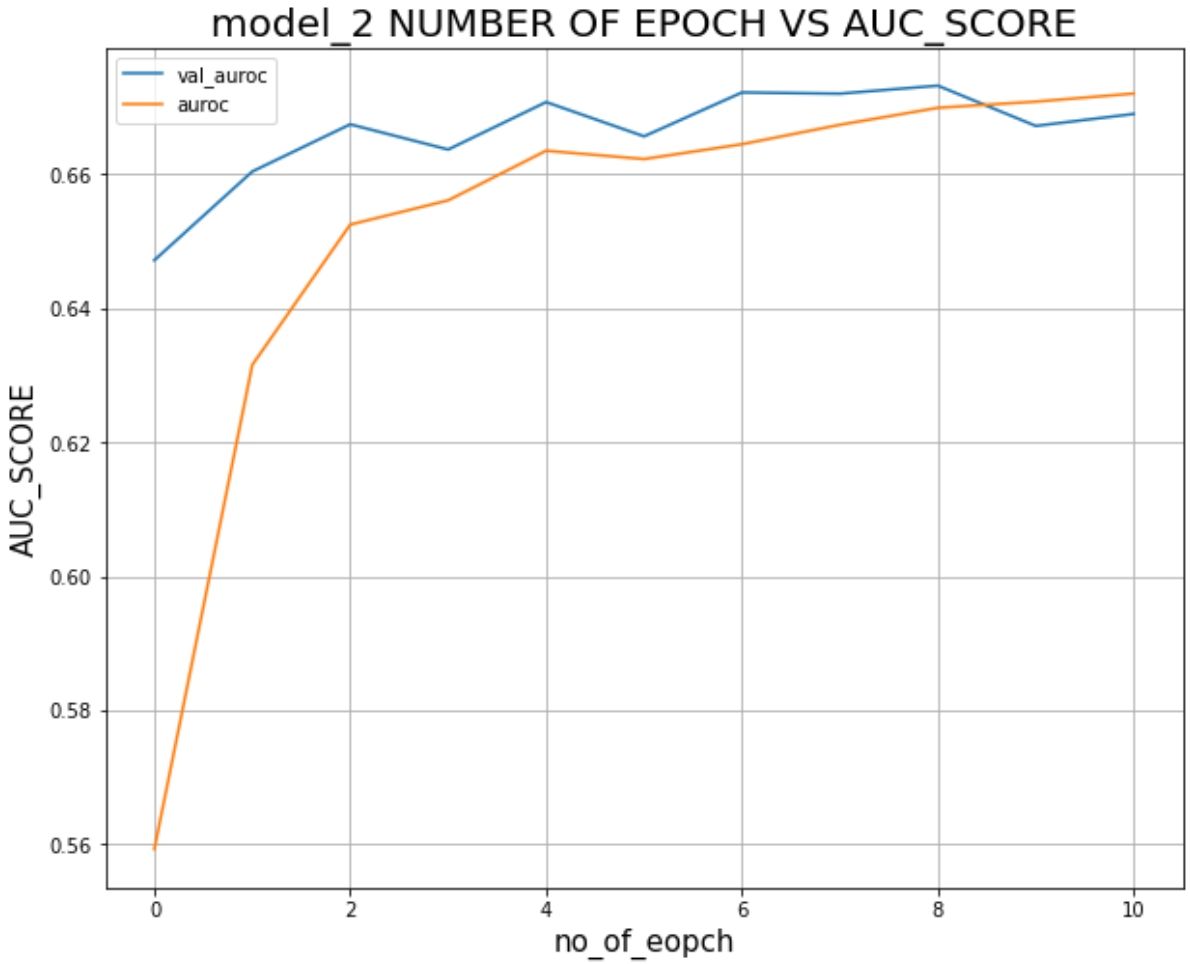
	val_loss	val_auroc	loss	auroc
0	0.906457	0.647161	1.352210	0.559291
1	0.694510	0.660402	0.790185	0.631536
2	0.600532	0.667444	0.645393	0.652481
3	0.547959	0.663682	0.575080	0.656099
4	0.507074	0.670761	0.527714	0.663503
5	0.487957	0.665652	0.499937	0.662264
6	0.469075	0.672201	0.479041	0.664473
7	0.458904	0.672024	0.463030	0.667379
8	0.447190	0.673210	0.453175	0.669908
9	0.448194	0.667206	0.444158	0.670831
10	0.437568	0.668991	0.438590	0.672036

In [0]:
```
model_2_auc=model_2_perfo.drop(['val_loss', 'loss'], axis=1)
```

2.7 model_2 NUMBER OF EPOCH VS AUC_SCORE

In [225]:
```
plt.figure()
ax=model_2_auc.plot(figsize=(10, 8))
ax.set_ylabel('AUC_SCORE',fontsize=15)
ax.set_xlabel('no_of_eopch',fontsize=15)
ax.set_title('model_2 NUMBER OF EPOCH VS AUC_SCORE',fontsize=20)
plt.grid()
```

```
<Figure size 432x288 with 0 Axes>
```

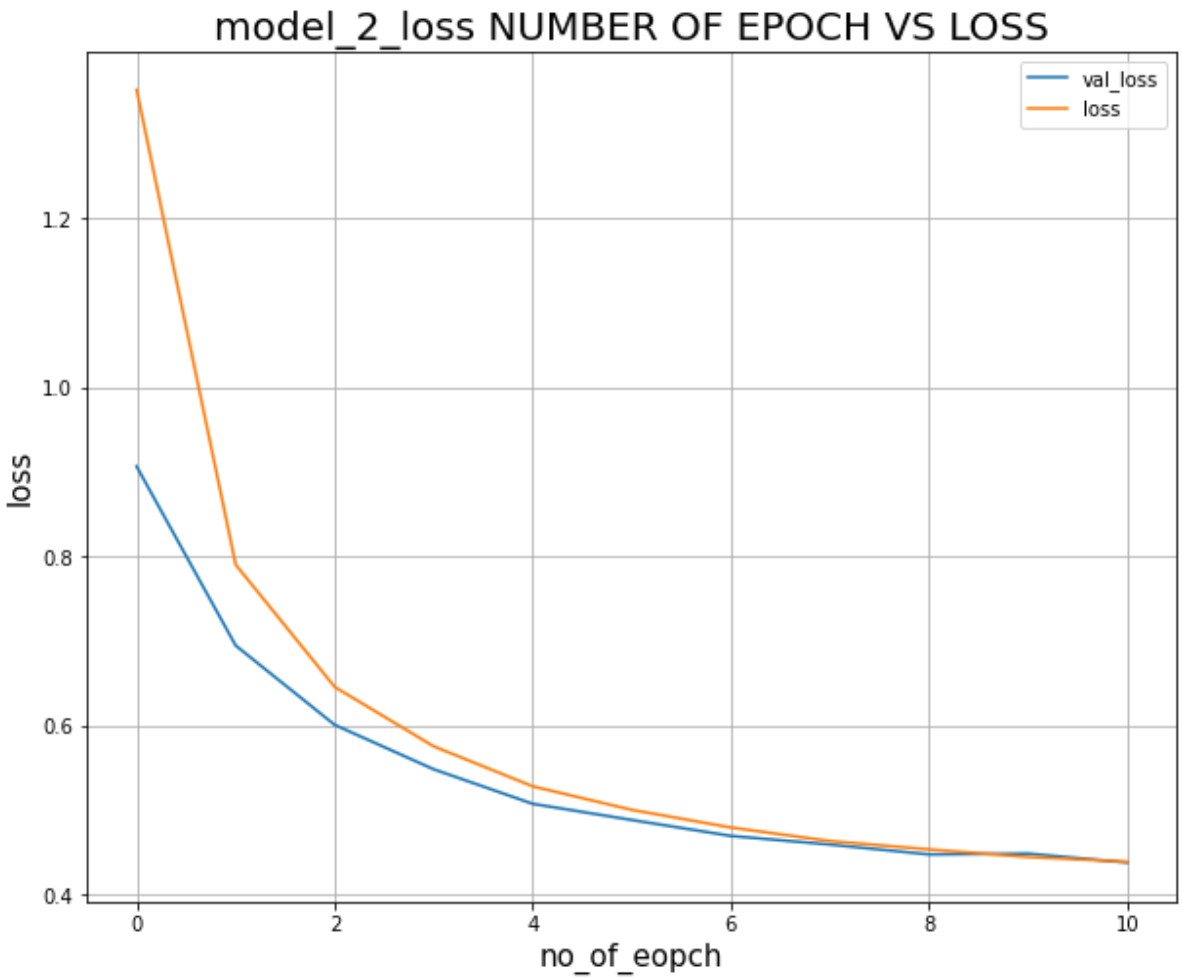


In [0]:
```
model_2_loss=model_2_perfo.drop(['val_auroc', 'auroc'], axis=1)
```

2.8 model_2 NUMBER OF EPOCH VS Loss

In [227]:
```
plt.figure()
ax=model_2_loss.plot(figsize=(10, 8))
ax.set_ylabel('loss',fontsize=15)
ax.set_xlabel('no_of_eopch',fontsize=15)
ax.set_title('model_2_loss NUMBER OF EPOCH VS LOSS',fontsize=20)
plt.grid()
```

```
<Figure size 432x288 with 0 Axes>
```

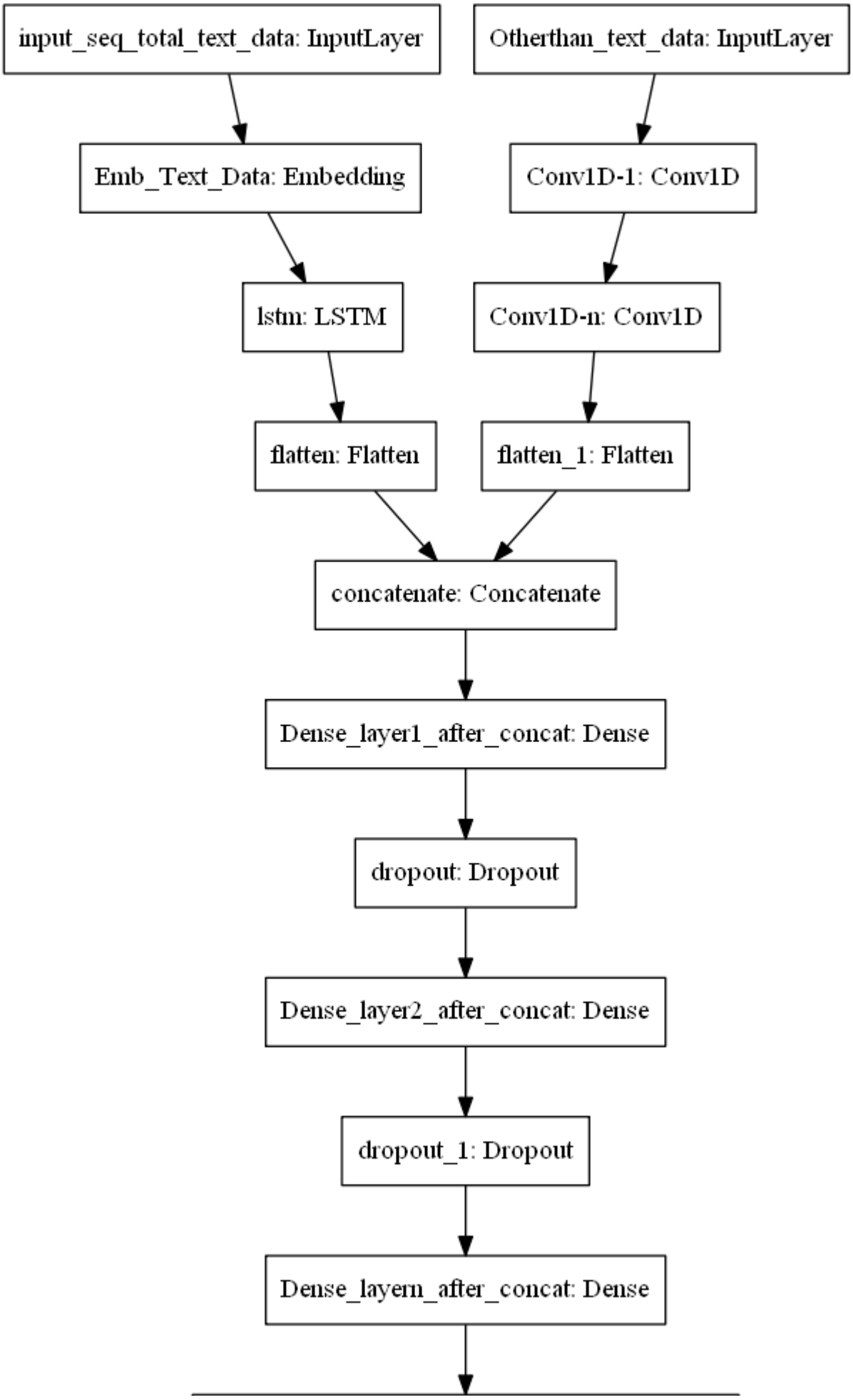


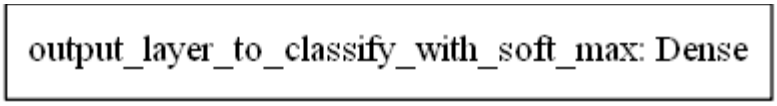
2.9 Train and Test AUC

In [228]:
```
y_train_pred_2 = model_2.predict(X_train_final_2)
print("model_2 Train AUC:",roc_auc_score(Y_train,y_train_pred_2))

y_test_pred_2 = model_2.predict(X_test_final_2)
print("model_2 Test AUC:",roc_auc_score(Y_test,y_test_pred_2))
```

```
model_2 Train AUC: 0.6927030914028343
model_2 Test AUC: 0.6688508216715249
```

MODEL 3


input_seq_total_text_data: InputLayer
Otherthan_text_data: InputLayer
Emb_Text_Data: Embedding
Conv1D-1: Conv1D
lstm: LSTM
Conv1D-n: Conv1D
flatten: Flatten
flatten_1: Flatten
concatenate: Concatenate
Dense_layer1_after_concat: Dense
dropout: Dropout
Dense_layer2_after_concat: Dense
dropout_1: Dropout
Dense_layern_after_concat: Dense



ref: https://i.imgur.com/fkQ8nGo.png (https://i.imgur.com/fkQ8nGo.png)

- **input_seq_total_text_data**:

. Use text column('essay'), and use the Embedding layer to get word vectors.

. Use given predefined glove word vectors, don't train any word vectors.

. Use LSTM that is given above, get the LSTM output and Flatten that output.

. You are free to preprocess the input text as you needed.

- **Other_than_text_data**:

. Convert all your Categorical values to onehot coded and then concatenate al
l these onehot vectors

. Neumerical values and use CNN1D (https://keras.io/getting-started/sequentia
l-model-guide/#sequence-classification-with-1d-convolutions) as shown in above
figure.

. You are free to choose all CNN parameters like kernel sizes, stride.

</pre>

3.1 Tokeninzing Text data essay

In [0]:
```
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Dono
rs_Choose.ipynb
#tokenizer keras : https://stackoverflow.com/a/51956230
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train["essay"].tolist())
X_train_essay_tokenized_3 = tokenizer.texts_to_sequences(X_train["essay"].valu
es)
X_test_essay_tokenized_3 = tokenizer.texts_to_sequences(X_test["essay"].values
)
```

In [0]:
```
from keras.preprocessing.sequence import pad_sequences
```

In [0]:
```
#this is done to make the input to the fisrt layer same length
X_train_padded_3 = pad_sequences(X_train_essay_tokenized_3,maxlen=300,padding=
'post', truncating='post')
X_test_padded_3 = pad_sequences(X_test_essay_tokenized_3, maxlen=300,padding=
'post', truncating='post')
```

In [31]:
```
print(X_train_padded_3.shape)
print(X_test_padded_3.shape)
```

```
(73196, 300)
(36052, 300)
```

In [0]:
```
import pickle
with open('glove_vectors', 'rb') as f:
    glove = pickle.load(f)
    glove_words =  set(glove.keys())
```

In [0]:
```
#https://learn-neural-networks.com/world-embedding-by-keras/

vocab_size = len(tokenizer.word_index) + 1

 #if integer data is encoded with values
# from 0 to 10, then the size of the dictionary will be 11 words.
# create a weight matrix for words in training docs
#https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

#max_vocabulary_length = len(tokenizer.word_index)

embedding_matrix = np.zeros((vocab_size, 300))

for word, i in tokenizer.word_index.items():

    if word in glove_words:
      embedding_vector = glove[word]
      embedding_matrix[i] = embedding_vector
```

3.2 Embedding and lstm layer on essay text

In [0]:
```
#flatten 1 of embbeded text
essay = Input(shape=(300,))
X = Embedding(output_dim=300,
              input_dim=vocab_size,
              input_length=len(padded_train[0]) ,
              weights=[embedding_matrix])(essay)
lstm_essay = LSTM(100,
                  recurrent_dropout=0.5,
                  kernel_regularizer=regularizers.l2(0.001),
                  return_sequences=True)(X)
flatten_essay_3 = Flatten()(lstm_essay)
```

3.3 One hot encoding of Category

In [33]:
```
##https://www.appliedaicourse.com/course/11/Applied-Machine-learning-course


# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(lowercase=False, binary=True)

X_train_categories_one_hot = vectorizer.fit_transform(X_train['clean_categories'].values)#fit to only training dataset
print(vectorizer.get_feature_names())
print("="*100)
print("Shape of matrix X_train_categories_one_hot after one hot encodig ",X_train_categories_one_hot.shape)

X_test_categories_one_hot = vectorizer.transform(X_test['clean_categories'].values)
print("Shape of matrix X_test_categories_one_hot  after one hot encodig ",X_test_categories_one_hot.shape)
```

```
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_science', 'music_arts', 'specialneeds', 'warmth']
====================================================================================================
Shape of matrix X_train_categories_one_hot after one hot encodig  (73196, 9)
Shape of matrix X_test_categories_one_hot  after one hot encodig  (36052, 9)
```

3.4 One hot encoding of subCategory

In [34]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(lowercase=False, binary=True)

X_train_sub_categories_one_hot = vectorizer.fit_transform(X_train['clean_subca
tegories'].values)
print(vectorizer.get_feature_names())

print("="*125)
print("Shape of matrix X_train_sub_categories_one_hot after one hot encodig ",
X_train_sub_categories_one_hot.shape)

X_test_sub_categories_one_hot = vectorizer.transform(X_test['clean_subcategori
es'].values)
print("Shape of matrix X_test_sub_categories_one_hot after one hot encodig ",X
_test_sub_categories_one_hot.shape)
```

```
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government',
'college_careerprep', 'communityservice', 'earlydevelopment', 'economics', 'e
nvironmentalscience', 'esl', 'extracurricular', 'financialliteracy', 'foreign
languages', 'gym_fitness', 'health_lifescience', 'health_wellness', 'history_
geography', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutrit
ioneducation', 'other', 'parentinvolvement', 'performingarts', 'socialscience
s', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
==========================================================================
==============================================
Shape of matrix X_train_sub_categories_one_hot after one hot encodig  (73196,
30)
Shape of matrix X_test_sub_categories_one_hot after one hot encodig  (36052,
30)
```

3.5 One hot encoding of School state

In [35]:
```
# we use count vectorizer to convert the values into one hot encoded features
#from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(lowercase=False, binary=True)

X_train_school_state_one_hot=vectorizer.fit_transform(X_train['school_state'].
values)
print(vectorizer.get_feature_names())

print("="*125)
print("Shape of matrix X_train_state after one hot encodig ",X_train_school_st
ate_one_hot.shape)


X_test_school_state_one_hot = vectorizer.transform(X_test['school_state'].valu
es)
print("Shape of matrix X_test_school_state_one_hot after one hot encodig ",X_t
est_school_state_one_hot.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'i
a', 'id', 'il', 'in', 'ks', '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 X_train_state after one hot encodig  (73196, 51)
Shape of matrix X_test_school_state_one_hot after one hot encodig  (36052, 5
1)
```

3.6 One hot encoding of Teacher Prefix

In [39]:
```
# we use count vectorizer to convert the values into one hot encoded features
#search term "np.nan is an invalid document, expected byte or unicode strin
g.":::https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-le
arn-valueerror-np-nan-is-an-invalid-document/39308809
# search tern "np.nan is an invalid document, expected byte or unicode string"
::https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn
-valueerror-np-nan-is-an-invalid-document


vectorizer = CountVectorizer(lowercase=False, binary=True)

X_train_teacher_prefix_one_hot = vectorizer.fit_transform(X_train['teacher_pre
fix'].values.astype('U'))
print(vectorizer.get_feature_names())
print("="*125)
print("Shape of X_train_teacher_prefix___one_hot matrix after one hot encodig
 ",X_train_teacher_prefix_one_hot.shape)


X_test_teacher_prefix_one_hot = vectorizer.transform(X_test['teacher_prefix'].
values.astype('U'))
print("Shape of X_test_teacher_prefix_one_hot matrix after one hot encodig ",X
_test_teacher_prefix_one_hot.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
=============================================================================
=============================================
Shape of X_train_teacher_prefix___one_hot matrix after one hot encodig  (7319
6, 5)
Shape of X_test_teacher_prefix_one_hot matrix after one hot encodig  (36052,
5)
```

3.7 One hot encoding of Project_grade_category

In [40]:
```
#reference::https://www.appliedaicourse.com/course/applied-ai-course-online
#https://www.kaggle.com/naveennagari/donorschoose-eda-and-tsne/notebook

vectorizer = CountVectorizer(lowercase=False, binary=True)

X_train_project_grade_one_hot = vectorizer.fit_transform(X_train['project_grade_category'])
print(vectorizer.get_feature_names())
print("="*125)
print("Shape of X_train_project_grade_one_hot matrix after one hot encodig for school states ",X_train_project_grade_one_hot.shape)



X_test_project_grade_one_hot = vectorizer.transform(X_test['project_grade_category'])
print("Shape of X_test_project_grade_one_hot matrix after one hot encodig for  school states ",X_test_project_grade_one_hot.shape)
```

```
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
=============================================================================================
Shape of X_train_project_grade_one_hot matrix after one hot encodig for school states  (73196, 4)
Shape of X_test_project_grade_one_hot matrix after one hot encodig for school states  (36052, 4)
```

3.8 Hstacking of One encoded features

In [0]:
```
from scipy.sparse import hstack
X_train_categorical_one_hot_all = hstack([X_train_categories_one_hot,
                                          X_train_sub_categories_one_hot,
                                          X_train_teacher_prefix_one_hot,
                                          X_train_school_state_one_hot,
                                          X_train_project_grade_one_hot]).todense()


X_test_categorical_one_hot_all = hstack([X_test_categories_one_hot,
                                         X_test_sub_categories_one_hot,
                                         X_test_teacher_prefix_one_hot,
                                         X_test_school_state_one_hot,
                                         X_test_project_grade_one_hot]).todense()
```

In [127]:
```
print("shape train :",X_train_categorical_one_hot_all.shape)
print("shape test :",X_test_categorical_one_hot_all.shape)
```

```
shape train : (73196, 99)
shape test : (36052, 99)
```

3.9 Vectorizing numerical Features

In [0]:
```
#https://github.com/vishnurapps/LSTM-on-Donors-Choose/blob/master/LSTM_On_Donors_Choose.ipynb
price_train=X_train['price'].values.reshape(-1,1)
price_test=X_test['price'].values.reshape(-1,1)


teacher_number_of_previously_posted_projects_train=X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)
teacher_number_of_previously_posted_projects_test=X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)

numeric_features_price_teacher_train_3=np.concatenate((price_train,
                                    teacher_number_of_previously_posted_projects_train),
                                   axis=1)


numeric_features_price_teacher_test_3=np.concatenate((price_test,
                                    teacher_number_of_previously_posted_projects_test),
                                   axis=1)
```

In [0]:
```
scalar=StandardScaler()

numeric_features_price_teacher_train_scaled_3=scalar.fit_transform(numeric_features_price_teacher_train_3)
numeric_features_price_teacher_test_scaled_3=scalar.transform(numeric_features_price_teacher_test_3)
```

In [130]:
```
numeric_features_price_teacher_train_scaled_3.shape
```

Out[130]:
```
(73196, 2)
```

3.10 Combining Non Text Data

In [0]:
```
from scipy.sparse import hstack
X_train_with_out_text = np.hstack([X_train_categorical_one_hot_all,
                                   numeric_features_price_teacher_train_scaled_3])


X_test_with_out_text = np.hstack([X_test_categorical_one_hot_all,
                                  numeric_features_price_teacher_test_scaled_3])
```

In [132]:
```
print("shape train :",X_train_with_out_text.shape)
print("shape test :",X_test_with_out_text.shape)
```

```
shape train : (73196, 101)
shape test : (36052, 101)
```

3.10 Conv layer on non text Data

In [0]:
```
with_out_text_ip = Input(shape=(101,1),name="with_out_text_ip")
x1_with_out_text = Conv1D(filters=128,
                          kernel_size=3,
                          activation='relu',
                          kernel_initializer="he_normal")(with_out_text_ip)

x2_with_out_text = Conv1D(filters=128,
                          kernel_size=3,
                          activation='relu',
                          kernel_initializer="he_normal")(x1_with_out_text)

x_with_out_text_conv_3= Flatten()(x2_with_out_text)
```

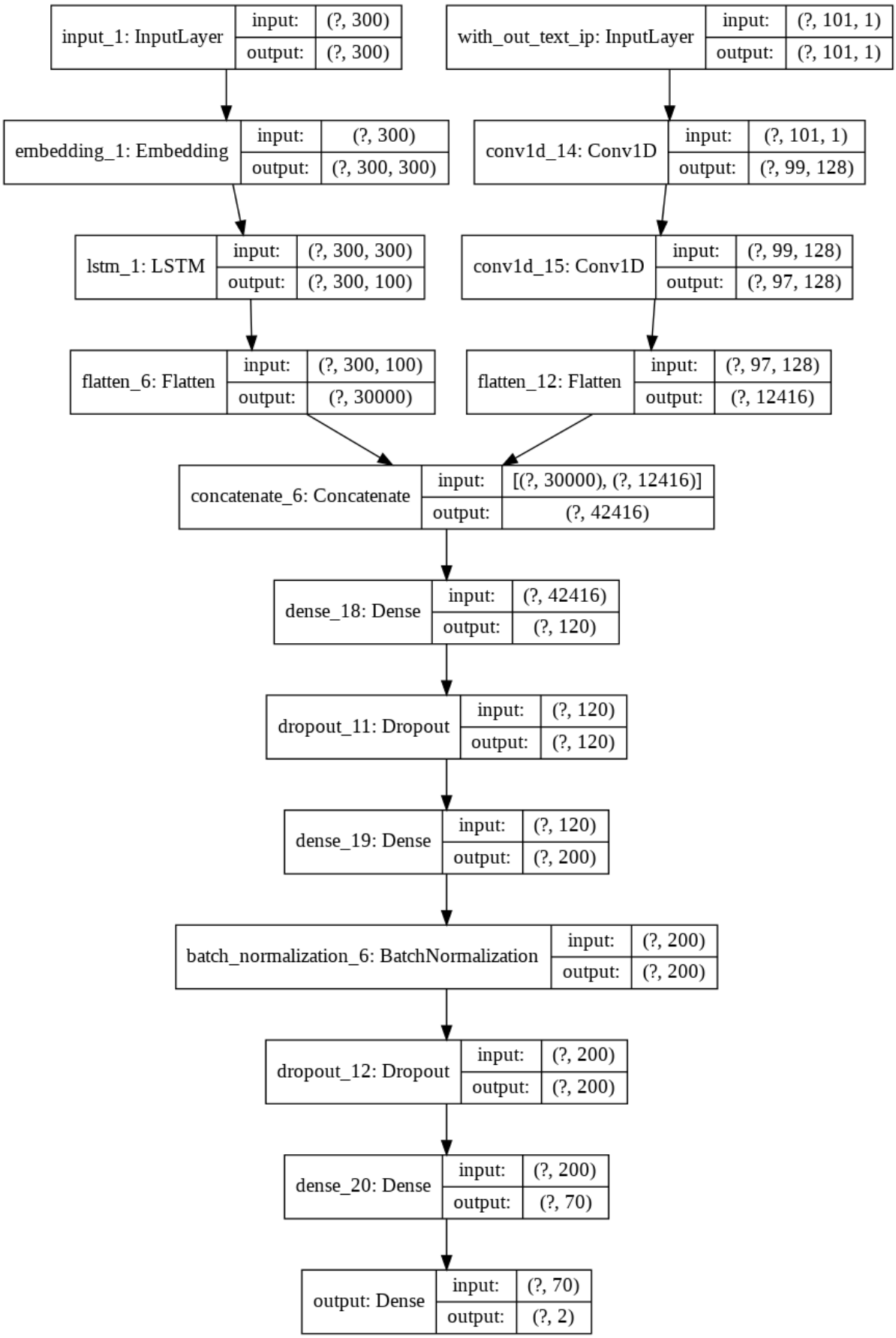
3.11 Model_3 Deep Learing model

In [0]:
```
x_concat_3 = concatenate([flatten_essay_3 ,
                          x_with_out_text_conv_3])


x_model_3 = Dense(120,
          activation="relu",
          kernel_initializer="he_normal" ,
          kernel_regularizer=regularizers.l2(0.001))(x_concat_3)

x1_model_3=Dropout(0.5)(x_model_3)

x2_model_3 = Dense(200,
          activation="sigmoid",
          kernel_initializer="glorot_normal" ,
          kernel_regularizer=regularizers.l2(0.001))(x1_model_3)

x3_model_3 = BatchNormalization()(x2_model_3)

x4_model_3=Dropout(0.5)(x3_model_3)

x5_model_3 = Dense(70,
                   activation="relu",
                   kernel_initializer="he_normal" ,
                   kernel_regularizer=regularizers.l2(0.001))(x4_model_3)



output_3 = Dense(2, activation='softmax', name='output')(x5_model_3)

model_3 = Model(inputs=[essay, with_out_text_ip],
                outputs=[output_3])
```

3.12 Model_3 Architecture and Summary

In [102]:
```
# summarize the model
from tensorflow.keras.utils import plot_model

plot_model(model_3, 'model_3.png', show_shapes=True)
```

Out[102]:


input_1: InputLayer
input: (?, 300)
output: (?, 300)
with_out_text_ip: InputLayer
input: (?, 101, 1)
output: (?, 101, 1)
embedding_1: Embedding
input: (?, 300)
output: (?, 300, 300)
conv1d_14: Conv1D
input: (?, 101, 1)
output: (?, 99, 128)
lstm_1: LSTM
input: (?, 300, 300)
output: (?, 300, 100)
conv1d_15: Conv1D
input: (?, 99, 128)
output: (?, 97, 128)
flatten_6: Flatten
input: (?, 300, 100)
output: (?, 30000)
flatten_12: Flatten
input: (?, 97, 128)
output: (?, 12416)
concatenate_6: Concatenate
input: [(?, 30000), (?, 12416)]
output: (?, 42416)
dense_18: Dense
input: (?, 42416)
output: (?, 120)
dropout_11: Dropout
input: (?, 120)
output: (?, 120)
dense_19: Dense
input: (?, 120)
output: (?, 200)
batch_normalization_6: BatchNormalization
input: (?, 200)
output: (?, 200)
dropout_12: Dropout
input: (?, 200)
output: (?, 200)
dense_20: Dense
input: (?, 200)
output: (?, 70)
output: Dense
input: (?, 70)
output: (?, 2)

In [103]:
```
print(model_3.summary())
```

```
Model: "model_6"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_1 (InputLayer)            (None, 300)          0
__________________________________________________________________________________________
with_out_text_ip (InputLayer)   (None, 101, 1)       0
__________________________________________________________________________________________
embedding_1 (Embedding)         (None, 300, 300)     14447100    input_1[0]
[0]
__________________________________________________________________________________________
conv1d_14 (Conv1D)              (None, 99, 128)      512         with_out_tex
t_ip[0][0]
__________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 300, 100)     160400      embedding_1
[0][0]
__________________________________________________________________________________________
conv1d_15 (Conv1D)              (None, 97, 128)      49280       conv1d_14[0]
[0]
__________________________________________________________________________________________
flatten_6 (Flatten)             (None, 30000)        0           lstm_1[0][0]
__________________________________________________________________________________________
flatten_12 (Flatten)            (None, 12416)        0           conv1d_15[0]
[0]
__________________________________________________________________________________________
concatenate_6 (Concatenate)     (None, 42416)        0           flatten_6[0]
[0]
                                                                 flatten_12
[0][0]
__________________________________________________________________________________________
dense_18 (Dense)                (None, 120)          5090040     concatenate_
6[0][0]
__________________________________________________________________________________________
dropout_11 (Dropout)            (None, 120)          0           dense_18[0]
[0]
__________________________________________________________________________________________
dense_19 (Dense)                (None, 200)          24200       dropout_11
[0][0]
__________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 200)          800         dense_19[0]
[0]
__________________________________________________________________________________________
```

```
dropout_12 (Dropout)            (None, 200)          0           batch_normal
ization_6[0][0]
__________________________________________________________________________________
_____________________
dense_20 (Dense)                (None, 70)           14070       dropout_12
[0][0]
__________________________________________________________________________________
_____________________
output (Dense)                  (None, 2)            142         dense_20[0]
[0]
==================================================================================
=====================
Total params: 19,786,544
Trainable params: 19,786,144
Non-trainable params: 400
__________________________________________________________________________________
_____________________
None
```

3.13 Data preparation and dimensions checking for model_3

In [0]:
```
#https://www.tutorialspoint.com/numpy/numpy_expand_dims.htm
X_train_with_out_text=np.expand_dims(X_train_with_out_text,1)
X_test_with_out_text=np.expand_dims(X_test_with_out_text,1)
```

In [136]:
```
X_train_with_out_text.shape
```

Out[136]: (73196, 1, 101)

In [0]:
```
X_train_with_out_text=np.reshape(X_train_with_out_text, (73196, 101,1))
```

In [138]:
```
X_train_with_out_text.shape
```

Out[138]: (73196, 101, 1)

In [139]:
```
X_test_with_out_text.shape
```

Out[139]: (36052, 1, 101)

In [141]:
```
X_test_with_out_text=np.reshape(X_test_with_out_text, (36052, 101,1))
print(X_test_with_out_text.shape)
```

```
(36052, 101, 1)
```

In [0]:
```
X_train_final_3 = [X_train_padded_3,X_train_with_out_text]

X_test_final_3 =[X_test_padded_3,X_test_with_out_text]
```

In [0]:
```
#X_train_final_3=np.array(X_train_final_3)
```

In [0]:
```
#X_train_final_3_reshaped= np.reshape(X_train_final_3, (X_train_final_3.shape
[0], 1, X_train_final_3.shape[1]))
#X_test_final_3_reshaped = np.reshape(X_test_final_3, (X_test_final_3.shape
[0], 1, X_test_final_3.shape[1]))
```

In [0]:
```
Y_train = np_utils.to_categorical(y_train, 2)
Y_test = np_utils.to_categorical(y_test, 2)
```

In [0]:
```
#y_train[1000:1500]
```

In [0]:
```
#Y_train[1000:1500]
```

3.14 Model_3 Complilation and model_3 Testing

In [0]:
```
from keras.callbacks import EarlyStopping,TensorBoard

checkpoint_3 = ModelCheckpoint("model_3.h5",
                               monitor="val_auroc",
                               mode="max",
                               save_best_only = True,
                               verbose=1)
early_stopping = EarlyStopping(monitor='val_auroc',
                           min_delta=0,
                           patience=2,
                           verbose=2, mode='auto')

NAME = 'model_3'
tensorboard_callback_3 =  TensorBoard(log_dir="logs_3",histogram_freq=1)
callbacks_3 = [tensorboard_callback_3,checkpoint_3,early_stopping]
```

In [0]:
```
from sklearn.metrics import roc_auc_score
```

In [0]:
```
model_3.compile(optimizer=Adam(lr=0.001),
                loss='categorical_crossentropy',
                metrics=[auroc])
```

In [150]:
```
history_3 = model_3.fit(X_train_final_3,
                        Y_train,batch_size=512,
                        epochs=15,
                        validation_data=(X_test_final_3,Y_test),
                        verbose=10,callbacks=callbacks_3)
```

```
Train on 73196 samples, validate on 36052 samples
Epoch 1/15

Epoch 00001: val_auroc improved from -inf to 0.69569, saving model to model_
3.h5
Epoch 2/15

Epoch 00002: val_auroc improved from 0.69569 to 0.74702, saving model to mode
l_3.h5
Epoch 3/15

Epoch 00003: val_auroc did not improve from 0.74702
Epoch 00003: early stopping
```

In [151]:
```
print(history_3.history)
```

```
{'val_loss': [0.7841685609337123, 0.7003731213424892, 0.5894991822344844], 'v
al_auroc': [0.6956866383552551, 0.7470161318778992, 0.7416136860847473], 'los
s': [1.0169004057592312, 0.6371915282086015, 0.5295879574406765], 'auroc':
[0.5482013, 0.6891363, 0.746603]}
```

In [0]:
```
model_3_perfo=pd.DataFrame(history_3.history)
```

In [154]:
```
model_3_perfo
```

Out[154]:

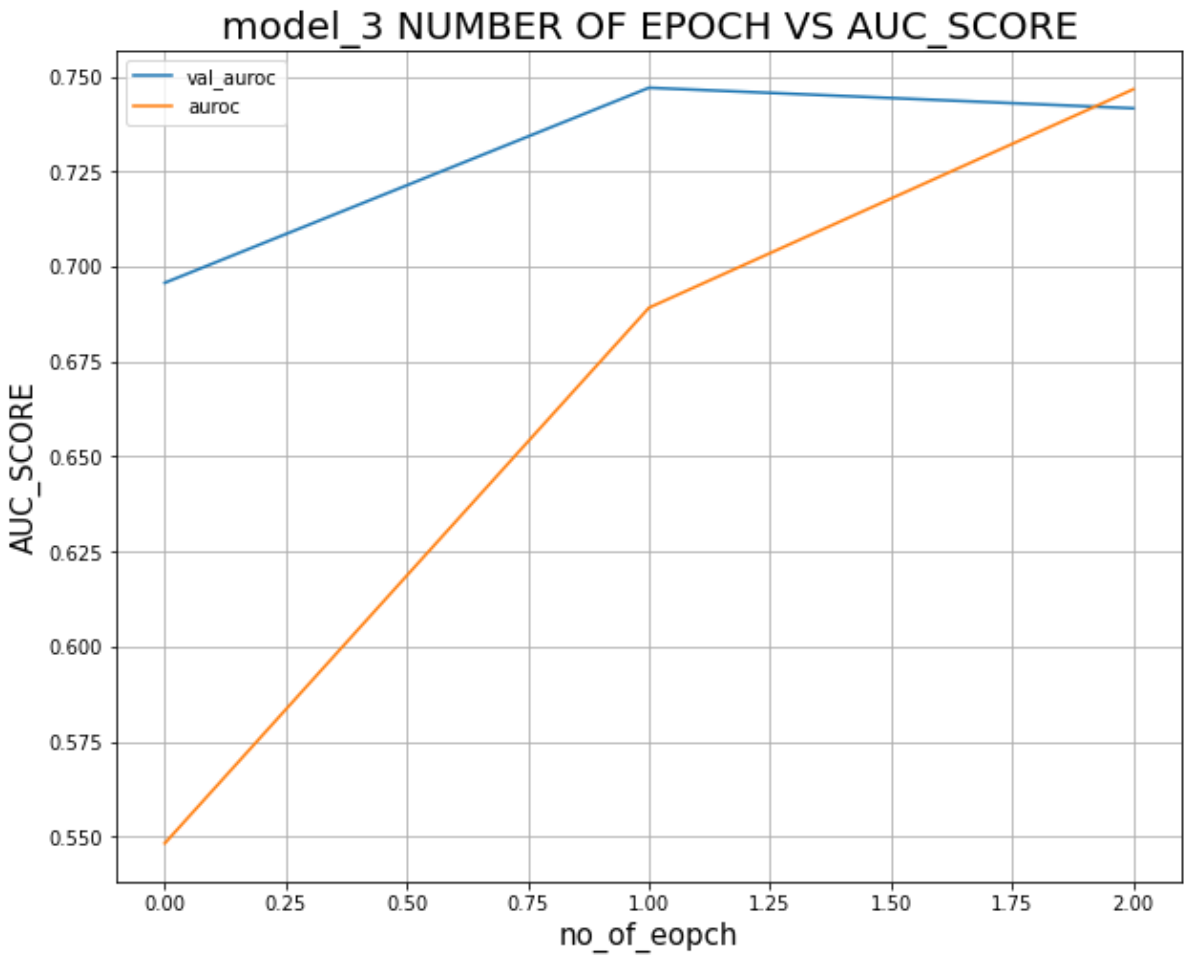
	val_loss	val_auroc	loss	auroc
0	0.784169	0.695687	1.016900	0.548201
1	0.700373	0.747016	0.637192	0.689136
2	0.589499	0.741614	0.529588	0.746603

In [0]:
```
model_3_auc=model_3_perfo.drop(['val_loss', 'loss'], axis=1)
```

3.15 model_3 NUMBER OF EPOCH VS AUC_SCORE

In [157]:
```
plt.figure()
ax=model_3_auc.plot(figsize=(10, 8))
ax.set_ylabel('AUC_SCORE',fontsize=15)
ax.set_xlabel('no_of_eopch',fontsize=15)
ax.set_title('model_3 NUMBER OF EPOCH VS AUC_SCORE',fontsize=20)
plt.grid()
```

```
<Figure size 432x288 with 0 Axes>
```



In [0]:
```
model_3_loss=model_3_perfo.drop(['val_auroc', 'auroc'], axis=1)
```

In [162]:
```
model_3_loss
```

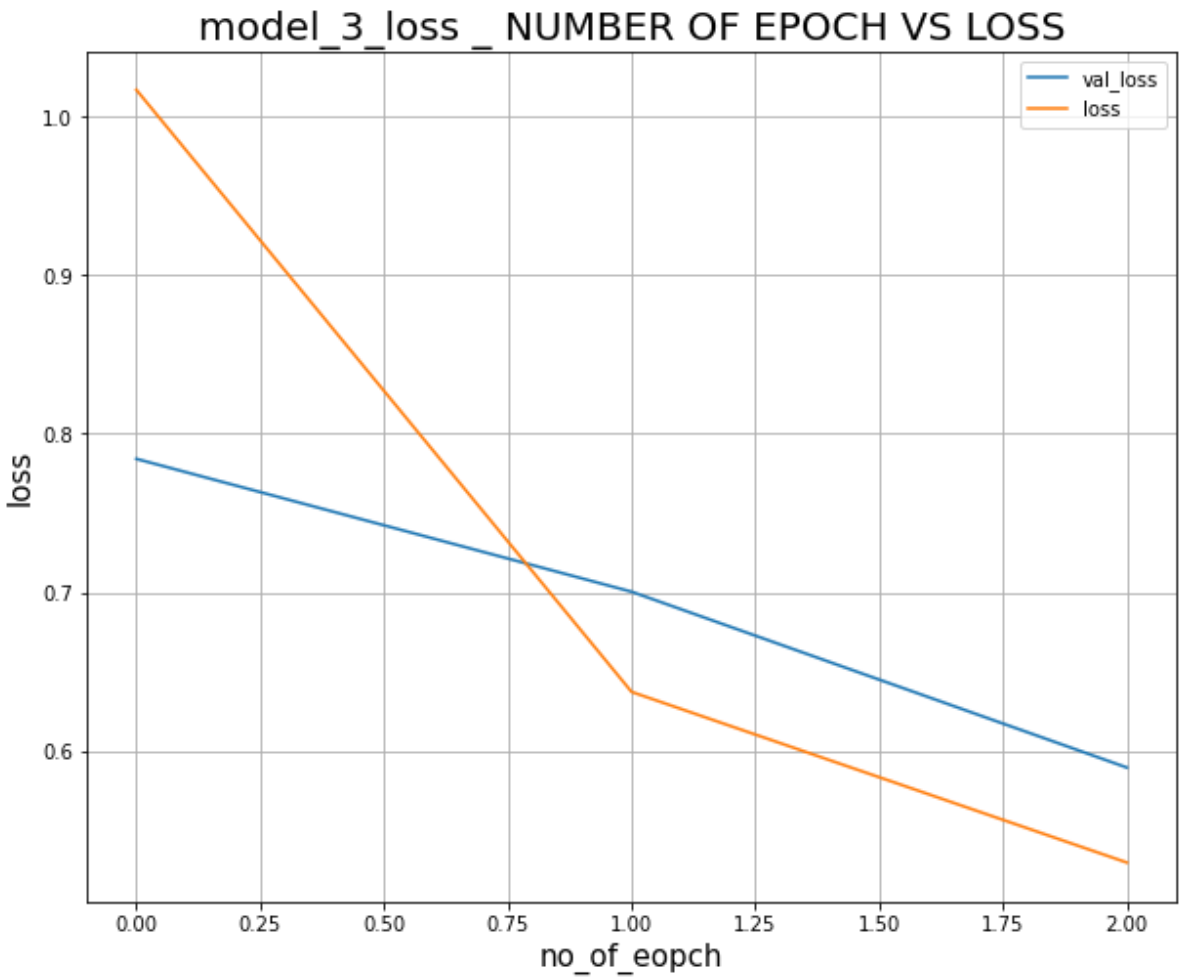
Out[162]:

	val_loss	loss
0	0.784169	1.016900
1	0.700373	0.637192
2	0.589499	0.529588

3.16 model_3*loss* NUMBER OF EPOCH VS LOSS

In [163]:
```
plt.figure()
ax=model_3_loss.plot(figsize=(10, 8))
ax.set_ylabel('loss',fontsize=15)
ax.set_xlabel('no_of_eopch',fontsize=15)
ax.set_title('model_3_loss _ NUMBER OF EPOCH VS LOSS',fontsize=20)
plt.grid()
```

```
<Figure size 432x288 with 0 Axes>
```



3.16 Train and test auc for model 3

In [165]:
```
y_train_pred_3 = model_3.predict(X_train_final_3)
print("Train AUC:",roc_auc_score(Y_train,y_train_pred_3))

y_test_pred_3 = model_3.predict(X_test_final_3)
print("Test AUC :",roc_auc_score(Y_test,y_test_pred_3))
```

```
Train AUC: 0.7822164003091008
Test AUC: 0.7419740160643506
```

CONCLUSION

In [4]:
```
#!/usr/bin/python3

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = [ "Model", "TRAIN AUC", "TEST AUC "]

x.add_row(["MODEL_1", "0.8042", '0.7547'])
x.add_row(["MODEL_2", "0.6927", '0.6688'])
x.add_row(["MODEL_3", "0.7822", '0.7419'])

print(x)
```

```
+---------+-----------+-----------+
|  Model  | TRAIN AUC | TEST AUC  |
+---------+-----------+-----------+
| MODEL_1 |   0.8042  |   0.7547  |
| MODEL_2 |   0.6927  |   0.6688  |
| MODEL_3 |   0.7822  |   0.7419  |
+---------+-----------+-----------+
```