Assignment : 14

```
1. You can work with preprocessed_data.csv for the assignment. You can get the
data from - Data folder
2. Load the data in your notebook.
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' as a metric. check this and this for using auc
as a metric
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.
7. For all the model's use TensorBoard and plot the Metric value and Loss with
epoch. While submitting, take a screenshot of plots and include those images in
a separate pad and write your observations about them.
8. Make sure that you are using GPU to train the given models.
```

In [ ]:

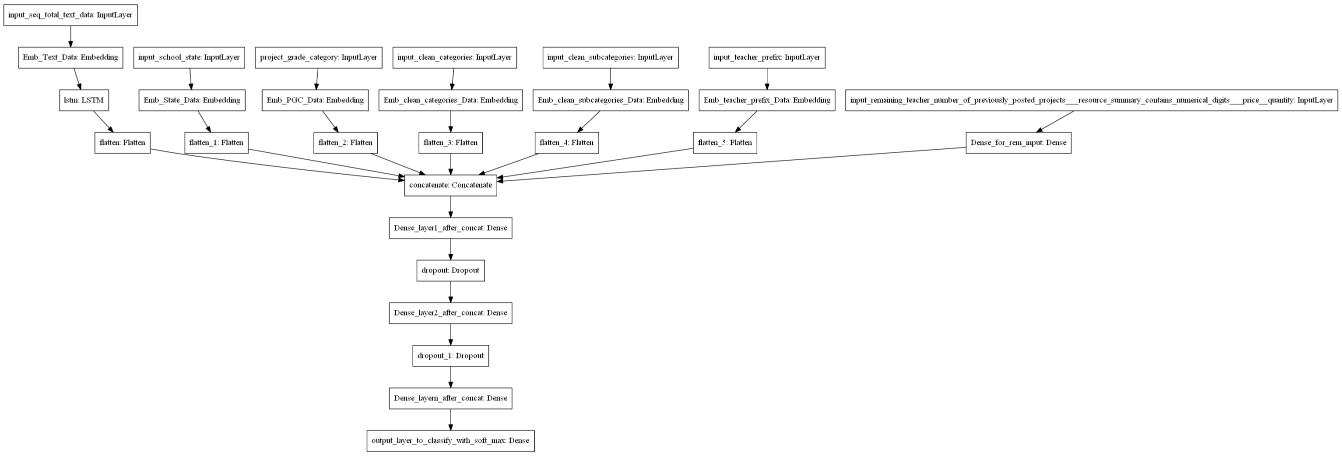
```
#you can use gdown modules to import dataset for the assignment
#for importing any file from drive to Colab you can write the syntax as !gdown --id file
#you can run the below cell to import the required preprocessed data.csv file and glove
```

In [ ]:

```
#!gdown --id 1GpATd_pM4mcnWWIs28-s1lgqdAg2Wdv-
#!gdown --id 1pGd5tLwA30M7wkbJKdXHaae9tYVDICJ_
```

Model-1

Build and Train deep neural network as shown below


input_seq_total_text_data: InputLayer
Emb_Text_Data: Embedding
input_school_state: InputLayer
project_grade_category: InputLayer
input_clean_categories: InputLayer
input_clean_subcategories: InputLayer
input_teacher_prefix: InputLayer
lstm: LSTM
Emb_State_Data: Embedding
Emb_PGC_Data: Embedding
Emb_clean_categories_Data: Embedding
Emb_clean_subcategories_Data: Embedding
Emb_teacher_prefix_Data: Embedding
input_remaining_teacher_number_of_previously_posted_projects___resource_summary_contains_numerical_digits___price__quantity: InputLayer
flatten: Flatten
flatten_1: Flatten
flatten_2: Flatten
flatten_3: Flatten
flatten_4: Flatten
flatten_5: Flatten
Dense_for_rem_input: Dense
concatenate: Concatenate
Dense_layer1_after_concat: Dense
dropout: Dropout
Dense_layer2_after_concat: Dense
dropout_1: Dropout
Dense_layern_after_concat: Dense
output_layer_to_classify_with_soft_max: Dense

ref: 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_nume** ---concatenate remaining columns and add a Dense layer after 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 [ ]:

```
# 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/

2. Please go through this link 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.

In [1]:

```
#Getting the glove_vectors, preprocessed_data from google drive
!gdown --id 1yFnPSoYrnYGBXl_vTeKJpEEna2Vl5kFp
!gdown --id 111-bcIjgYZ850hYN6ML1spMVCGV5noxW
```

```
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` wa
s deprecated in version 4.3.1 and will be removed in 5.0. You don't need to pass it anym
ore to use a file ID.
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=1yFnPSoYrnYGBXl_vTeKJpEEna2Vl5kFp
To: /content/glove_vectors
100% 128M/128M [00:03<00:00, 42.1MB/s]
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` wa
s deprecated in version 4.3.1 and will be removed in 5.0. You don't need to pass it anym
ore to use a file ID.
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=111-bcIjgYZ850hYN6ML1spMVCGV5noxW
To: /content/preprocessed_data.csv
100% 124M/124M [00:02<00:00, 56.1MB/s]
```

Model-1

In [127…

```
# import all the libraries
#make sure that you import your libraries from tf.keras and not just keras
import tensorflow as tf
import numpy as np
from tensorflow.keras.layers import Input, Dense, Embedding, Flatten, Concatenate, Dropo
from tensorflow.compat.v1.keras.layers import CuDNNLSTM
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras import Model
import pickle
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
#read the csv file
import pandas as pd
df = pd.read_csv('preprocessed_data.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	project_is_ap
0	ca	mrs	grades_prek_2	53	
1	ut	ms	grades_3_5	4	
2	ca	mrs	grades_prek_2	10	
3	ga	mrs	grades_prek_2	2	
4	wa	mrs	grades_3_5	2	

In [5]:

```
#combining all the numerical data into single column
df['remaining_input'] = df['teacher_number_of_previously_posted_projects'] +\
                        df['price']
```

In [6]:

```
y = df['project_is_approved']
df.drop(['project_is_approved', 'teacher_number_of_previously_posted_projects', 'price']
        inplace = True)
df.head()
```

Out[6]:

	school_state	teacher_prefix	project_grade_category	clean_categories	clean_subcategories	essay	remaining
0	ca	mrs	grades_prek_2	math_science	appliedsciences health_lifescience	i fortunate enough	

						use fairy tale stem kits cl...
1	ut	ms	grades_3_5	specialneeds	specialneeds	imagine 8 9 years old you third grade classroo...
2	ca	mrs	grades_prek_2	literacy_language	literacy	having class 24 students comes diverse learner...
3	ga	mrs	grades_prek_2	appliedlearning	earlydevelopment	i recently read article giving students choice...
4	wa	mrs	grades_3_5	literacy_language	literacy	my students crave challenge eat obstacles brea...

In [7]:
```
df.isnull().apply(sum)
```

Out[7]:
```
school_state              0
teacher_prefix            0
project_grade_category    0
clean_categories          0
clean_subcategories       0
essay                     0
remaining_input           0
dtype: int64
```

In [8]:
```
sum(y.isnull())
```

Out[8]:
```
0
```

In [9]:
```
# perform stratified train test split on the dataset
from sklearn.model_selection import train_test_split

x_temp, x_test, y_temp, y_test = train_test_split(df, y, test_size = 0.2, random_state =
x_train, x_cv, y_train, y_cv = train_test_split(x_temp, y_temp, test_size = 0.1, random_
                                                stratify = y_temp)

print(x_train.shape)
print(x_cv.shape)
print(x_test.shape)
```

```
(78658, 7)
(8740, 7)
(21850, 7)
```

1.1 Text Vectorization

In [ ]:

```
#since the data is already preprocessed, we can directly move to vectorization part
#first we will vectorize the text data
#for vectorization of text data in deep learning we use tokenizer, you can go through be
# https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html
#https://stackoverflow.com/questions/51956000/what-does-keras-tokenizer-method-exactly-d
# after text vectorization you should get train_padded_docs and test_padded_docs
```

In [10]:

```
my_tokenizer = Tokenizer()

my_tokenizer.fit_on_texts(x_train['essay'])
sequences_train = my_tokenizer.texts_to_sequences(x_train['essay'])
padded_train = pad_sequences(sequences_train, padding = 'post', truncating = 'pre', maxl

sequences_cv = my_tokenizer.texts_to_sequences(x_cv['essay'])
padded_cv = pad_sequences(sequences_cv, padding = 'post', truncating = 'pre', maxlen = 2

sequences_test = my_tokenizer.texts_to_sequences(x_test['essay'])
padded_test = pad_sequences(sequences_test, padding = 'post', truncating = 'pre', maxlen
```

In [11]:

```
#after getting the padded_docs you have to use predefined glove vectors to get 300 dim r
# we will be storing this data in form of an embedding matrix and will use it while defi
# Please go through following blog's 'Example of Using Pre-Trained GloVe Embedding' sect
# https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
```

In [12]:

```
import pickle
with open('glove_vectors', 'rb') as f:
    glove_vec = pickle.load(f)
glove_vec['nipping'].shape
```

Out[12]:

```
(300,)
```

In [13]:

```
tokenz = my_tokenizer.word_index
len(tokenz)
```

Out[13]:

```
49677
```

In [14]:

```
vocab_size = len(tokenz)+ 1
weight_matrix = np.zeros((vocab_size, 300))
for word,i in tokenz.items():
    if word in glove_vec:
        weight_matrix[i] = glove_vec[word]
```

In [15]:

```
weight_matrix.shape
```

Out[15]:

```
(49678, 300)
```

1.2 Categorical feature Vectorization

In [ ]:

```
# for model 1 and model 2, we have to assign a unique number to each feature in a partic
# you can either use tokenizer,label encoder or ordinal encoder to perform the task
# label encoder gives an error for 'unseen values' (values present in test but not in tr
# handle unseen values with label encoder - https://stackoverflow.com/a/56876351
# ordinal encoder also gives error with unseen values but you can use modify handle_unkn
# documentation of ordianl encoder https://scikit-learn.org/stable/modules/generated/skl
# after categorical feature vectorization you will have column_train_data and column_tes
```

In [ ]:

```
'''
school_state            0
teacher_prefix          0
```

```
project_grade_category    0
clean_categories          0
clean_subcategories       0
'''
```

In [16]:

```
#for school_state
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
school_state_train = enc.fit_transform(x_train['school_state'].values.reshape(-1,1))
print(school_state_train.shape)
enc.categories_
```

Out[16]:

```
(78658, 1)
[array(['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga',
        'hi', 'ia', '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'], dtype=object)]
```

In [17]:

```
school_state_cv = enc.transform(x_cv['school_state'].values.reshape(-1,1))
school_state_test = enc.transform(x_test['school_state'].values.reshape(-1,1))
```

In [18]:

```
#for teacher_prefix
enc1 = OrdinalEncoder()
teacher_prefix_train = enc1.fit_transform(x_train['teacher_prefix'].values.reshape(-1,1)
print(teacher_prefix_train.shape)
enc1.categories_
```

Out[18]:

```
(78658, 1)
[array(['dr', 'mr', 'mrs', 'ms', 'teacher'], dtype=object)]
```

In [19]:

```
teacher_prefix_cv = enc1.transform(x_cv['teacher_prefix'].values.reshape(-1,1))
teacher_prefix_test = enc1.transform(x_test['teacher_prefix'].values.reshape(-1,1))
```

In [20]:

```
#for project_grade_category
enc2 = OrdinalEncoder()
project_grade_category_train = enc2.fit_transform(x_train['project_grade_category'].valu
print(project_grade_category_train.shape)
enc2.categories_
```

Out[20]:

```
(78658, 1)
[array(['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2'],
       dtype=object)]
```

In [21]:

```
project_grade_category_cv = enc2.transform(x_cv['project_grade_category'].values.reshape
project_grade_category_test = enc2.transform(x_test['project_grade_category'].values.res
```

In [22]:

```
#for clean_categories
enc3 = OrdinalEncoder()
clean_categories_train = enc3.fit_transform(x_train['clean_categories'].values.reshape(-
print(clean_categories_train.shape)
enc3.categories_
```

Out[22]:

```
(78658, 1)
[array(['appliedlearning', 'appliedlearning health_sports',
        'appliedlearning history_civics',
        'appliedlearning literacy_language',
        'appliedlearning math_science', 'appliedlearning music_arts',
        'appliedlearning specialneeds',
        'appliedlearning warmth care_hunger', 'health_sports',
        'health_sports appliedlearning', 'health_sports history_civics',
        'health_sports literacy_language', 'health_sports math_science',
        'health_sports music_arts', 'health_sports specialneeds',
        'health_sports warmth care_hunger', 'history_civics',
        'history_civics appliedlearning', 'history_civics health_sports',
        'history_civics literacy_language', 'history_civics math_science',
        'history_civics music_arts', 'history_civics specialneeds',
        'history_civics warmth care_hunger', 'literacy_language',
        'literacy_language appliedlearning',
        'literacy_language health_sports',
        'literacy_language history_civics',
        'literacy_language math_science', 'literacy_language music_arts',
        'literacy_language specialneeds',
        'literacy_language warmth care_hunger', 'math_science',
        'math_science appliedlearning', 'math_science health_sports',
        'math_science history_civics', 'math_science literacy_language',
        'math_science music_arts', 'math_science specialneeds',
        'math_science warmth care_hunger', 'music_arts',
        'music_arts appliedlearning', 'music_arts health_sports',
        'music_arts history_civics', 'music_arts specialneeds',
        'music_arts warmth care_hunger', 'specialneeds',
        'specialneeds health_sports', 'specialneeds music_arts',
        'specialneeds warmth care_hunger', 'warmth care_hunger'],
       dtype=object)]
```

In [23]:

```
clean_categories_cv = enc3.transform(x_cv['clean_categories'].values.reshape(-1,1))
clean_categories_test = enc3.transform(x_test['clean_categories'].values.reshape(-1,1))
```

In [24]:

```
#for clean_subcategories
enc4 = OrdinalEncoder(handle_unknown='use_encoded_value', unknown_value=-1)
clean_subcategories_train = enc4.fit_transform(x_train['clean_subcategories'].values.res
print(clean_subcategories_train.shape)
enc4.categories_
```

```
(78658, 1)
```

Out[24]:

```
[array(['appliedsciences', 'appliedsciences charactereducation',
        'appliedsciences civics_government',
        'appliedsciences college_careerprep',
        'appliedsciences communityservice',
        'appliedsciences earlydevelopment', 'appliedsciences economics',
        'appliedsciences environmentalscience', 'appliedsciences esl',
        'appliedsciences extracurricular',
        'appliedsciences financialliteracy',
        'appliedsciences foreignlanguages', 'appliedsciences gym_fitness',
        'appliedsciences health_lifescience',
        'appliedsciences health_wellness',
        'appliedsciences history_geography', 'appliedsciences literacy',
        'appliedsciences literature_writing',
        'appliedsciences mathematics', 'appliedsciences music',
        'appliedsciences nutritioneducation', 'appliedsciences other',
        'appliedsciences parentinvolvement',
        'appliedsciences performingarts', 'appliedsciences socialsciences',
        'appliedsciences specialneeds', 'appliedsciences teamsports',
        'appliedsciences visualarts', 'appliedsciences warmth care_hunger',
        'charactereducation', 'charactereducation civics_government',
        'charactereducation college_careerprep',
        'charactereducation communityservice',
        'charactereducation earlydevelopment',
        'charactereducation economics',
        'charactereducation environmentalscience',
        'charactereducation esl', 'charactereducation extracurricular',
        'charactereducation financialliteracy',
        'charactereducation foreignlanguages',
        'charactereducation gym_fitness',
        'charactereducation health_lifescience',
        'charactereducation health_wellness',
        'charactereducation history_geography',
        'charactereducation literacy',
        'charactereducation literature_writing',
        'charactereducation mathematics', 'charactereducation music',
        'charactereducation nutritioneducation',
        'charactereducation other', 'charactereducation parentinvolvement',
        'charactereducation performingarts',
        'charactereducation socialsciences',
        'charactereducation specialneeds', 'charactereducation teamsports',
        'charactereducation visualarts',
        'charactereducation warmth care_hunger', 'civics_government',
        'civics_government college_careerprep',
        'civics_government communityservice',
        'civics_government economics',
        'civics_government environmentalscience', 'civics_government esl',
        'civics_government financialliteracy',
        'civics_government foreignlanguages',
        'civics_government health_lifescience',
        'civics_government health_wellness',
        'civics_government history_geography',
        'civics_government literacy',
        'civics_government literature_writing',
        'civics_government mathematics',
        'civics_government nutritioneducation',
        'civics_government parentinvolvement',
        'civics_government performingarts',
        'civics_government socialsciences',
        'civics_government specialneeds', 'civics_government teamsports',
        'civics_government visualarts', 'college_careerprep',
        'college_careerprep communityservice',
        'college_careerprep earlydevelopment',
        'college_careerprep economics',
        'college_careerprep environmentalscience',
        'college_careerprep esl', 'college_careerprep extracurricular',
        'college_careerprep financialliteracy',
        'college_careerprep foreignlanguages',
        'college_careerprep gym_fitness',
        'college_careerprep health_lifescience',
        'college_careerprep health_wellness',
        'college_careerprep history_geography',
        'college_careerprep literacy',
        'college_careerprep literature_writing',
        'college_careerprep mathematics', 'college_careerprep music',
        'college_careerprep nutritioneducation',
        'college_careerprep other', 'college_careerprep parentinvolvement',
        'college_careerprep performingarts',
        'college_careerprep socialsciences',
        'college_careerprep specialneeds', 'college_careerprep teamsports',
        'college_careerprep visualarts', 'communityservice',
        'communityservice earlydevelopment', 'communityservice economics',
        'communityservice environmentalscience', 'communityservice esl',
        'communityservice extracurricular',
        'communityservice financialliteracy',
        'communityservice gym_fitness',
        'communityservice health_lifescience',
        'communityservice health_wellness',
        'communityservice history_geography', 'communityservice literacy',
        'communityservice literature_writing',
        'communityservice mathematics',
        'communityservice nutritioneducation', 'communityservice other',
        'communityservice parentinvolvement',
        'communityservice performingarts',
        'communityservice socialsciences', 'communityservice specialneeds',
        'communityservice visualarts', 'earlydevelopment',
        'earlydevelopment economics',
        'earlydevelopment environmentalscience',
        'earlydevelopment extracurricular',
        'earlydevelopment financialliteracy',
        'earlydevelopment foreignlanguages',
       'earlydevelopment gym_fitness',
       'earlydevelopment health_lifescience',
       'earlydevelopment health_wellness',
       'earlydevelopment history_geography', 'earlydevelopment literacy',
       'earlydevelopment literature_writing',
       'earlydevelopment mathematics', 'earlydevelopment music',
       'earlydevelopment nutritioneducation', 'earlydevelopment other',
       'earlydevelopment parentinvolvement',
       'earlydevelopment performingarts',
       'earlydevelopment socialsciences', 'earlydevelopment specialneeds',
       'earlydevelopment teamsports', 'earlydevelopment visualarts',
       'earlydevelopment warmth care_hunger', 'economics',
       'economics environmentalscience', 'economics financialliteracy',
       'economics foreignlanguages', 'economics health_lifescience',
       'economics history_geography', 'economics literacy',
       'economics literature_writing', 'economics mathematics',
       'economics music', 'economics nutritioneducation',
       'economics other', 'economics socialsciences',
       'economics specialneeds', 'economics visualarts',
       'environmentalscience', 'environmentalscience extracurricular',
       'environmentalscience financialliteracy',
       'environmentalscience foreignlanguages',
       'environmentalscience gym_fitness',
       'environmentalscience health_lifescience',
       'environmentalscience health_wellness',
       'environmentalscience history_geography',
       'environmentalscience literacy',
       'environmentalscience literature_writing',
       'environmentalscience mathematics', 'environmentalscience music',
       'environmentalscience nutritioneducation',
       'environmentalscience other',
       'environmentalscience parentinvolvement',
       'environmentalscience performingarts',
       'environmentalscience socialsciences',
       'environmentalscience specialneeds',
       'environmentalscience teamsports',
       'environmentalscience visualarts',
       'environmentalscience warmth care_hunger', 'esl',
       'esl earlydevelopment', 'esl environmentalscience',
       'esl extracurricular', 'esl financialliteracy',
       'esl foreignlanguages', 'esl gym_fitness',
       'esl health_lifescience', 'esl health_wellness',
       'esl history_geography', 'esl literacy', 'esl literature_writing',
       'esl mathematics', 'esl music', 'esl nutritioneducation',
       'esl other', 'esl parentinvolvement', 'esl performingarts',
       'esl socialsciences', 'esl specialneeds', 'esl teamsports',
       'esl visualarts', 'extracurricular',
       'extracurricular financialliteracy',
       'extracurricular foreignlanguages', 'extracurricular gym_fitness',
       'extracurricular health_lifescience',
       'extracurricular health_wellness',
       'extracurricular history_geography', 'extracurricular literacy',
       'extracurricular literature_writing',
       'extracurricular mathematics', 'extracurricular music',
       'extracurricular nutritioneducation', 'extracurricular other',
       'extracurricular parentinvolvement',
       'extracurricular performingarts', 'extracurricular socialsciences',
       'extracurricular specialneeds', 'extracurricular teamsports',
       'extracurricular visualarts', 'financialliteracy',
       'financialliteracy foreignlanguages',
       'financialliteracy health_lifescience',
       'financialliteracy health_wellness',
       'financialliteracy history_geography',
       'financialliteracy literacy',
       'financialliteracy literature_writing',
       'financialliteracy mathematics', 'financialliteracy other',
'financialliteracy parentinvolvement',
'financialliteracy performingarts',
'financialliteracy socialsciences',
'financialliteracy specialneeds', 'financialliteracy visualarts',
'foreignlanguages', 'foreignlanguages gym_fitness',
'foreignlanguages health_lifescience',
'foreignlanguages health_wellness',
'foreignlanguages history_geography', 'foreignlanguages literacy',
'foreignlanguages literature_writing',
'foreignlanguages mathematics', 'foreignlanguages music',
'foreignlanguages other', 'foreignlanguages performingarts',
'foreignlanguages socialsciences', 'foreignlanguages specialneeds',
'foreignlanguages visualarts', 'gym_fitness',
'gym_fitness health_lifescience', 'gym_fitness health_wellness',
'gym_fitness history_geography', 'gym_fitness literacy',
'gym_fitness literature_writing', 'gym_fitness mathematics',
'gym_fitness music', 'gym_fitness nutritioneducation',
'gym_fitness other', 'gym_fitness parentinvolvement',
'gym_fitness performingarts', 'gym_fitness socialsciences',
'gym_fitness specialneeds', 'gym_fitness teamsports',
'gym_fitness visualarts', 'gym_fitness warmth care_hunger',
'health_lifescience', 'health_lifescience health_wellness',
'health_lifescience history_geography',
'health_lifescience literacy',
'health_lifescience literature_writing',
'health_lifescience mathematics', 'health_lifescience music',
'health_lifescience nutritioneducation',
'health_lifescience other', 'health_lifescience parentinvolvement',
'health_lifescience performingarts',
'health_lifescience socialsciences',
'health_lifescience specialneeds', 'health_lifescience teamsports',
'health_lifescience visualarts',
'health_lifescience warmth care_hunger', 'health_wellness',
'health_wellness history_geography', 'health_wellness literacy',
'health_wellness literature_writing',
'health_wellness mathematics', 'health_wellness music',
'health_wellness nutritioneducation', 'health_wellness other',
'health_wellness parentinvolvement',
'health_wellness performingarts', 'health_wellness socialsciences',
'health_wellness specialneeds', 'health_wellness teamsports',
'health_wellness visualarts', 'health_wellness warmth care_hunger',
'history_geography', 'history_geography literacy',
'history_geography literature_writing',
'history_geography mathematics', 'history_geography music',
'history_geography other', 'history_geography parentinvolvement',
'history_geography performingarts',
'history_geography socialsciences',
'history_geography specialneeds', 'history_geography teamsports',
'history_geography visualarts',
'history_geography warmth care_hunger', 'literacy',
'literacy literature_writing', 'literacy mathematics',
'literacy music', 'literacy nutritioneducation', 'literacy other',
'literacy parentinvolvement', 'literacy performingarts',
'literacy socialsciences', 'literacy specialneeds',
'literacy teamsports', 'literacy visualarts',
'literacy warmth care_hunger', 'literature_writing',
'literature_writing mathematics', 'literature_writing music',
'literature_writing other', 'literature_writing parentinvolvement',
'literature_writing performingarts',
'literature_writing socialsciences',
'literature_writing specialneeds', 'literature_writing teamsports',
'literature_writing visualarts',
'literature_writing warmth care_hunger', 'mathematics',
'mathematics music', 'mathematics nutritioneducation',
'mathematics other', 'mathematics parentinvolvement',
'mathematics performingarts', 'mathematics socialsciences',
        'mathematics specialneeds', 'mathematics teamsports',
        'mathematics visualarts', 'music', 'music other',
        'music parentinvolvement', 'music performingarts',
        'music socialsciences', 'music specialneeds', 'music teamsports',
        'music visualarts', 'nutritioneducation',
        'nutritioneducation other', 'nutritioneducation socialsciences',
        'nutritioneducation specialneeds', 'nutritioneducation teamsports',
        'nutritioneducation visualarts',
        'nutritioneducation warmth care_hunger', 'other',
        'other parentinvolvement', 'other performingarts',
        'other socialsciences', 'other specialneeds', 'other teamsports',
        'other visualarts', 'other warmth care_hunger',
        'parentinvolvement', 'parentinvolvement performingarts',
        'parentinvolvement socialsciences',
        'parentinvolvement specialneeds', 'parentinvolvement teamsports',
        'parentinvolvement visualarts',
        'parentinvolvement warmth care_hunger', 'performingarts',
        'performingarts socialsciences', 'performingarts specialneeds',
        'performingarts teamsports', 'performingarts visualarts',
        'socialsciences', 'socialsciences specialneeds',
        'socialsciences teamsports', 'socialsciences visualarts',
        'specialneeds', 'specialneeds teamsports',
        'specialneeds visualarts', 'specialneeds warmth care_hunger',
        'teamsports', 'teamsports visualarts', 'visualarts',
        'visualarts warmth care_hunger', 'warmth care_hunger'],
       dtype=object)]
```

In [25]:

```
clean_subcategories_cv = enc4.transform(x_cv['clean_subcategories'].values.reshape(-1,1)
clean_subcategories_test = enc4.transform(x_test['clean_subcategories'].values.reshape(-
```

1.3 Numerical feature Vectorization

In [ ]:

```
# you have to standardise the numerical columns
# stack both the numerical features
#after numerical feature vectorization you will have numerical_data_train and numerical_
```

In [26]:

```
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
remaining_input_train = std.fit_transform(x_train['remaining_input'].values.reshape(-1,1
```

In [27]:

```
remaining_input_cv = std.transform(x_cv['remaining_input'].values.reshape(-1,1))
remaining_input_test = std.transform(x_test['remaining_input'].values.reshape(-1,1))
```

Getting the target variable in this case class label column ready for processing.

In [28]:

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.utils import compute_class_weight
y_train_coded = OneHotEncoder().fit_transform(y_train.values.reshape(-1,1)).toarray()
y_test_coded = OneHotEncoder().fit_transform(y_test.values.reshape(-1,1)).toarray()
y_cv_coded = OneHotEncoder().fit_transform(y_cv.values.reshape(-1,1)).toarray()
class_wts = compute_class_weight('balanced', classes = np.unique(y), y = y)
```

In [29]:

```
y_train_coded.shape
```

Out[29]:

```
(78658, 2)
```

In [30]:

```
class_wts = {0: class_wts[0], 1: class_wts[1]}
class_wts
```

Out[30]:

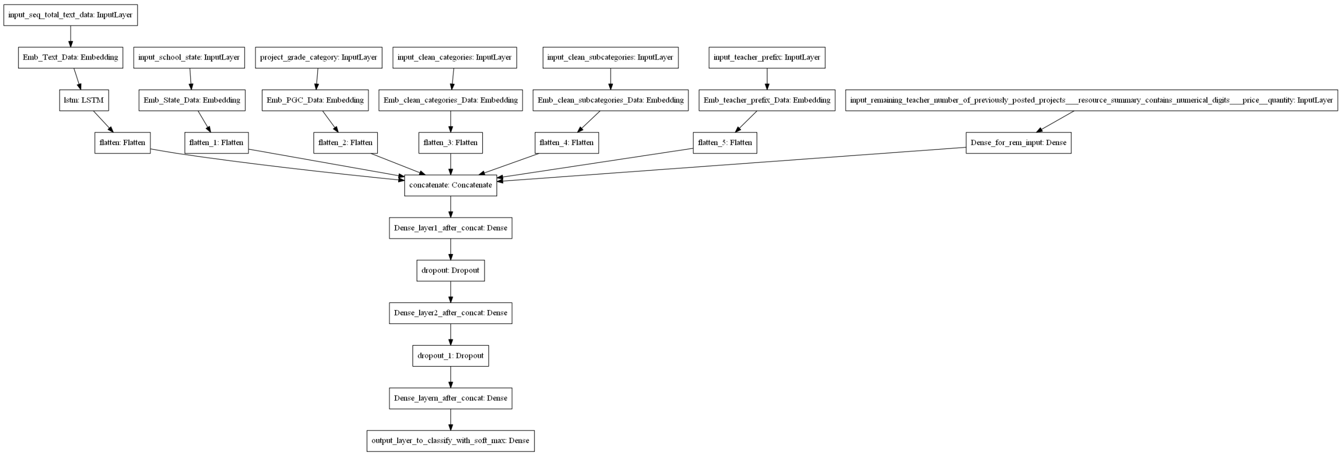
```
{0: 3.3021400072542617, 1: 0.5892175263736975}
```

In [31]:

```
print("school_state: ",len(enc.categories_[0]))
print("teacher_prefix: ",len(enc1.categories_[0]))
print("project_grade_category: ",len(enc2.categories_[0]))
print("clean_categories: ",len(enc3.categories_[0]))
print("clean_subcategories: ",len(enc4.categories_[0]))
```

```
school_state:  51
teacher_prefix:  5
project_grade_category:  4
clean_categories:  51
clean_subcategories:  395
```

1.4 Defining the model


input_seq_total_text_data: InputLayer
Emb_Text_Data: Embedding
input_school_state: InputLayer
project_grade_category: InputLayer
input_clean_categories: InputLayer
input_clean_subcategories: InputLayer
input_teacher_prefix: InputLayer
lstm: LSTM
Emb_State_Data: Embedding
Emb_PGC_Data: Embedding
Emb_clean_categories_Data: Embedding
Emb_clean_subcategories_Data: Embedding
Emb_teacher_prefix_Data: Embedding
input_remaining_teacher_number_of_previously_posted_projects___resource_summary_contains_numerical_digits___price__quantity: InputLayer
flatten: Flatten
flatten_1: Flatten
flatten_2: Flatten
flatten_3: Flatten
flatten_4: Flatten
flatten_5: Flatten
Dense_for_rem_input: Dense
concatenate: Concatenate
Dense_layer1_after_concat: Dense
dropout: Dropout
Dense_layer2_after_concat: Dense
dropout_1: Dropout
Dense_layern_after_concat: Dense
output_layer_to_classify_with_soft_max: Dense

In [ ]:

```
# as of now we have vectorized all our features now we will define our model.
# as it is clear from above image that the given model has multiple input layers and hen
# Please go through - https://keras.io/guides/functional_api/
# it is a good programming practise to define your complete model i.e all inputs , inter
# while defining your model make sure that you use variable names while defining any len
#for ex.- you should write the code as 'input_text = Input(shape=(pad_length,))' and not
# the embedding layer for text data should be non trainable
# the embedding layer for categorical data should be trainable
# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work
# https://towardsdatascience.com/deep-embeddings-for-categorical-variables-cat2vec-b05c8
#print model.summary() after you have defined the model
#plot the model using utils.plot_model module and make sure that it is similar to the ab
```

In [38]:

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

In [39]:

```
len(tokenz)
```

Out[39]:

```
49677
```

In [40]:

```
from tensorflow.keras.initializers import HeNormal
from tensorflow.keras.regularizers import L2
from tensorflow.keras.optimizers import Adam

#input 1
input1 = Input(shape=(250,))
embed1 = Embedding(input_dim = 49678, output_dim = 300, weights = [weight_matrix], train
```

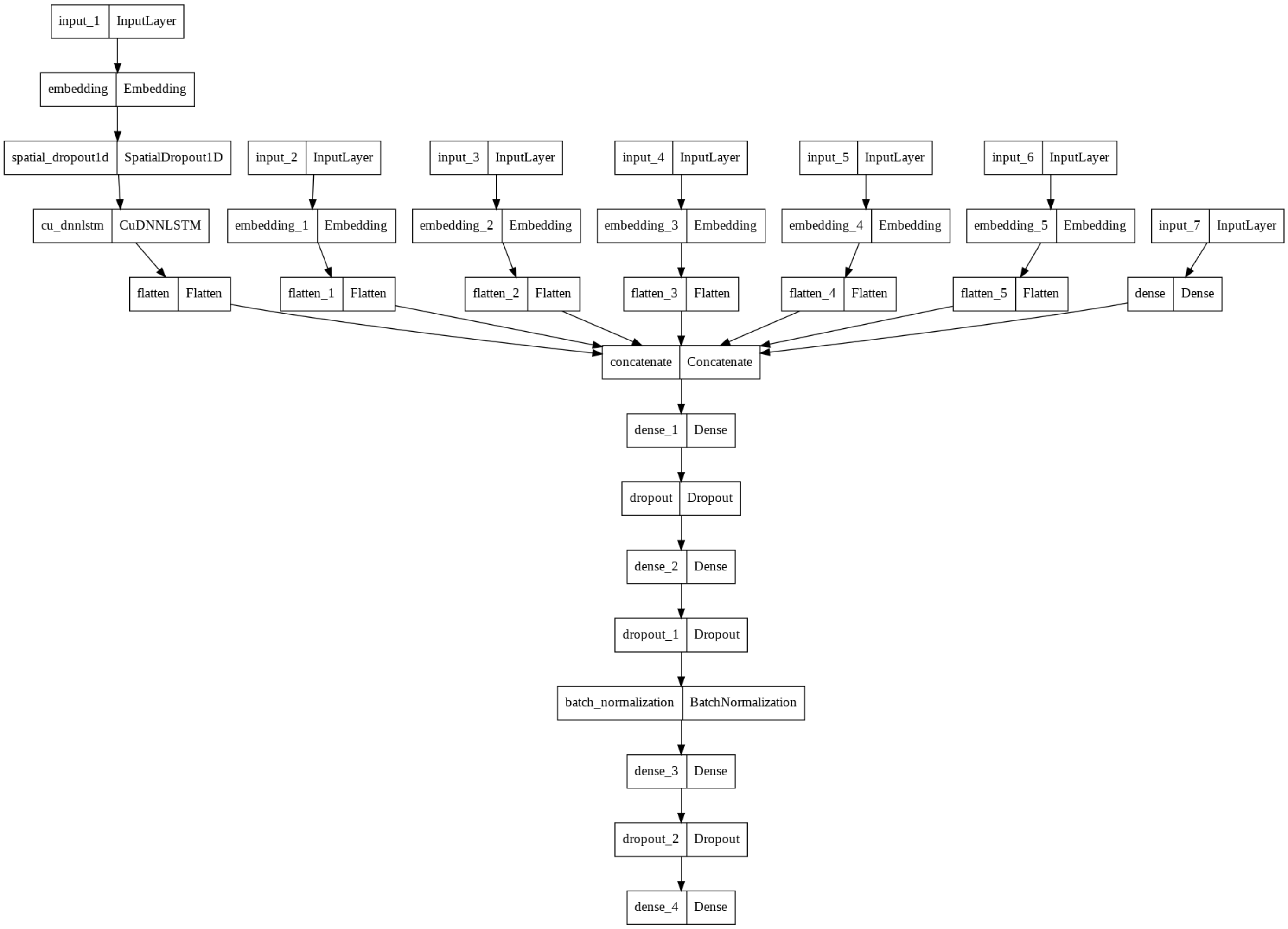
```python
dropout_layer = SpatialDropout1D(0.3)(embed1)
lstm_layer = CuDNNLSTM(128,return_sequences=True)(dropout_layer)
flat1 = Flatten()(lstm_layer)

cat_vars = ["teacher_prefix","school_state","project_grade_category","clean_categories",
cat_sizes = {}
cat_embsizes = {}
for cat in cat_vars:
    cat_sizes[cat] = x_train[cat].nunique()
    cat_embsizes[cat] = min(50, cat_sizes[cat]//2+1)

#input 2
input2 = Input(shape = (1,))
embed2 = Embedding(input_dim=cat_sizes['school_state']+1, output_dim=cat_embsizes['schoo
flat2 = Flatten()(embed2)

#input 3
input3 = Input(shape = (1,))
embed3 = Embedding(input_dim=cat_sizes['teacher_prefix']+1, output_dim=cat_embsizes['tea
flat3 = Flatten()(embed3)

#input 4
input4 = Input(shape = (1,))
embed4 = Embedding(input_dim=cat_sizes['project_grade_category']+1, output_dim=cat_embsi
flat4 = Flatten()(embed4)

#input 5
input5 = Input(shape = (1,))
embed5 = Embedding(input_dim=cat_sizes['clean_categories']+1, output_dim=cat_embsizes['c
flat5 = Flatten()(embed5)

#input 6
input6 = Input(shape = (1,))
embed6 = Embedding(input_dim=cat_sizes['clean_subcategories']+1, output_dim=cat_embsizes
flat6 = Flatten()(embed6)

#input 7
input7 = Input(shape=(1,))
dense1 = Dense(16,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
concat = Concatenate()([flat1, flat2, flat3, flat4, flat5, flat6, dense1])


dense2 = Dense(128,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulari
dropout1 = Dropout(0.5)(dense2)
dense3 = Dense(64,activation='relu', kernel_initializer = HeNormal(), kernel_regularizer
dropout2 = Dropout(0.5)(dense3)
batch_norm = BatchNormalization()(dropout2)
dense4 = Dense(32,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout3 = Dropout(0.5)(dense4)
output = Dense(2, activation = 'softmax')(dropout3)


model1 = Model([input1, input2, input3, input4, input5, input6, input7], output)
model1.compile(loss = 'categorical_crossentropy', optimizer = Adam(learning_rate = 0.000

print(model1.summary())
```

```
Model: "model"
________________________________________________________________________________________
__________
 Layer (type)                   Output Shape         Param #     Connected to

========================================================================================
==========
 input_1 (InputLayer)           [(None, 250)]        0           []
 embedding (Embedding)          (None, 250, 300)     14903400    ['input_1[0][0]']


 spatial_dropout1d (SpatialDrop  (None, 250, 300)    0           ['embedding[0][0]']

 out1D)


 input_2 (InputLayer)           [(None, 1)]          0           []


 input_3 (InputLayer)           [(None, 1)]          0           []


 input_4 (InputLayer)           [(None, 1)]          0           []


 input_5 (InputLayer)           [(None, 1)]          0           []


 input_6 (InputLayer)           [(None, 1)]          0           []


 cu_dnnlstm (CuDNNLSTM)         (None, 250, 128)     220160      ['spatial_dropout1d[0]
[0]']

 embedding_1 (Embedding)        (None, 1, 26)        1352        ['input_2[0][0]']


 embedding_2 (Embedding)        (None, 1, 3)         18          ['input_3[0][0]']


 embedding_3 (Embedding)        (None, 1, 3)         15          ['input_4[0][0]']


 embedding_4 (Embedding)        (None, 1, 26)        1352        ['input_5[0][0]']


 embedding_5 (Embedding)        (None, 1, 50)        19800       ['input_6[0][0]']


 input_7 (InputLayer)           [(None, 1)]          0           []


 flatten (Flatten)              (None, 32000)        0           ['cu_dnnlstm[0][0]']


 flatten_1 (Flatten)            (None, 26)           0           ['embedding_1[0][0]']
 flatten_2 (Flatten)            (None, 3)            0            ['embedding_2[0][0]']

 flatten_3 (Flatten)            (None, 3)            0            ['embedding_3[0][0]']

 flatten_4 (Flatten)            (None, 26)           0            ['embedding_4[0][0]']

 flatten_5 (Flatten)            (None, 50)           0            ['embedding_5[0][0]']

 dense (Dense)                  (None, 16)           32           ['input_7[0][0]']

 concatenate (Concatenate)      (None, 32124)        0            ['flatten[0][0]',
                                                                   'flatten_1[0][0]',
                                                                   'flatten_2[0][0]',
                                                                   'flatten_3[0][0]',
                                                                   'flatten_4[0][0]',
                                                                   'flatten_5[0][0]',
                                                                   'dense[0][0]']

 dense_1 (Dense)                (None, 128)          4112000      ['concatenate[0][0]']

 dropout (Dropout)              (None, 128)          0            ['dense_1[0][0]']

 dense_2 (Dense)                (None, 64)           8256         ['dropout[0][0]']

 dropout_1 (Dropout)            (None, 64)           0            ['dense_2[0][0]']

 batch_normalization (BatchNorm  (None, 64)          256          ['dropout_1[0][0]']
 alization)

 dense_3 (Dense)                (None, 32)           2080         ['batch_normalization
[0][0]']

 dropout_2 (Dropout)            (None, 32)           0            ['dense_3[0][0]']
 dense_4 (Dense)                (None, 2)            66          ['dropout_2[0][0]']



==================================================================================================
Total params: 19,268,787
Trainable params: 4,365,259
Non-trainable params: 14,903,528
__________________________________________________________________________________________________
None
```

In [41]:

```python
tf.keras.utils.plot_model(
    model1, to_file='model_1.png', show_shapes=False, show_layer_names=True,
    rankdir='TB', expand_nested=False, dpi=96
)
```

Out[41]:



In [42]:

```python
tf.keras.backend.clear_session()
!rm -rf ./logs/
```

In [43]:

```python
%load_ext tensorboard
import datetime, os
from tensorflow.keras.callbacks import ModelCheckpoint
filepath="weights_copy.best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc', verbose=1, save_best_only=True
log_dir = os.path.join("logs",'fits', datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,w
model1.fit(x = [padded_train, school_state_train, teacher_prefix_train, project_grade_ca
            clean_categories_train, clean_subcategories_train, remaining_input_train], y
           batch_size=256,
          validation_data = ([padded_cv, school_state_cv, teacher_prefix_cv, project_gra
            clean_categories_cv, clean_subcategories_cv, remaining_input_cv] , y_cv_code
           callbacks = [checkpoint, tensorboard_callback])
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Epoch 1/20
308/308 [==============================] - ETA: 0s - loss: 0.6974 - auc: 0.5183
Epoch 1: val_auc improved from -inf to 0.44269, saving model to weights_copy.best.hdf5
308/308 [==============================] - 18s 52ms/step - loss: 0.6974 - auc: 0.5183 -
val_loss: 0.4936 - val_auc: 0.4427
Epoch 2/20
308/308 [==============================] - ETA: 0s - loss: 0.5302 - auc: 0.5245
Epoch 2: val_auc did not improve from 0.44269
308/308 [==============================] - 15s 47ms/step - loss: 0.5302 - auc: 0.5245 -
val_loss: 0.5187 - val_auc: 0.4389
Epoch 3/20
308/308 [==============================] - ETA: 0s - loss: 0.4813 - auc: 0.5682
Epoch 3: val_auc improved from 0.44269 to 0.67520, saving model to weights_copy.best.hdf
5
308/308 [==============================] - 15s 49ms/step - loss: 0.4813 - auc: 0.5682 -
val_loss: 0.4757 - val_auc: 0.6752
Epoch 4/20
307/308 [============================>.] - ETA: 0s - loss: 0.4771 - auc: 0.5204
Epoch 4: val_auc did not improve from 0.67520
308/308 [==============================] - 14s 47ms/step - loss: 0.4770 - auc: 0.5209 -
val_loss: 0.4787 - val_auc: 0.6245
Epoch 5/20
307/308 [============================>.] - ETA: 0s - loss: 0.4457 - auc: 0.6084
Epoch 5: val_auc improved from 0.67520 to 0.71764, saving model to weights_copy.best.hdf
5
308/308 [==============================] - 15s 48ms/step - loss: 0.4456 - auc: 0.6086 -
val_loss: 0.4698 - val_auc: 0.7176
Epoch 6/20
307/308 [============================>.] - ETA: 0s - loss: 0.4233 - auc: 0.6882
Epoch 6: val_auc improved from 0.71764 to 0.72950, saving model to weights_copy.best.hdf
5
308/308 [==============================] - 15s 48ms/step - loss: 0.4233 - auc: 0.6882 -
val_loss: 0.4559 - val_auc: 0.7295
Epoch 7/20
307/308 [============================>.] - ETA: 0s - loss: 0.4134 - auc: 0.7093
Epoch 7: val_auc improved from 0.72950 to 0.74720, saving model to weights_copy.best.hdf
5
308/308 [==============================] - 15s 48ms/step - loss: 0.4135 - auc: 0.7091 -
val_loss: 0.4181 - val_auc: 0.7472
Epoch 8/20
307/308 [============================>.] - ETA: 0s - loss: 0.4043 - auc: 0.7248
Epoch 8: val_auc did not improve from 0.74720
308/308 [==============================] - 14s 47ms/step - loss: 0.4043 - auc: 0.7249 -
val_loss: 0.3998 - val_auc: 0.7465
Epoch 9/20
307/308 [============================>.] - ETA: 0s - loss: 0.4011 - auc: 0.7287
Epoch 9: val_auc improved from 0.74720 to 0.75143, saving model to weights_copy.best.hdf
5
308/308 [==============================] - 15s 48ms/step - loss: 0.4011 - auc: 0.7287 -
val_loss: 0.3993 - val_auc: 0.7514
Epoch 10/20
307/308 [============================>.] - ETA: 0s - loss: 0.3968 - auc: 0.7329
Epoch 10: val_auc improved from 0.75143 to 0.75223, saving model to weights_copy.best.hd
f5
308/308 [==============================] - 15s 48ms/step - loss: 0.3969 - auc: 0.7331 -
val_loss: 0.4017 - val_auc: 0.7522
Epoch 11/20
307/308 [============================>.] - ETA: 0s - loss: 0.3927 - auc: 0.7402
Epoch 11: val_auc improved from 0.75223 to 0.75927, saving model to weights_copy.best.hd
f5
```

```
308/308 [==============================] - 15s 47ms/step - loss: 0.3926 - auc: 0.7403 -
val_loss: 0.4244 - val_auc: 0.7593
Epoch 12/20
307/308 [============================>.] - ETA: 0s - loss: 0.3908 - auc: 0.7433
Epoch 12: val_auc did not improve from 0.75927
308/308 [==============================] - 14s 46ms/step - loss: 0.3910 - auc: 0.7430 -
val_loss: 0.3967 - val_auc: 0.7557
Epoch 13/20
307/308 [============================>.] - ETA: 0s - loss: 0.3881 - auc: 0.7484
Epoch 13: val_auc improved from 0.75927 to 0.76013, saving model to weights_copy.best.hd
f5
308/308 [==============================] - 15s 48ms/step - loss: 0.3881 - auc: 0.7485 -
val_loss: 0.3971 - val_auc: 0.7601
Epoch 14/20
307/308 [============================>.] - ETA: 0s - loss: 0.3870 - auc: 0.7532
Epoch 14: val_auc did not improve from 0.76013
308/308 [==============================] - 14s 46ms/step - loss: 0.3871 - auc: 0.7529 -
val_loss: 0.3852 - val_auc: 0.7593
Epoch 15/20
307/308 [============================>.] - ETA: 0s - loss: 0.3846 - auc: 0.7581
Epoch 15: val_auc improved from 0.76013 to 0.76254, saving model to weights_copy.best.hd
f5
308/308 [==============================] - 15s 48ms/step - loss: 0.3845 - auc: 0.7586 -
val_loss: 0.3898 - val_auc: 0.7625
Epoch 16/20
307/308 [============================>.] - ETA: 0s - loss: 0.3836 - auc: 0.7640
Epoch 16: val_auc improved from 0.76254 to 0.76436, saving model to weights_copy.best.hd
f5
308/308 [==============================] - 15s 48ms/step - loss: 0.3835 - auc: 0.7646 -
val_loss: 0.3922 - val_auc: 0.7644
Epoch 17/20
307/308 [============================>.] - ETA: 0s - loss: 0.3824 - auc: 0.7696
Epoch 17: val_auc improved from 0.76436 to 0.76702, saving model to weights_copy.best.hd
f5
308/308 [==============================] - 15s 48ms/step - loss: 0.3823 - auc: 0.7699 -
val_loss: 0.3917 - val_auc: 0.7670
Epoch 18/20
307/308 [============================>.] - ETA: 0s - loss: 0.3793 - auc: 0.7805
Epoch 18: val_auc did not improve from 0.76702
308/308 [==============================] - 14s 46ms/step - loss: 0.3792 - auc: 0.7807 -
val_loss: 0.3866 - val_auc: 0.7659
Epoch 19/20
307/308 [============================>.] - ETA: 0s - loss: 0.3789 - auc: 0.7873
Epoch 19: val_auc did not improve from 0.76702
308/308 [==============================] - 14s 46ms/step - loss: 0.3788 - auc: 0.7876 -
val_loss: 0.4007 - val_auc: 0.7635
Epoch 20/20
307/308 [============================>.] - ETA: 0s - loss: 0.3774 - auc: 0.7949
Epoch 20: val_auc did not improve from 0.76702
308/308 [==============================] - 14s 46ms/step - loss: 0.3774 - auc: 0.7951 -
val_loss: 0.3921 - val_auc: 0.7630
```

```
Out[43]: <keras.callbacks.History at 0x7f7ae250a7d0>
```

```
In [44]: %tensorboard --logdir logs/fits
```

Observations:

1. The histograms keep varying well as the epochs progress. Therefore, the we can infer that the model is training well.

2. The difference between train auc and validation auc is low therefore there is no overfit.

3. The train and validation loss did converge well for the 20 epochs we ran as can be noticed from the graph.

1.5 Compiling and fititng your model

In [ ]:

```
#define custom auc as metric , do not use tf.keras.metrics
# https://stackoverflow.com/a/46844409 - custom AUC reference 1
# https://www.kaggle.com/c/santander-customer-transaction-prediction/discussion/80807  -
# compile and fit your model
```

In [45]:

```
#input 1
input1 = Input(shape=(250,))
embed1 = Embedding(input_dim = 49678, output_dim = 300, weights = [weight_matrix], train
dropout_layer = SpatialDropout1D(0.3)(embed1)
lstm_layer = CuDNNLSTM(128,return_sequences=True)(dropout_layer)
flat1 = Flatten()(lstm_layer)

cat_vars = ["teacher_prefix","school_state","project_grade_category","clean_categories",
cat_sizes = {}
cat_embsizes = {}
for cat in cat_vars:
    cat_sizes[cat] = x_train[cat].nunique()
    cat_embsizes[cat] = min(50, cat_sizes[cat]//2+1)

#input 2
input2 = Input(shape = (1,))
embed2 = Embedding(input_dim=cat_sizes['school_state']+1, output_dim=cat_embsizes['schoo
flat2 = Flatten()(embed2)

#input 3
input3 = Input(shape = (1,))
embed3 = Embedding(input_dim=cat_sizes['teacher_prefix']+1, output_dim=cat_embsizes['tea
flat3 = Flatten()(embed3)

#input 4
input4 = Input(shape = (1,))
embed4 = Embedding(input_dim=cat_sizes['project_grade_category']+1, output_dim=cat_embsi
flat4 = Flatten()(embed4)

#input 5
input5 = Input(shape = (1,))
embed5 = Embedding(input_dim=cat_sizes['clean_categories']+1, output_dim=cat_embsizes['c
flat5 = Flatten()(embed5)

#input 6
input6 = Input(shape = (1,))
embed6 = Embedding(input_dim=cat_sizes['clean_subcategories']+1, output_dim=cat_embsizes
flat6 = Flatten()(embed6)

#input 7
input7 = Input(shape=(1,))
dense1 = Dense(16,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
concat = Concatenate()([flat1, flat2, flat3, flat4, flat5, flat6, dense1])


dense2 = Dense(128,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulari
dropout1 = Dropout(0.5)(dense2)
dense3 = Dense(64,activation='relu', kernel_initializer = HeNormal(), kernel_regularizer
dropout2 = Dropout(0.5)(dense3)
batch_norm = BatchNormalization()(dropout2)
dense4 = Dense(32,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout3 = Dropout(0.5)(dense4)
```

```
output = Dense(2, activation = 'softmax')(dropout3)


model1 = Model([input1, input2, input3, input4, input5, input6, input7], output)
model1.load_weights(filepath)
model1.compile(loss = 'categorical_crossentropy', optimizer = Adam(learning_rate = 0.000

print(model1.summary())
```

```
Model: "model"
__________________________________________________________________________________________
__________
 Layer (type)                   Output Shape          Param #     Connected to

==========================================================================================
==========
 input_1 (InputLayer)           [(None, 250)]         0           []


 embedding (Embedding)          (None, 250, 300)      14903400    ['input_1[0][0]']


 spatial_dropout1d (SpatialDrop  (None, 250, 300)     0           ['embedding[0][0]']

 out1D)


 input_2 (InputLayer)           [(None, 1)]           0           []


 input_3 (InputLayer)           [(None, 1)]           0           []


 input_4 (InputLayer)           [(None, 1)]           0           []


 input_5 (InputLayer)           [(None, 1)]           0           []


 input_6 (InputLayer)           [(None, 1)]           0           []


 cu_dnnlstm (CuDNNLSTM)         (None, 250, 128)      220160      ['spatial_dropout1d[0]
[0]']

 embedding_1 (Embedding)        (None, 1, 26)         1352        ['input_2[0][0]']


 embedding_2 (Embedding)        (None, 1, 3)          18          ['input_3[0][0]']


 embedding_3 (Embedding)        (None, 1, 3)          15          ['input_4[0][0]']
```

```
embedding_4 (Embedding)         (None, 1, 26)        1352         ['input_5[0][0]']

embedding_5 (Embedding)         (None, 1, 50)        19800        ['input_6[0][0]']

input_7 (InputLayer)            [(None, 1)]          0            []

flatten (Flatten)               (None, 32000)        0            ['cu_dnnlstm[0][0]']

flatten_1 (Flatten)             (None, 26)           0            ['embedding_1[0][0]']

flatten_2 (Flatten)             (None, 3)            0            ['embedding_2[0][0]']

flatten_3 (Flatten)             (None, 3)            0            ['embedding_3[0][0]']

flatten_4 (Flatten)             (None, 26)           0            ['embedding_4[0][0]']

flatten_5 (Flatten)             (None, 50)           0            ['embedding_5[0][0]']

dense (Dense)                   (None, 16)           32           ['input_7[0][0]']

concatenate (Concatenate)       (None, 32124)        0            ['flatten[0][0]',
                                                                   'flatten_1[0][0]',
                                                                   'flatten_2[0][0]',
                                                                   'flatten_3[0][0]',
                                                                   'flatten_4[0][0]',
                                                                   'flatten_5[0][0]',
                                                                   'dense[0][0]']

dense_1 (Dense)                 (None, 128)          4112000      ['concatenate[0][0]']

dropout (Dropout)               (None, 128)          0            ['dense_1[0][0]']

dense_2 (Dense)                 (None, 64)           8256         ['dropout[0][0]']
```

```
 dropout_1 (Dropout)            (None, 64)           0           ['dense_2[0][0]']


 batch_normalization (BatchNorm  (None, 64)          256         ['dropout_1[0][0]']

 alization)


 dense_3 (Dense)                (None, 32)           2080        ['batch_normalization
[0][0]']


 dropout_2 (Dropout)            (None, 32)           0           ['dense_3[0][0]']


 dense_4 (Dense)                (None, 2)            66          ['dropout_2[0][0]']


==========================================================================================
==========
Total params: 19,268,787
Trainable params: 4,365,259
Non-trainable params: 14,903,528
__________________________________________________________________________________________
__________
None
```

In [55]:

```
train_pred = model1.predict([padded_train, school_state_train, teacher_prefix_train, pro
            clean_categories_train, clean_subcategories_train, remaining_input_train])

cv_pred = model1.predict([padded_cv, school_state_cv, teacher_prefix_cv, project_grade_c
            clean_categories_cv, clean_subcategories_cv, remaining_input_cv])

test_pred = model1.predict([padded_test, school_state_test, teacher_prefix_test, project
            clean_categories_test, clean_subcategories_test, remaining_input_test])
```

In [58]:

```
print("Train AUC: ",roc_auc_score(y_train, train_pred[:,1]))

print("CV AUC: ",roc_auc_score(y_cv, cv_pred[:,1]))

print("Test AUC: ",roc_auc_score(y_test, test_pred[:,1]))
```

```
Train AUC:  0.8110597746003532
CV AUC:  0.7673032810265807
Test AUC:  0.7701849339197511
```

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. Fit TF-IDF vectorizer on the Train data

2. Get the idf value for each word we have in the train data. Please go through
this
```

```
3. 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.
Hint - A preferable IDF range is 2-11 for model 2.

4.Remove the low idf value and high idf value words from the train and test
data. You can go through each of the
sentence of train and test data and include only those features(words) which
are present in the defined IDF range.
5. Perform tokenization on the modified text data same as you have done for
previous model.
6. Create embedding matrix for model 2 and then use the rest of the features
similar to previous model.
7. Define the model, compile and fit the model.
```

In [60]:

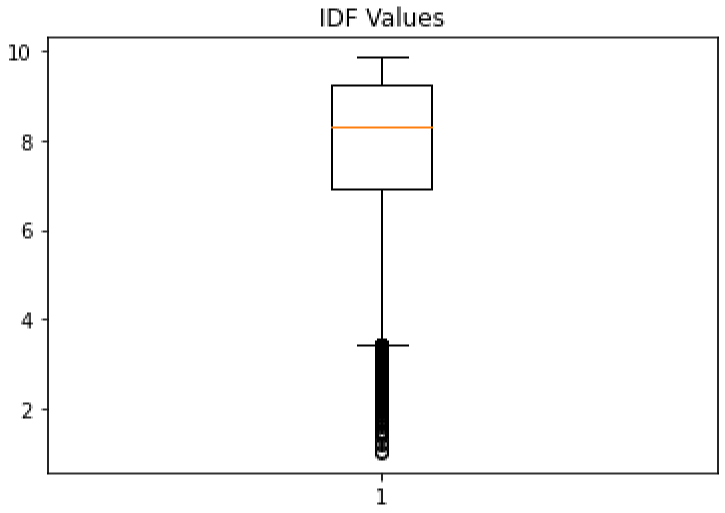
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(min_df=10) #Defining TFIDF with min_df=10
tfidf_vals = vectorizer.fit(x_train['essay'])
```

In [62]:

```
idf_vals = vectorizer.idf_
```

In [64]:

```
import matplotlib.pyplot as plt

plt.boxplot(idf_vals)
plt.title("IDF Values")
plt.show()
```



In [66]:

```
print("25th percentile value: ",np.quantile(idf_vals, 0.25))
print("75th percentile value: ",np.quantile(idf_vals, 0.75))
```

```
25th percentile value:  6.945001164173634
75th percentile value:  9.277145059409225
```

In [67]:

```
print("10th percentile value: ",np.quantile(idf_vals, 0.1))
print("90th percentile value: ",np.quantile(idf_vals, 0.9))
```

```
10th percentile value:  5.435437279661449
90th percentile value:  9.70792797550168
```

In [68]:

```
print("5th percentile value: ",np.quantile(idf_vals, 0.05))
print("95th percentile value: ",np.quantile(idf_vals, 0.95))
```

```
5th percentile value:  4.559494781292484
```

```
95th percentile value:  9.787970683175216
```

The idf range selected here is 4.5-9.7 since 5th percentile to 95th percentile values.

In [74]:

```
vocab_idf = zip(tfidf_vals.get_feature_names(),idf_vals)

vocabulary = []
for i,j in vocab_idf:

    if j >= 4.5 and j <= 9.7:
        vocabulary.append(i)

print(len(vocabulary))
```

```
12393
```

In [76]:

```
def imp_words_alone(essay_text):
  '''
  Method to retain only words between 5th and 95th percentile
  '''
  imp_text = []
  for sent in essay_text:
    words = sent.split()
    final_sent = ''
    for word in words:
      if(word in vocabulary):
        final_sent += ' ' + word

    imp_text.append(final_sent)

  return imp_text
```

In [81]:

```
#getting imp words alone for each essay in train, cv and test.
train_idf = imp_words_alone(x_train['essay'])
cv_idf = imp_words_alone(x_cv['essay'])
test_idf = imp_words_alone(x_test['essay'])
```

In [82]:

```
with open('train_idf.pkl', 'wb') as f:
  pickle.dump(train_idf, f)
```

In [83]:

```
with open('test_idf.pkl', 'wb') as f:
  pickle.dump(test_idf, f)
```

In [84]:

```
with open('cv_idf.pkl', 'wb') as f:
  pickle.dump(cv_idf, f)
```

In [85]:

```
my_tokenizer1 = Tokenizer()

my_tokenizer1.fit_on_texts(train_idf)
sequences_train1 = my_tokenizer.texts_to_sequences(train_idf)
padded_train1 = pad_sequences(sequences_train, padding = 'post', truncating = 'pre', max

sequences_cv1 = my_tokenizer1.texts_to_sequences(cv_idf)
padded_cv1 = pad_sequences(sequences_cv, padding = 'post', truncating = 'pre', maxlen =

sequences_test1 = my_tokenizer1.texts_to_sequences(test_idf)
padded_test1 = pad_sequences(sequences_test, padding = 'post', truncating = 'pre', maxle
```

In [90]:

```
tokenz1 = my_tokenizer1.word_index
len(tokenz1)
```

Out[90]:

```
12393
```

In [91]:

```
vocab_size1 = len(tokenz1)+ 1
weight_matrix1 = np.zeros((vocab_size1, 300))
for word,i in tokenz1.items():
    if word in glove_vec:
        weight_matrix1[i] = glove_vec[word]
```

In [92]:

```
len(weight_matrix1)
```

Out[92]:

```
12394
```

In [104…

```
#input 1
input1 = Input(shape=(250,))
embed1 = Embedding(input_dim = 12394, output_dim = 300, weights = [weight_matrix1], trai
dropout_layer = SpatialDropout1D(0.3)(embed1)
lstm_layer = CuDNNLSTM(128,return_sequences=True)(dropout_layer)
flat1 = Flatten()(lstm_layer)

cat_vars = ["teacher_prefix","school_state","project_grade_category","clean_categories",
cat_sizes = {}
cat_embsizes = {}
for cat in cat_vars:
    cat_sizes[cat] = x_train[cat].nunique()
    cat_embsizes[cat] = min(50, cat_sizes[cat]//2+1)

#input 2
input2 = Input(shape = (1,))
embed2 = Embedding(input_dim=cat_sizes['school_state']+1, output_dim=cat_embsizes['schoo
flat2 = Flatten()(embed2)

#input 3
input3 = Input(shape = (1,))
embed3 = Embedding(input_dim=cat_sizes['teacher_prefix']+1, output_dim=cat_embsizes['tea
flat3 = Flatten()(embed3)

#input 4
input4 = Input(shape = (1,))
embed4 = Embedding(input_dim=cat_sizes['project_grade_category']+1, output_dim=cat_embsi
flat4 = Flatten()(embed4)

#input 5
input5 = Input(shape = (1,))
embed5 = Embedding(input_dim=cat_sizes['clean_categories']+1, output_dim=cat_embsizes['c
flat5 = Flatten()(embed5)

#input 6
input6 = Input(shape = (1,))
embed6 = Embedding(input_dim=cat_sizes['clean_subcategories']+1, output_dim=cat_embsizes
flat6 = Flatten()(embed6)

#input 7
input7 = Input(shape=(1,))
dense1 = Dense(16,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
concat = Concatenate()([flat1, flat2, flat3, flat4, flat5, flat6, dense1])


dense2 = Dense(128,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulari
dropout1 = Dropout(0.5)(dense2)
dense3 = Dense(64,activation='relu', kernel_initializer = HeNormal(), kernel_regularizer
dropout2 = Dropout(0.5)(dense3)
batch_norm = BatchNormalization()(dropout2)
dense4 = Dense(32,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout3 = Dropout(0.5)(dense4)
output = Dense(2, activation = 'softmax')(dropout3)
```

```
model2 = Model([input1, input2, input3, input4, input5, input6, input7], output)
model2.compile(loss = 'categorical_crossentropy', optimizer = Adam(learning_rate = 0.000

print(model2.summary())
```

```
Model: "model"
__________________________________________________________________________________________
__________
 Layer (type)                   Output Shape         Param #     Connected to

==========================================================================================
==========
 input_1 (InputLayer)           [(None, 250)]        0           []


 embedding (Embedding)          (None, 250, 300)     3718200     ['input_1[0][0]']


 spatial_dropout1d (SpatialDrop  (None, 250, 300)    0           ['embedding[0][0]']

 out1D)


 input_2 (InputLayer)           [(None, 1)]          0           []


 input_3 (InputLayer)           [(None, 1)]          0           []


 input_4 (InputLayer)           [(None, 1)]          0           []


 input_5 (InputLayer)           [(None, 1)]          0           []


 input_6 (InputLayer)           [(None, 1)]          0           []


 cu_dnnlstm (CuDNNLSTM)         (None, 250, 128)     220160      ['spatial_dropout1d[0]
[0]']

 embedding_1 (Embedding)        (None, 1, 26)        1352        ['input_2[0][0]']


 embedding_2 (Embedding)        (None, 1, 3)         18          ['input_3[0][0]']


 embedding_3 (Embedding)        (None, 1, 3)         15          ['input_4[0][0]']


 embedding_4 (Embedding)        (None, 1, 26)        1352        ['input_5[0][0]']
```

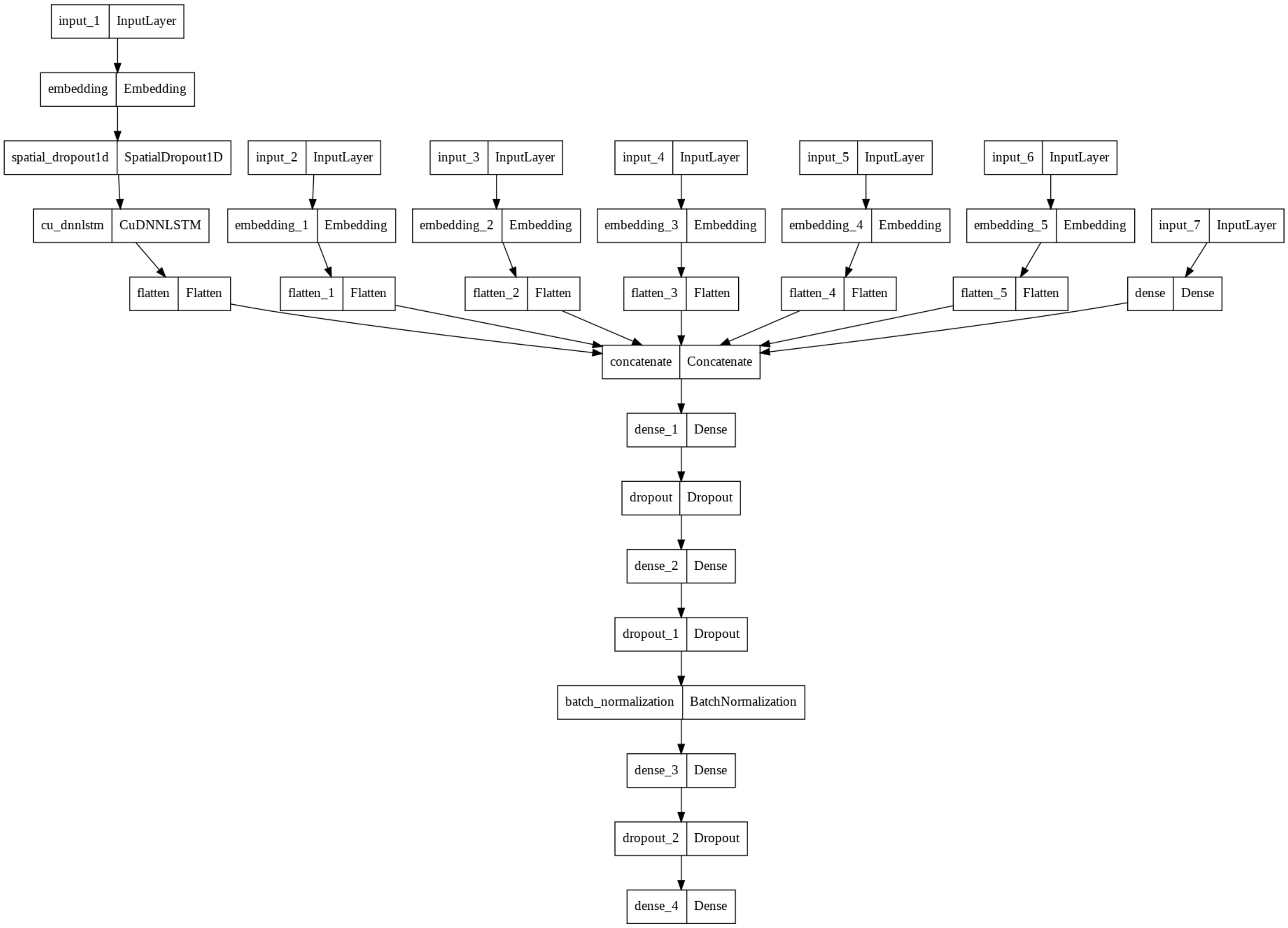
```
embedding_5 (Embedding)        (None, 1, 50)        19800        ['input_6[0][0]']

input_7 (InputLayer)           [(None, 1)]          0            []

flatten (Flatten)              (None, 32000)        0            ['cu_dnnlstm[0][0]']

flatten_1 (Flatten)            (None, 26)           0            ['embedding_1[0][0]']

flatten_2 (Flatten)            (None, 3)            0            ['embedding_2[0][0]']

flatten_3 (Flatten)            (None, 3)            0            ['embedding_3[0][0]']

flatten_4 (Flatten)            (None, 26)           0            ['embedding_4[0][0]']

flatten_5 (Flatten)            (None, 50)           0            ['embedding_5[0][0]']

dense (Dense)                  (None, 16)           32           ['input_7[0][0]']

concatenate (Concatenate)      (None, 32124)        0            ['flatten[0][0]',
                                                                  'flatten_1[0][0]',
                                                                  'flatten_2[0][0]',
                                                                  'flatten_3[0][0]',
                                                                  'flatten_4[0][0]',
                                                                  'flatten_5[0][0]',
                                                                  'dense[0][0]']

dense_1 (Dense)                (None, 128)          4112000      ['concatenate[0][0]']

dropout (Dropout)              (None, 128)          0            ['dense_1[0][0]']

dense_2 (Dense)                (None, 64)           8256         ['dropout[0][0]']

dropout_1 (Dropout)            (None, 64)           0            ['dense_2[0][0]']
```

```
 batch_normalization (BatchNorm  (None, 64)          256         ['dropout_1[0][0]']
 alization)



 dense_3 (Dense)                (None, 32)          2080        ['batch_normalization
[0][0]']


 dropout_2 (Dropout)            (None, 32)          0           ['dense_3[0][0]']



 dense_4 (Dense)                (None, 2)           66          ['dropout_2[0][0]']



==========================================================================================
==========
Total params: 8,083,587
Trainable params: 4,365,259
Non-trainable params: 3,718,328
__________________________________________________________________________________________
__________
None
```

In [105…

```python
tf.keras.utils.plot_model(
    model2, to_file='model_2.png', show_shapes=False, show_layer_names=True,
    rankdir='TB', expand_nested=False, dpi=96
)
```

Out[105]:



In [106…

```
tf.keras.backend.clear_session()
!rm -rf ./logs/
```

In [107…

```
%load_ext tensorboard
import datetime, os
from tensorflow.keras.callbacks import ModelCheckpoint
filepath="weights_copy1.best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc', verbose=1, save_best_only=True
log_dir = os.path.join("logs",'fits', datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,w
model2.fit(x = [padded_train1, school_state_train, teacher_prefix_train, project_grade_c
            clean_categories_train, clean_subcategories_train, remaining_input_train], y
           epochs = 20, verbose=1,
           batch_size=256,
          validation_data = ([padded_cv1, school_state_cv, teacher_prefix_cv, project_gr
            clean_categories_cv, clean_subcategories_cv, remaining_input_cv] , y_cv_code
           callbacks = [checkpoint, tensorboard_callback])
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Epoch 1/20
307/308 [============================>.] - ETA: 0s - loss: 0.7209 - auc: 0.5100
Epoch 1: val_auc improved from -inf to 0.54525, saving model to weights_copy1.best.hdf5
308/308 [==============================] - 17s 46ms/step - loss: 0.7208 - auc: 0.5092 -
val_loss: 0.4899 - val_auc: 0.5453
Epoch 2/20
307/308 [============================>.] - ETA: 0s - loss: 0.5347 - auc: 0.5116
Epoch 2: val_auc did not improve from 0.54525
308/308 [==============================] - 13s 44ms/step - loss: 0.5346 - auc: 0.5116 -
val_loss: 0.4831 - val_auc: 0.4477
Epoch 3/20
307/308 [============================>.] - ETA: 0s - loss: 0.4950 - auc: 0.5133
Epoch 3: val_auc did not improve from 0.54525
308/308 [==============================] - 13s 44ms/step - loss: 0.4949 - auc: 0.5140 -
val_loss: 0.4735 - val_auc: 0.4329
Epoch 4/20
307/308 [============================>.] - ETA: 0s - loss: 0.4733 - auc: 0.5228
Epoch 4: val_auc did not improve from 0.54525
308/308 [==============================] - 14s 44ms/step - loss: 0.4734 - auc: 0.5229 -
val_loss: 0.4613 - val_auc: 0.5241
Epoch 5/20
307/308 [============================>.] - ETA: 0s - loss: 0.4618 - auc: 0.5385
Epoch 5: val_auc improved from 0.54525 to 0.62222, saving model to weights_copy1.best.hd
f5
308/308 [==============================] - 14s 45ms/step - loss: 0.4618 - auc: 0.5381 -
val_loss: 0.4674 - val_auc: 0.6222
Epoch 6/20
308/308 [==============================] - ETA: 0s - loss: 0.4511 - auc: 0.5672
Epoch 6: val_auc improved from 0.62222 to 0.65134, saving model to weights_copy1.best.hd
f5
308/308 [==============================] - 14s 46ms/step - loss: 0.4511 - auc: 0.5672 -
val_loss: 0.4447 - val_auc: 0.6513
Epoch 7/20
307/308 [============================>.] - ETA: 0s - loss: 0.4430 - auc: 0.5990
Epoch 7: val_auc improved from 0.65134 to 0.66505, saving model to weights_copy1.best.hd
f5
308/308 [==============================] - 14s 45ms/step - loss: 0.4430 - auc: 0.5990 -
val_loss: 0.4341 - val_auc: 0.6650
Epoch 8/20
307/308 [============================>.] - ETA: 0s - loss: 0.4338 - auc: 0.6306
Epoch 8: val_auc improved from 0.66505 to 0.68433, saving model to weights_copy1.best.hd
f5
308/308 [==============================] - 14s 45ms/step - loss: 0.4339 - auc: 0.6309 -
val_loss: 0.4357 - val_auc: 0.6843
Epoch 9/20
```

```
307/308 [============================>.] - ETA: 0s - loss: 0.4297 - auc: 0.6479
Epoch 9: val_auc improved from 0.68433 to 0.69657, saving model to weights_copy1.best.hd
f5
308/308 [==============================] - 14s 45ms/step - loss: 0.4297 - auc: 0.6475 -
val_loss: 0.4369 - val_auc: 0.6966
Epoch 10/20
307/308 [============================>.] - ETA: 0s - loss: 0.4242 - auc: 0.6736
Epoch 10: val_auc improved from 0.69657 to 0.71000, saving model to weights_copy1.best.h
df5
308/308 [==============================] - 14s 45ms/step - loss: 0.4242 - auc: 0.6739 -
val_loss: 0.4238 - val_auc: 0.7100
Epoch 11/20
307/308 [============================>.] - ETA: 0s - loss: 0.4214 - auc: 0.6939
Epoch 11: val_auc improved from 0.71000 to 0.71925, saving model to weights_copy1.best.h
df5
308/308 [==============================] - 14s 45ms/step - loss: 0.4214 - auc: 0.6939 -
val_loss: 0.4299 - val_auc: 0.7193
Epoch 12/20
307/308 [============================>.] - ETA: 0s - loss: 0.4199 - auc: 0.7146
Epoch 12: val_auc improved from 0.71925 to 0.72247, saving model to weights_copy1.best.h
df5
308/308 [==============================] - 14s 45ms/step - loss: 0.4199 - auc: 0.7144 -
val_loss: 0.4423 - val_auc: 0.7225
Epoch 13/20
307/308 [============================>.] - ETA: 0s - loss: 0.4211 - auc: 0.7334
Epoch 13: val_auc improved from 0.72247 to 0.72572, saving model to weights_copy1.best.h
df5
308/308 [==============================] - 14s 45ms/step - loss: 0.4210 - auc: 0.7333 -
val_loss: 0.4249 - val_auc: 0.7257
Epoch 14/20
307/308 [============================>.] - ETA: 0s - loss: 0.4219 - auc: 0.7528
Epoch 14: val_auc did not improve from 0.72572
308/308 [==============================] - 13s 44ms/step - loss: 0.4220 - auc: 0.7531 -
val_loss: 0.4404 - val_auc: 0.7228
Epoch 15/20
307/308 [============================>.] - ETA: 0s - loss: 0.4224 - auc: 0.7750
Epoch 15: val_auc did not improve from 0.72572
308/308 [==============================] - 13s 44ms/step - loss: 0.4223 - auc: 0.7752 -
val_loss: 0.4483 - val_auc: 0.7236
Epoch 16/20
307/308 [============================>.] - ETA: 0s - loss: 0.4204 - auc: 0.8023
Epoch 16: val_auc did not improve from 0.72572
308/308 [==============================] - 14s 44ms/step - loss: 0.4205 - auc: 0.8017 -
val_loss: 0.4756 - val_auc: 0.7130
Epoch 17/20
307/308 [============================>.] - ETA: 0s - loss: 0.4185 - auc: 0.8279
Epoch 17: val_auc did not improve from 0.72572
308/308 [==============================] - 13s 44ms/step - loss: 0.4184 - auc: 0.8280 -
val_loss: 0.4992 - val_auc: 0.7069
Epoch 18/20
308/308 [==============================] - ETA: 0s - loss: 0.4152 - auc: 0.8581
Epoch 18: val_auc did not improve from 0.72572
308/308 [==============================] - 14s 44ms/step - loss: 0.4152 - auc: 0.8581 -
val_loss: 0.5305 - val_auc: 0.7018
Epoch 19/20
307/308 [============================>.] - ETA: 0s - loss: 0.4109 - auc: 0.8876
Epoch 19: val_auc did not improve from 0.72572
308/308 [==============================] - 13s 43ms/step - loss: 0.4109 - auc: 0.8876 -
val_loss: 0.5711 - val_auc: 0.6942
Epoch 20/20
307/308 [============================>.] - ETA: 0s - loss: 0.4052 - auc: 0.9104
Epoch 20: val_auc did not improve from 0.72572
308/308 [==============================] - 14s 44ms/step - loss: 0.4052 - auc: 0.9103 -
val_loss: 0.6105 - val_auc: 0.6886
```

```
Out[107]: <keras.callbacks.History at 0x7f7a36590510>
```

In [108…

```
%tensorboard --logdir logs/fits
```

Observations:

1. The model is starting to overfit as the epochs progress as can be seen from the epoch vs loss graph for both train and validation data.
2. The overfit can be reduced by experimenting by increasing dropout rate of dropout layers.

In [109…

```
#input 1
input1 = Input(shape=(250,))
embed1 = Embedding(input_dim = 12394, output_dim = 300, weights = [weight_matrix1], trai
dropout_layer = SpatialDropout1D(0.3)(embed1)
lstm_layer = CuDNNLSTM(128,return_sequences=True)(dropout_layer)
flat1 = Flatten()(lstm_layer)

cat_vars = ["teacher_prefix","school_state","project_grade_category","clean_categories",
cat_sizes = {}
cat_embsizes = {}
for cat in cat_vars:
    cat_sizes[cat] = x_train[cat].nunique()
    cat_embsizes[cat] = min(50, cat_sizes[cat]//2+1)

#input 2
input2 = Input(shape = (1,))
embed2 = Embedding(input_dim=cat_sizes['school_state']+1, output_dim=cat_embsizes['schoo
flat2 = Flatten()(embed2)

#input 3
input3 = Input(shape = (1,))
embed3 = Embedding(input_dim=cat_sizes['teacher_prefix']+1, output_dim=cat_embsizes['tea
flat3 = Flatten()(embed3)

#input 4
input4 = Input(shape = (1,))
embed4 = Embedding(input_dim=cat_sizes['project_grade_category']+1, output_dim=cat_embsi
flat4 = Flatten()(embed4)

#input 5
input5 = Input(shape = (1,))
embed5 = Embedding(input_dim=cat_sizes['clean_categories']+1, output_dim=cat_embsizes['c
flat5 = Flatten()(embed5)

#input 6
input6 = Input(shape = (1,))
embed6 = Embedding(input_dim=cat_sizes['clean_subcategories']+1, output_dim=cat_embsizes
flat6 = Flatten()(embed6)

#input 7
input7 = Input(shape=(1,))
dense1 = Dense(16,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
concat = Concatenate()([flat1, flat2, flat3, flat4, flat5, flat6, dense1])


dense2 = Dense(128,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulari
dropout1 = Dropout(0.5)(dense2)
dense3 = Dense(64,activation='relu', kernel_initializer = HeNormal(), kernel_regularizer
dropout2 = Dropout(0.5)(dense3)
batch_norm = BatchNormalization()(dropout2)
dense4 = Dense(32,activation = 'relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout3 = Dropout(0.5)(dense4)
```

```
output = Dense(2, activation = 'softmax')(dropout3)


model2 = Model([input1, input2, input3, input4, input5, input6, input7], output)
model2.load_weights(filepath)
model2.compile(loss = 'categorical_crossentropy', optimizer = Adam(learning_rate = 0.000
```

In [110…

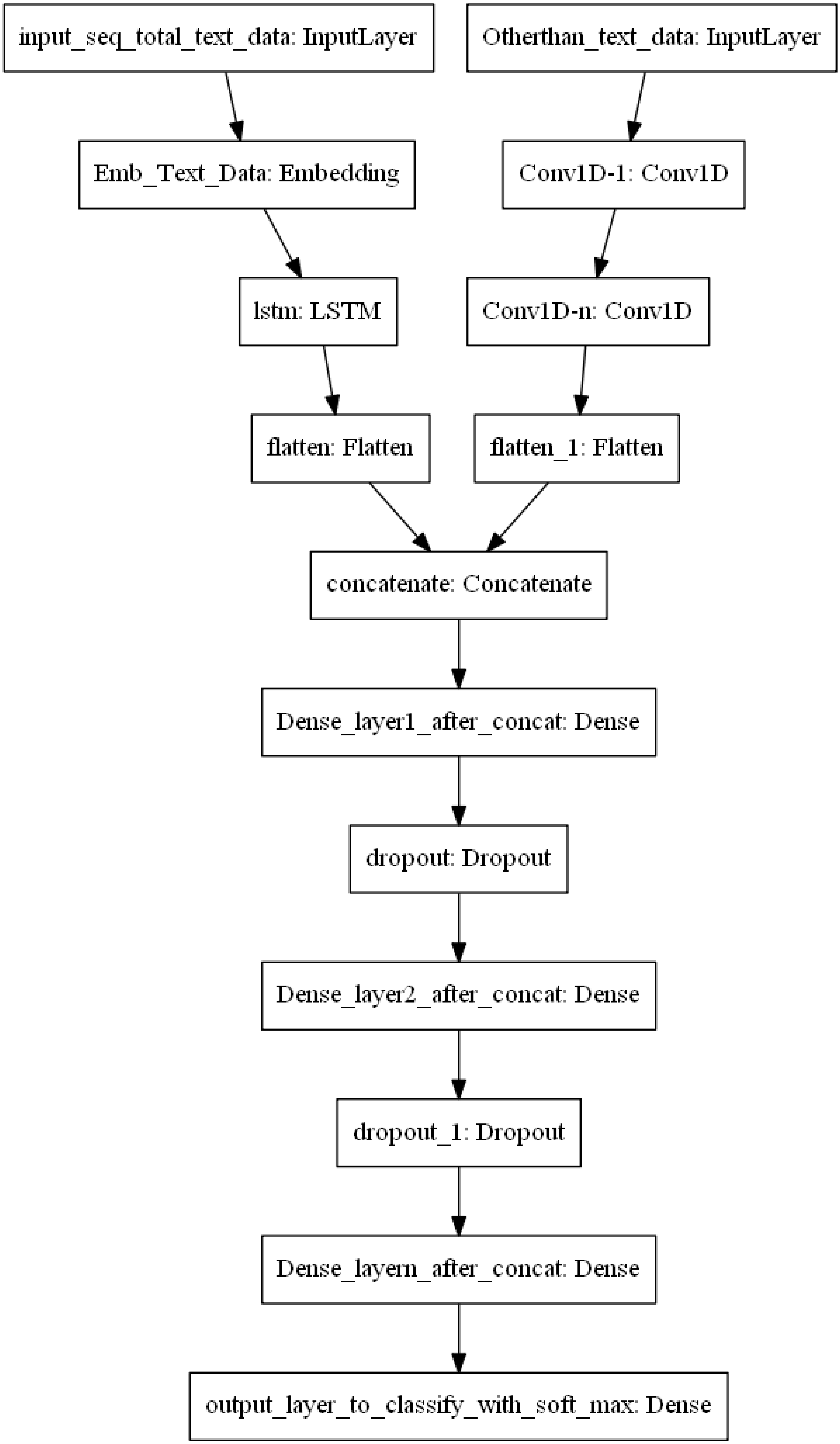
```
train_pred1 = model2.predict([padded_train1, school_state_train, teacher_prefix_train, p
            clean_categories_train, clean_subcategories_train, remaining_input_train])

cv_pred1 = model2.predict([padded_cv1, school_state_cv, teacher_prefix_cv, project_grade
            clean_categories_cv, clean_subcategories_cv, remaining_input_cv])

test_pred1 = model2.predict([padded_test1, school_state_test, teacher_prefix_test, proje
            clean_categories_test, clean_subcategories_test, remaining_input_test])
```

In [111…

```
print("Train AUC: ",roc_auc_score(y_train, train_pred1[:,1]))

print("CV AUC: ",roc_auc_score(y_cv, cv_pred1[:,1]))

print("Test AUC: ",roc_auc_score(y_test, test_pred1[:,1]))
```

```
Train AUC:  0.8021176007551988
CV AUC:  0.7235825524313362
Test AUC:  0.7299088105737789
```

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
output_layer_to_classify_with_soft_max: Dense

ref:

https://i.imgur.com/fkQ8nGo.png

In [ ]:
```
#in this model you can use the text vectorized data from model1
#for other than text data consider the following steps
# you have to perform one hot encoding of categorical features. You can use onehotencode
# Stack up standardised numerical features and all the one hot encoded categorical featu
#the input to conv1d layer is 3d, you can convert your 2d data to 3d using np.newaxis
# Note - deep learning models won't work with sparse features, you have to convert them
```

In [ ]:
```
print("school_state: ",len(enc.categories_[0]))
print("teacher_prefix: ",len(enc1.categories_[0]))
print("project_grade_category: ",len(enc2.categories_[0]))
print("clean_categories: ",len(enc3.categories_[0]))
print("clean_subcategories: ",len(enc4.categories_[0]))
```

In [113…
```
#one hot encoding school_state_col
onehot = OneHotEncoder()
ss_train = onehot.fit_transform(x_train['school_state'].values.reshape(-1,1))
ss_cv = onehot.transform(x_cv['school_state'].values.reshape(-1,1))
ss_test = onehot.transform(x_test['school_state'].values.reshape(-1,1))
```

In [114…
```
#one hot encoding teacher_title
onehot = OneHotEncoder()
tp_train = onehot.fit_transform(x_train['teacher_prefix'].values.reshape(-1,1))
tp_cv = onehot.transform(x_cv['teacher_prefix'].values.reshape(-1,1))
tp_test = onehot.transform(x_test['teacher_prefix'].values.reshape(-1,1))
```

In [115…
```
onehot = OneHotEncoder()
pgc_train = onehot.fit_transform(x_train['project_grade_category'].values.reshape(-1,1))
pgc_cv = onehot.transform(x_cv['project_grade_category'].values.reshape(-1,1))
pgc_test = onehot.transform(x_test['project_grade_category'].values.reshape(-1,1))
```

In [116…
```
onehot = OneHotEncoder()
cc_train = onehot.fit_transform(x_train['clean_categories'].values.reshape(-1,1))
cc_cv = onehot.transform(x_cv['clean_categories'].values.reshape(-1,1))
cc_test = onehot.transform(x_test['clean_categories'].values.reshape(-1,1))
```

In [118…
```
onehot = OneHotEncoder(handle_unknown = 'ignore')
cs_train = onehot.fit_transform(x_train['clean_subcategories'].values.reshape(-1,1))
cs_cv = onehot.transform(x_cv['clean_subcategories'].values.reshape(-1,1))
cs_test = onehot.transform(x_test['clean_subcategories'].values.reshape(-1,1))
```

In [125…
```
from scipy.sparse import hstack

train_non_text = hstack((ss_train, tp_train, pgc_train, cc_train, cs_train, remaining_in
cv_non_text = hstack((ss_cv, tp_cv, pgc_cv, cc_cv, cs_cv, remaining_input_cv)).toarray()
test_non_text = hstack((ss_test, tp_test, pgc_test, cc_test, cs_test, remaining_input_te
```

In [126…
```
print(train_non_text.shape)
print(cv_non_text.shape)
print(test_non_text.shape)
```

```
(78658, 507)
(8740, 507)
(21850, 507)
```

In [129…
```
# input 1
input1 = Input(shape=(250,))
embed1 = Embedding(input_dim = 49678, output_dim = 300, weights = [weight_matrix], train
dropout_layer = SpatialDropout1D(0.3)(embed1)
lstm_layer = CuDNNLSTM(256,return_sequences=True)(dropout_layer)
flat1 = Flatten()(lstm_layer)
```

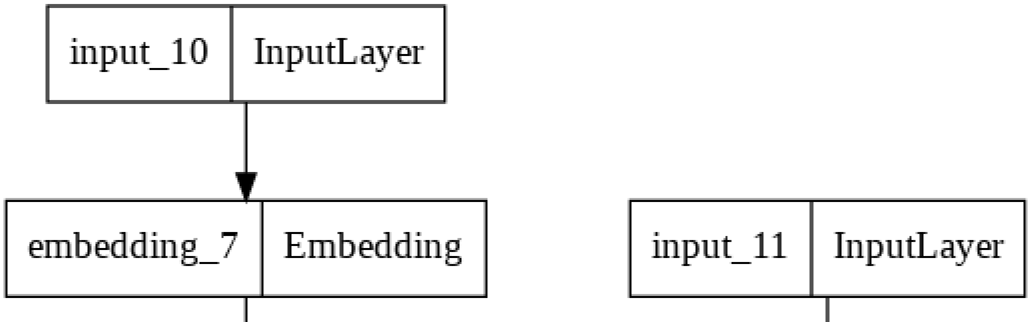
```
# input 2
input2 = Input(shape=(507,1))
conv1 = Conv1D(filters = 64, kernel_size = 3, strides = 1)(input2)
conv2 = Conv1D(filters = 64, kernel_size = 3, strides = 1)(conv1)
flat2 = Flatten()(conv2)

# merging both the inputs
concat = Concatenate()([flat1, flat2])
dense1 = Dense(300, activation = 'relu', kernel_initializer = HeNormal(), kernel_regular
dropout1 = Dropout(0.4)(dense1)
dense2 = Dense(256, activation='relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout2 = Dropout(0.5)(dense2)
batch_norm = BatchNormalization()(dropout2)
dense3 = Dense(128, activation='relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout3 = Dropout(0.6)(dense3)
output = Dense(2, activation = 'softmax')(dropout3)

# create model with two inputs
model3 = Model([input1, input2], output)
model3.compile(loss='categorical_crossentropy', optimizer = Adam(learning_rate = 0.0006)
print(model3.summary())
```

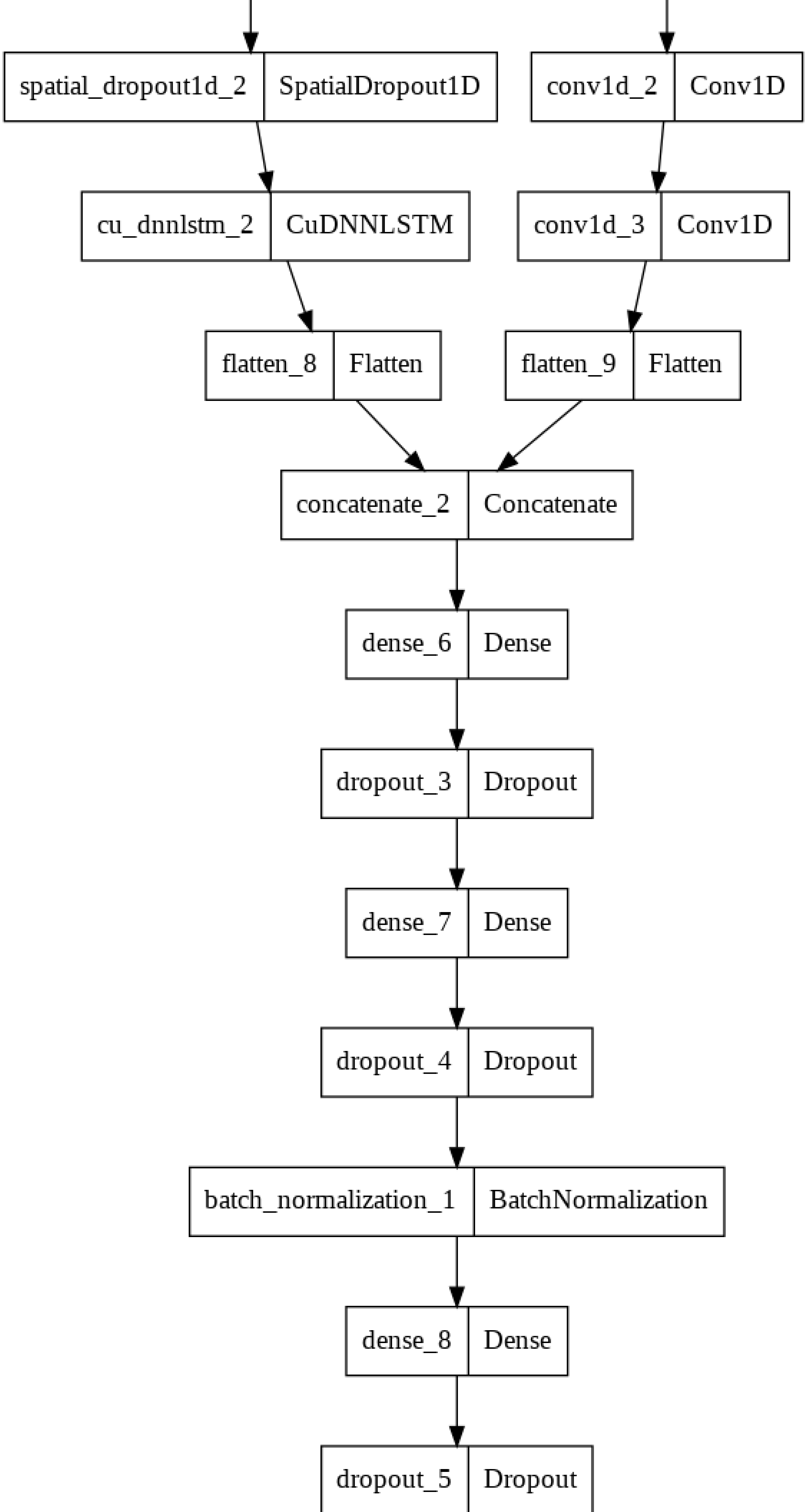
```
Model: "model_1"
__________________________________________________________________________________________
__________
 Layer (type)                   Output Shape         Param #     Connected to

==========================================================================================
==========
 input_10 (InputLayer)          [(None, 250)]        0           []


 embedding_7 (Embedding)        (None, 250, 300)     14903400    ['input_10[0][0]']


 input_11 (InputLayer)          [(None, 507, 1)]     0           []


 spatial_dropout1d_2 (SpatialDr  (None, 250, 300)    0           ['embedding_7[0][0]']

 opout1D)


 conv1d_2 (Conv1D)              (None, 505, 64)      256         ['input_11[0][0]']


 cu_dnnlstm_2 (CuDNNLSTM)       (None, 250, 256)     571392      ['spatial_dropout1d_2
[0][0]']


 conv1d_3 (Conv1D)              (None, 503, 64)      12352       ['conv1d_2[0][0]']


 flatten_8 (Flatten)            (None, 64000)        0           ['cu_dnnlstm_2[0][0]']


 flatten_9 (Flatten)            (None, 32192)        0           ['conv1d_3[0][0]']
 concatenate_2 (Concatenate)    (None, 96192)        0           ['flatten_8[0][0]',
                                                                  'flatten_9[0][0]']


 dense_6 (Dense)                (None, 300)          28857900    ['concatenate_2[0][0]']


 dropout_3 (Dropout)            (None, 300)          0           ['dense_6[0][0]']


 dense_7 (Dense)                (None, 256)          77056       ['dropout_3[0][0]']


 dropout_4 (Dropout)            (None, 256)          0           ['dense_7[0][0]']


 batch_normalization_1 (BatchNo  (None, 256)         1024        ['dropout_4[0][0]']

 rmalization)


 dense_8 (Dense)                (None, 128)          32896       ['batch_normalization_1
[0][0]']

 dropout_5 (Dropout)            (None, 128)          0           ['dense_8[0][0]']


 dense_9 (Dense)                (None, 2)            258         ['dropout_5[0][0]']


========================================================================================
==========
Total params: 44,456,534
Trainable params: 29,552,622
Non-trainable params: 14,903,912
________________________________________________________________________________________
__________
None
```

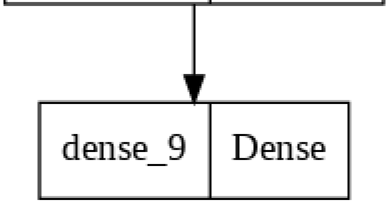
In [130…

```python
tf.keras.utils.plot_model(
    model3, to_file='model_3.png', show_shapes=False, show_layer_names=True,
    rankdir='TB', expand_nested=False, dpi=96
)
```

Out[130]:




spatial_dropout1d_2 | SpatialDropout1D
conv1d_2 | Conv1D
cu_dnnlstm_2 | CuDNNLSTM
conv1d_3 | Conv1D
flatten_8 | Flatten
flatten_9 | Flatten
concatenate_2 | Concatenate
dense_6 | Dense
dropout_3 | Dropout
dense_7 | Dense
dropout_4 | Dropout
batch_normalization_1 | BatchNormalization
dense_8 | Dense
dropout_5 | Dropout



In [131…

```
tf.keras.backend.clear_session()
!rm -rf ./logs/
```

In [132…

```
%load_ext tensorboard
import datetime, os
from tensorflow.keras.callbacks import ModelCheckpoint
filepath="weights_copy2.best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc', verbose=1, save_best_only=True
log_dir = os.path.join("logs",'fits', datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,w
model3.fit(x = [padded_train, train_non_text], y = y_train_coded,
           epochs = 20, verbose=1,
           batch_size=256,
          validation_data = ([padded_cv, cv_non_text] , y_cv_coded),
           callbacks = [checkpoint, tensorboard_callback])
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Epoch 1/20
308/308 [==============================] - ETA: 0s - loss: 0.8100 - auc: 0.5053
Epoch 1: val_auc improved from -inf to 0.57373, saving model to weights_copy2.best.hdf5
308/308 [==============================] - 42s 109ms/step - loss: 0.8100 - auc: 0.5053 -
val_loss: 0.5992 - val_auc: 0.5737
Epoch 2/20
307/308 [============================>.] - ETA: 0s - loss: 0.5960 - auc: 0.5071
Epoch 2: val_auc did not improve from 0.57373
308/308 [==============================] - 31s 99ms/step - loss: 0.5959 - auc: 0.5079 -
val_loss: 0.5903 - val_auc: 0.4211
Epoch 3/20
307/308 [============================>.] - ETA: 0s - loss: 0.5311 - auc: 0.5543
Epoch 3: val_auc did not improve from 0.57373
308/308 [==============================] - 32s 104ms/step - loss: 0.5310 - auc: 0.5543 -
val_loss: 0.6335 - val_auc: 0.5527
Epoch 4/20
307/308 [============================>.] - ETA: 0s - loss: 0.4922 - auc: 0.6511
Epoch 4: val_auc improved from 0.57373 to 0.71733, saving model to weights_copy2.best.hd
f5
308/308 [==============================] - 32s 104ms/step - loss: 0.4921 - auc: 0.6511 -
val_loss: 0.5202 - val_auc: 0.7173
Epoch 5/20
307/308 [============================>.] - ETA: 0s - loss: 0.4598 - auc: 0.7091
Epoch 5: val_auc improved from 0.71733 to 0.74428, saving model to weights_copy2.best.hd
f5
308/308 [==============================] - 33s 106ms/step - loss: 0.4598 - auc: 0.7092 -
val_loss: 0.5124 - val_auc: 0.7443
Epoch 6/20
307/308 [============================>.] - ETA: 0s - loss: 0.4436 - auc: 0.7319
Epoch 6: val_auc improved from 0.74428 to 0.75712, saving model to weights_copy2.best.hd
f5
308/308 [==============================] - 33s 106ms/step - loss: 0.4435 - auc: 0.7324 -
val_loss: 0.5204 - val_auc: 0.7571
Epoch 7/20
307/308 [============================>.] - ETA: 0s - loss: 0.4320 - auc: 0.7428
Epoch 7: val_auc improved from 0.75712 to 0.76104, saving model to weights_copy2.best.hd
f5
308/308 [==============================] - 32s 103ms/step - loss: 0.4320 - auc: 0.7426 -
val_loss: 0.4960 - val_auc: 0.7610
Epoch 8/20
```

```
307/308 [============================>.] - ETA: 0s - loss: 0.4305 - auc: 0.7436
Epoch 8: val_auc improved from 0.76104 to 0.76329, saving model to weights_copy2.best.hd
f5
308/308 [==============================] - 32s 104ms/step - loss: 0.4304 - auc: 0.7440 -
val_loss: 0.4717 - val_auc: 0.7633
Epoch 9/20
307/308 [============================>.] - ETA: 0s - loss: 0.4249 - auc: 0.7468
Epoch 9: val_auc did not improve from 0.76329
308/308 [==============================] - 30s 98ms/step - loss: 0.4249 - auc: 0.7468 -
val_loss: 0.4566 - val_auc: 0.7613
Epoch 10/20
307/308 [============================>.] - ETA: 0s - loss: 0.4180 - auc: 0.7579
Epoch 10: val_auc improved from 0.76329 to 0.76578, saving model to weights_copy2.best.h
df5
308/308 [==============================] - 31s 102ms/step - loss: 0.4179 - auc: 0.7584 -
val_loss: 0.4722 - val_auc: 0.7658
Epoch 11/20
307/308 [============================>.] - ETA: 0s - loss: 0.4165 - auc: 0.7634
Epoch 11: val_auc did not improve from 0.76578
308/308 [==============================] - 30s 98ms/step - loss: 0.4166 - auc: 0.7631 -
val_loss: 0.4345 - val_auc: 0.7525
Epoch 12/20
307/308 [============================>.] - ETA: 0s - loss: 0.4149 - auc: 0.7657
Epoch 12: val_auc improved from 0.76578 to 0.76835, saving model to weights_copy2.best.h
df5
308/308 [==============================] - 31s 102ms/step - loss: 0.4148 - auc: 0.7663 -
val_loss: 0.4407 - val_auc: 0.7683
Epoch 13/20
307/308 [============================>.] - ETA: 0s - loss: 0.4106 - auc: 0.7728
Epoch 13: val_auc did not improve from 0.76835
308/308 [==============================] - 30s 99ms/step - loss: 0.4107 - auc: 0.7727 -
val_loss: 0.4328 - val_auc: 0.7657
Epoch 14/20
307/308 [============================>.] - ETA: 0s - loss: 0.4107 - auc: 0.7754
Epoch 14: val_auc did not improve from 0.76835
308/308 [==============================] - 30s 98ms/step - loss: 0.4108 - auc: 0.7752 -
val_loss: 0.4334 - val_auc: 0.7683
Epoch 15/20
307/308 [============================>.] - ETA: 0s - loss: 0.4119 - auc: 0.7822
Epoch 15: val_auc did not improve from 0.76835
308/308 [==============================] - 30s 98ms/step - loss: 0.4119 - auc: 0.7822 -
val_loss: 0.4487 - val_auc: 0.7681
Epoch 16/20
307/308 [============================>.] - ETA: 0s - loss: 0.4075 - auc: 0.7887
Epoch 16: val_auc improved from 0.76835 to 0.77143, saving model to weights_copy2.best.h
df5
308/308 [==============================] - 31s 102ms/step - loss: 0.4075 - auc: 0.7887 -
val_loss: 0.4346 - val_auc: 0.7714
Epoch 17/20
307/308 [============================>.] - ETA: 0s - loss: 0.4016 - auc: 0.7982
Epoch 17: val_auc did not improve from 0.77143
308/308 [==============================] - 30s 98ms/step - loss: 0.4016 - auc: 0.7980 -
val_loss: 0.4227 - val_auc: 0.7637
Epoch 18/20
307/308 [============================>.] - ETA: 0s - loss: 0.3981 - auc: 0.8061
Epoch 18: val_auc did not improve from 0.77143
308/308 [==============================] - 30s 99ms/step - loss: 0.3981 - auc: 0.8057 -
val_loss: 0.4447 - val_auc: 0.7655
Epoch 19/20
307/308 [============================>.] - ETA: 0s - loss: 0.3950 - auc: 0.8152
Epoch 19: val_auc did not improve from 0.77143
308/308 [==============================] - 30s 98ms/step - loss: 0.3950 - auc: 0.8152 -
val_loss: 0.4345 - val_auc: 0.7589
Epoch 20/20
307/308 [============================>.] - ETA: 0s - loss: 0.3941 - auc: 0.8273
Epoch 20: val_auc did not improve from 0.77143
```

```
308/308 [==============================] - 30s 99ms/step - loss: 0.3941 - auc: 0.8275 -
val_loss: 0.4383 - val_auc: 0.7557
```

Out[132]:

```
<keras.callbacks.History at 0x7f790e3e2e50>
```

In [133…

```
%tensorboard --logdir logs/fits
```

```
Reusing TensorBoard on port 6006 (pid 1930), started 0:42:27 ago. (Use '!kill 1930' to k
ill it.)
```

Observations:

1. Unlike the previous model there isn't any overfitting in this case.
2. The train and validation loss have converged well as epochs progressed.
3. After few epochs, there was no improvement in auc score. Architectural modifications would be needed to increase the auc score further.

In [134…

```
# input 1
input1 = Input(shape=(250,))
embed1 = Embedding(input_dim = 49678, output_dim = 300, weights = [weight_matrix], train
dropout_layer = SpatialDropout1D(0.3)(embed1)
lstm_layer = CuDNNLSTM(256,return_sequences=True)(dropout_layer)
flat1 = Flatten()(lstm_layer)

# input 2
input2 = Input(shape=(507,1))
conv1 = Conv1D(filters = 64, kernel_size = 3, strides = 1)(input2)
conv2 = Conv1D(filters = 64, kernel_size = 3, strides = 1)(conv1)
flat2 = Flatten()(conv2)

# merging both the inputs
concat = Concatenate()([flat1, flat2])
dense1 = Dense(300, activation = 'relu', kernel_initializer = HeNormal(), kernel_regular
dropout1 = Dropout(0.4)(dense1)
dense2 = Dense(256, activation='relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout2 = Dropout(0.5)(dense2)
batch_norm = BatchNormalization()(dropout2)
dense3 = Dense(128, activation='relu', kernel_initializer = HeNormal(), kernel_regulariz
dropout3 = Dropout(0.6)(dense3)
output = Dense(2, activation = 'softmax')(dropout3)

# create model with two inputs
model3 = Model([input1, input2], output)
model3.load_weights(filepath)
model3.compile(loss='categorical_crossentropy', optimizer = Adam(learning_rate = 0.0006)
```

In [135…

```
train_pred2 = model3.predict([padded_train, train_non_text])

cv_pred2 = model3.predict([padded_cv, cv_non_text])

test_pred2 = model3.predict([padded_test, test_non_text])
```

In [136…

```
print("Train AUC: ",roc_auc_score(y_train, train_pred2[:,1]))

print("CV AUC: ",roc_auc_score(y_cv, cv_pred2[:,1]))

print("Test AUC: ",roc_auc_score(y_test, test_pred2[:,1]))
```

```
Train AUC:  0.8333919894598538
CV AUC:  0.771462843372934
Test AUC:  0.7724210335514641
```

SUMMARY

1. The use of idf values to reduce the no. of text features, help clamp down the no. of parameters, thereby reducing the training time and compuational resources requires without very significant negative impact on the accuracy, compared to the first model where no such reduction of text features was implemented.

2. The addition of convulution along with LSTM did help improve the to a very slight extent compared to the first model but the no. of parameters increased significantly due to which the training time and computational resuources required also increased.

In [137…

```
from prettytable import PrettyTable


x = PrettyTable()
x.field_names = ["Features", "Model", "Epochs", "Train AUC", "CV AUC", "Test AUC"]
x.add_row(["Text embedding using glove vec and ordinal encoded categorical features", "C
x.add_row(["TFIDF based text + glove_vec and ordinal encoded categorical features", "CuD
x.add_row(["Text embedding using glove vec and one-hot encoded categorical features", "C

print(x)
```

```
+-------------------------------------------------------------------------+-------------
-------+--------+-----------+--------+----------+
|                                 Features                                |       Model
       | Epochs | Train AUC | CV AUC | Test AUC |
+-------------------------------------------------------------------------+-------------
-------+--------+-----------+--------+----------+
| Text embedding using glove vec and ordinal encoded categorical features |     CuDNNLST
M      |   20   |   0.811   | 0.767  |   0.77   |
|  TFIDF based text + glove_vec and ordinal encoded categorical features  |     CuDNNLST
M      |   20   |   0.802   | 0.724  |   0.73   |
| Text embedding using glove vec and one-hot encoded categorical features | CuDNNLSTM +
Conv1D |   20   |   0.833   | 0.771  |  0.772   |
+-------------------------------------------------------------------------+-------------
-------+--------+-----------+--------+----------+
```