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

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
import re
from nltk.corpus import stopwords
import pickle
from tqdm import tqdm
import os
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly (https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly)

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

In [0]:
```
#we already have data available for all the initials we need fot poject title
```

In [0]:
```
#loading the data
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [0]:
```
#merging both the data to get a final dataset
price_data = resource_data.groupby('id').agg({'price':'sum','quantity':'sum'}
price_data.head(2)
```

Out[8]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [0]:
```
project_data = pd.merge(project_data,price_data,on = 'id',how = 'left')#getti
```

In [0]:
```
#print(project_data.columns)
project_data.head(2)
```

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sub
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [0]:
```
target = project_data['project_is_approved']
project_data.drop(['Unnamed: 0','id','teacher_id','project_is_approved'],axis
```

In [0]:
```
print('shape of the final data is:',project_data.shape)
print('Shape of target variable is:',target.shape)
print('Features of the final dataset is:',project_data.columns)
```

```
shape of the final data is: (109248, 15)
Shape of target variable is: (109248,)
Features of the final dataset is: Index(['teacher_prefix', 'school_state', 'pro
ject_submitted_datetime',
       'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity'],
      dtype='object')
```

so we have following final features

Categorical Features:

- Teacher_prefix
- school_stae
- project_grade_category
- project_subject_categories
- project_subject_subcategories

Text Features:

- project essay 1
- project essay 2
- project essay 3
- project essay 4
- project resource summary
- project title

Numerical Feature:

- teacher number of previously posted categories
- quantity
- price

Preprocessing the features

1.Preprocessing the categorical features

project_grade_category

In [0]:
```
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-functio
project_data['project_grade_category'] = project_data['project_grade_category
project_data['project_grade_category'] = project_data['project_grade_category
project_data['project_grade_category'] = project_data['project_grade_category
project_data['project_grade_category'].value_counts()
```

Out[25]:
```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

project_subject_category

In [0]:
```
project_data['project_subject_categories'] = project_data['project_subject_ca
project_data['project_subject_categories'] = project_data['project_subject_ca
project_data['project_subject_categories'] = project_data['project_subject_ca
project_data['project_subject_categories'] = project_data['project_subject_ca
project_data['project_subject_categories'] = project_data['project_subject_ca
project_data['project_subject_categories'].value_counts()
```

Out[26]:
```
literacy_language                       23655
math_science                            17072
literacy_language_math_science          14636
health_sports                           10177
music_arts                               5180
specialneeds                             4226
literacy_language_specialneeds           3961
appliedlearning                          3771
math_science_literacy_language           2289
appliedlearning_literacy_language        2191
history_civics                           1851
math_science_specialneeds                1840
literacy_language_music_arts             1757
math_science_music_arts                  1642
appliedlearning_specialneeds             1467
history_civics_literacy_language         1421
health_sports_specialneeds               1391
warmth_care_hunger                       1309
math_science_appliedlearning             1220
```

Teacher prefix

In [0]:
```
# check if we have any nan values are there
print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.s
```

```
True
number of nan values 3
```

> numebr of missing values are very less in number, we can replace it with Mrs. as most of the projects are submitted by Mrs.

In [0]:
```
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.replace('
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix'].value_counts()
```

Out[28]:
```
mrs        57272
ms         38955
mr         10648
teacher     2360
dr            13
Name: teacher_prefix, dtype: int64
```

Project subject subcategories

In [0]:
```
project_data['project_subject_subcategories'] = project_data['project_subject
project_data['project_subject_subcategories'] = project_data['project_subject
project_data['project_subject_subcategories'] = project_data['project_subject
project_data['project_subject_subcategories'] = project_data['project_subject
project_data['project_subject_subcategories'] = project_data['project_subject
project_data['project_subject_subcategories'].value_counts()
```

Out[29]:
```
literacy                                9486
literacy_mathematics                    8325
literature_writing_mathematics          5923
literacy_literature_writing             5571
mathematics                             5379
                                        ...
gym_fitness_warmth_care_hunger             1
civics_government_foreignlanguages         1
economics_nutritioneducation               1
financialliteracy_performingarts           1
extracurricular_financialliteracy          1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

School state

In [0]:
```
project_data['school_state'] = project_data['school_state'].str.lower()
project_data['school_state'].value_counts()
```

Out[30]:
```
ca    15388
tx     7396
ny     7318
fl     6185
nc     5091
il     4350
ga     3963
sc     3936
mi     3161
pa     3109
in     2620
mo     2576
oh     2467
la     2394
ma     2389
wa     2334
ok     2276
nj     2237
az     2147
```

2.Preproocessing text features

In [0]:

```
#writing the utility functions for text cleaning
from tqdm import tqdm
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'i
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', '
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'beca
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'th
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shou
            'won', "won't", 'wouldn', "wouldn't"]

#================================================================================
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    """Function for opening thee decontracted words"""
    # specific
    phrase = re.sub(r"won't", "will not", phrase)#converting to will not
    phrase = re.sub(r"can\'t", "can not", phrase)#converting to can not

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

# Combining all the above stundents
def preprocess_text(text_data):
    """cleaning and remving the stopwords"""
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords
        preprocessed_text.append(sent.lower().strip())
```

```
    return preprocessed_text
```

project title

In [0]:
```
preprocessed_titles = preprocess_text(project_data['project_title'].values)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 1
09248/109248 [00:05<00:00, 19899.20it/s]
```

In [0]:
```
project_data['cleaned_project_titles'] = preprocessed_titles
project_data.drop(['project_title'],axis = 1,inplace = True)
```

Project resource summary

In [0]:
```
preprocessed_resource_summary = preprocess_text(project_data['project_resourc
project_data['cleaned_resource_summary'] = preprocessed_resource_summary
project_data.drop(['project_resource_summary'],axis = 1,inplace = True)
```

```
100%|██████████████████████████████████████████████████████████████████████████████|
109248/109248 [00:10<00:00, 9999.71it/s]
```

project essays

In [0]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [0]:
```
preprocessed_essays = preprocess_text(project_data['essay'].values)
project_data['cleaned_essay'] = preprocessed_essays
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','pro
```

```
100%|██████████████████████████████████████████████████████████████████████████████|
109248/109248 [02:04<00:00, 874.01it/s]
```

3.Preprocessing nuerical features

MinmaxScaler

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

scaler = MinMaxScaler()
scaler.fit(project_data['price'].values.reshape(-1, 1))
project_data['nrm_price']=scaler.transform(project_data['price'].values.resha
project_data
project_data.head(2)
```

Out[41]:

	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_subject_
0	mrs	in	2016-12-05 13:43:57	grades_prek_2	literac
1	mr	fl	2016-10-25 09:22:10	grades_6_8	history_civics_h

In [0]:
```
#Save the processed dataset into a pandas CSV file.
project_data.to_csv("processed_data.csv", index=False)
```

Preparing the data for modelling

- Vectorizing categorical features using one hot encoding
- Vectorizing text data using BOW or TFIDF/W2V TFIDF
- Standardizing or normalizing numerical data

In [0]:
```
#splitting the data
project_data = pd.read_csv('processed_data.csv')
project_data.head(2)
```

Out[13]:

	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_subject_
0	mrs	in	2016-12-05 13:43:57	grades_prek_2	literac
1	mr	fl	2016-10-25 09:22:10	grades_6_8	history_civics_h

In [0]:
```
project_data.columns
```

Out[14]:
```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'cleaned_project_titles', 'cleaned_resource_summary', 'cleaned_essay',
       'nrm_price'],
      dtype='object')
```

In [0]:
```
#splitting the data in train,validation and test
from sklearn.model_selection import train_test_split
#80% training data and 20% test data
X_train,X_test,Y_train,Y_test = train_test_split(project_data,target,test_siz
X_train,X_cv,Y_train,Y_cv = train_test_split(X_train,Y_train,test_size = 0.2,

print('shape of training data after vectorization is:',X_train.shape)
print('shape of validation data after vectorization is:',X_cv.shape)
print('shape of test data after vectorization is:',X_test.shape)
```

In [0]:
```
print('shape of training data after vectorization is:',X_train.shape)
print('shape of validation data after vectorization is:',X_cv.shape)
print('shape of test data after vectorization is:',X_test.shape)
```

```
shape of training data after vectorization is: (69918, 13)
shape of validation data after vectorization is: (17480, 13)
shape of test data after vectorization is: (21850, 13)
```

In [3]:
```
#trying the embeddings
from keras.models import Sequential
from keras.preprocessing import sequence
from keras.initializers import he_normal
from keras.layers import Input
from keras.layers import Dense
from keras.layers.embeddings import Embedding
from keras.regularizers import L1L2
from keras.layers import BatchNormalization
from keras.layers import Dropout
from keras.layers import LSTM
from keras.layers import Flatten
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Model
from keras.layers import concatenate
from sklearn.metrics import accuracy_score
from IPython.core.display import display,HTML
from numpy import zeros
display(HTML("<style>.container { width:100% !important; }</style>"))
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

2. Vectorizing the categorical features

2.1 Project_category

In [0]:
```
from collections import Counter
def count(df):

    """Returns the sorted dinctionary keys"""
    my_counter = Counter() #initiating counter to calculate occurence of word
    for word in df.values:
        my_counter.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
    cat_dict = dict(my_counter)
    sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
    return sorted_cat_dict
```

In [0]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
cat_dict = count(project_data['project_subject_categories'])
vectorizer = CountVectorizer(vocabulary=list(cat_dict.keys()), lowercase=Fals
vectorizer.fit(X_train['project_subject_categories'].values)
print(vectorizer.get_feature_names()[:10])


train_categories = vectorizer.transform(X_train['project_subject_categories']
val_categories = vectorizer.transform(X_cv['project_subject_categories'].valu
test_categories = vectorizer.transform(X_test['project_subject_categories'].v
print("Shape of train matrix after one hot encodig ",train_categories.shape)
print("Shape of val matrix after one hot encodig ",val_categories.shape)
print("Shape of test matrix after one hot encodig ",test_categories.shape)

#vocab_size_categories = train_categories.shape[1] + 1

```

```
['history_civics_warmth_care_hunger', 'music_arts_warmth_care_hunger', 'literac
y_language_warmth_care_hunger', 'music_arts_appliedlearning', 'appliedlearning_
warmth_care_hunger', 'math_science_warmth_care_hunger', 'history_civics_health_
sports', 'music_arts_history_civics', 'music_arts_health_sports', 'health_sport
s_warmth_care_hunger']
Shape of train matrix after one hot encodig  (69918, 51)
Shape of val matrix after one hot encodig  (17480, 51)
Shape of test matrix after one hot encodig  (21850, 51)
```

In [0]:
```
#getting the embedding layer
#Get the flattened output for clean_categories
input_layer_clean_categories = Input(shape=(train_categories.shape[1],), name
embedding_layer_clean_categories = Embedding(input_dim=train_categories.shape
flatten_clean_categories = Flatten()(embedding_layer_clean_categories)
```

2.2 Project subcategories

In [0]:
```
# we use count vectorizer to convert the values into one hot encoded features
#from sklearn.feature_extraction.text import CountVectorizer
cat_dict = count(project_data['project_subject_subcategories'])
vectorizer = CountVectorizer(vocabulary=list(cat_dict.keys()), lowercase=Fals
vectorizer.fit(X_train['project_subject_subcategories'].values)
print(vectorizer.get_feature_names()[:10])


train_subcategories = vectorizer.transform(X_train['project_subject_subcatego
val_subcategories = vectorizer.transform(X_cv['project_subject_subcategories'
test_subcategories = vectorizer.transform(X_test['project_subject_subcategori
print("Shape of train matrix after one hot encodig ",train_subcategories.shap
print("Shape of val matrix after one hot encodig ",val_subcategories.shape)
print("Shape of test matrix after one hot encodig ",test_subcategories.shape)

```

```
['gym_fitness_parentinvolvement', 'communityservice_gym_fitness', 'college_care
erprep_warmth_care_hunger', 'economics_other', 'parentinvolvement_warmth_care_h
unger', 'communityservice_music', 'gym_fitness_socialsciences', 'parentinvolvem
ent_teamsports', 'civics_government_nutritioneducation', 'financialliteracy_for
eignlanguages']
Shape of train matrix after one hot encodig  (69918, 401)
Shape of val matrix after one hot encodig  (17480, 401)
Shape of test matrix after one hot encodig  (21850, 401)
```

In [0]:
```
#getting the embedding layer
#Get the flattened output for clean_subcategories
input_layer_subcategories = Input(shape=(train_subcategories.shape[1],), name
embedding_layer_subcategories = Embedding(input_dim=train_subcategories.shape
flatten_subcategories = Flatten()(embedding_layer_subcategories)
```

2.3 Teacher Prefix

In [0]:
```
cat_dict = count(project_data['teacher_prefix'])
vectorizer = CountVectorizer(vocabulary=list(cat_dict.keys()), lowercase=Fals
vectorizer.fit(X_train['teacher_prefix'].values)
print(vectorizer.get_feature_names()[:10])


train_prefix = vectorizer.transform(X_train['teacher_prefix'].values)
val_prefix = vectorizer.transform(X_cv['teacher_prefix'].values)
test_prefix= vectorizer.transform(X_test['teacher_prefix'].values)
print("Shape of train matrix after one hot encodig ",train_prefix.shape)
print("Shape of val matrix after one hot encodig ",val_prefix.shape)
print("Shape of test matrix after one hot encodig ",test_prefix.shape)

```

```
['dr', 'teacher', 'mr', 'ms', 'mrs']
Shape of train matrix after one hot encodig  (69918, 5)
Shape of val matrix after one hot encodig  (17480, 5)
Shape of test matrix after one hot encodig  (21850, 5)
```

In [0]:
```
#getting the embedding layer
#Get the flattened output for teacher_prefix
input_layer_prefix = Input(shape=(train_prefix.shape[1],), name = "input_pref
embedding_layer_prefix = Embedding(input_dim=train_prefix.shape[1], output_di
flatten_prefix = Flatten()(embedding_layer_prefix)
```

2.4School state

In [0]:
```
cat_dict = count(project_data['school_state'])
vectorizer = CountVectorizer(vocabulary=list(cat_dict.keys()), lowercase=Fals
vectorizer.fit(X_train['school_state'].values)
print(vectorizer.get_feature_names())


train_state = vectorizer.transform(X_train['school_state'].values)
val_state = vectorizer.transform(X_cv['school_state'].values)
test_state= vectorizer.transform(X_test['school_state'].values)
print("Shape of train matrix after one hot encodig ",train_state.shape)
print("Shape of val matrix after one hot encodig ",val_state.shape)
print("Shape of test matrix after one hot encodig ",test_state.shape)
```

```
['vt', 'wy', 'nd', 'mt', 'ri', 'sd', 'ne', 'de', 'ak', 'nh', 'wv', 'me', 'hi',
'dc', 'nm', 'ks', 'ia', 'id', 'ar', 'co', 'mn', 'or', 'ky', 'ms', 'nv', 'md',
'ct', 'tn', 'ut', 'al', 'wi', 'va', 'az', 'nj', 'ok', 'wa', 'ma', 'la', 'oh',
'mo', 'in', 'pa', 'mi', 'sc', 'ga', 'il', 'nc', 'fl', 'ny', 'tx', 'ca']
Shape of train matrix after one hot encodig  (69918, 51)
Shape of val matrix after one hot encodig  (17480, 51)
Shape of test matrix after one hot encodig  (21850, 51)
```

In [0]:
```
#getting the embedding layer
#Get the flattened output for clean_subcategories
input_layer_state = Input(shape=(train_state.shape[1],), name = "input_state"
embedding_layer_state = Embedding(input_dim=train_state.shape[1], output_dim=
flatten_state = Flatten()(embedding_layer_state)
```

2.5 Project grade category

In [0]:
```
cat_dict = count(project_data['project_grade_category'])
vectorizer = CountVectorizer(vocabulary=list(cat_dict.keys()), lowercase=Fals
vectorizer.fit(X_train['project_grade_category'].values)
print(vectorizer.get_feature_names()[:10])


train_grade_category = vectorizer.transform(X_train['project_grade_category']
val_grade_category = vectorizer.transform(X_cv['project_grade_category'].valu
test_grade_category= vectorizer.transform(X_test['project_grade_category'].va
print("Shape of train matrix after one hot encodig ",train_grade_category.sha
print("Shape of val matrix after one hot encodig ",val_grade_category.shape)
print("Shape of test matrix after one hot encodig ",test_grade_category.shape
```

```
['grades_9_12', 'grades_6_8', 'grades_3_5', 'grades_prek_2']
Shape of train matrix after one hot encodig  (69918, 4)
Shape of val matrix after one hot encodig  (17480, 4)
Shape of test matrix after one hot encodig  (21850, 4)
```

In [0]:
```
#getting the embedding layer
#Get the flattened output for project grade category
input_layer_grade_category = Input(shape=(train_grade_category.shape[1],), na
embedding_layer_grade_category = Embedding(input_dim=train_grade_category.sha
flatten_grade_category = Flatten()(embedding_layer_grade_category)
```

Vectorizing the text data

as we want to input the total text that's why we must get the final text by combining all

In [0]:
```
#utility function
def final_text(df):
    """Returns the final concatenated text"""
    df['final_Text'] = df['cleaned_essay'].map(str) + " " + df['cleaned_resou
        " " + df['cleaned_project_titles'].map(str)
    return df

#getting the train,val and test data
train_text = final_text(X_train)
val_text = final_text(X_cv)
test_text = final_text(X_test)
```

In [0]:
```
#building the data for modelling


#initializing the tokenizer
tokens = Tokenizer()
tokens.fit_on_texts(list(train_text['final_Text'].values))#will convert texts

#integer encoding the documents
docs_train = tokens.texts_to_sequences(list(train_text['final_Text'].values))
docs_val = tokens.texts_to_sequences(list(val_text['final_Text'].values))
docs_test = tokens.texts_to_sequences(list(test_text['final_Text'].values))

vocab_size = len(tokens.word_index) + 1

max_length = 300#for padding the sequences

#finally padding the squences
padded_docs_train = pad_sequences(docs_train, maxlen=max_length, padding='pos
padded_docs_val = pad_sequences(docs_val, maxlen=max_length, padding='post')
padded_docs_test = pad_sequences(docs_test, maxlen=max_length, padding='post'

#creating a matrix for each word in training data
pickle_in = open("glove_vectors","rb")
glove_words = pickle.load(pickle_in)
embedding_matrix = zeros((vocab_size, 300))
for word, i in tokens.word_index.items(): #enumerating all unique words
    embedding_vector = glove_words.get(word)#locating for loaded glove model
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [0]:
```
vocab_size
```

Out[123]: 51153

In [0]:
```
#as we choose the vector to be 300 dimensional
#that's why the output dimension must be 300
input_text_layer = Input(shape = (300,),name = 'input_total_text_sequence')
embedding_layer_text = Embedding(input_dim = vocab_size,output_dim = 300,weig
#here input_dim is size of vocablury of text data
#output_dim is size of embedded vector
```

In [0]:
```
#lstm for this layer
lstm_layer = LSTM(32,activation = 'relu',return_sequences = True)(embedding_l
#and finally flttening the output
flatten_text = Flatten()(lstm_layer)
```

Normalizing the numerical data

Teacher number of previously posted projects

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

norm = Normalizer()
train_ppp = norm.fit_transform(X_train['teacher_number_of_previously_posted_p
val_ppp = norm.transform(X_cv['teacher_number_of_previously_posted_projects']
test_ppp = norm.transform(X_test['teacher_number_of_previously_posted_project
```

In [0]:
```
train_ppp = train_ppp.reshape(len(X_train),1)
val_ppp = val_ppp.reshape(len(X_cv),1)
test_ppp = test_ppp.reshape(len(X_test),1)
```

In [0]:
```
input_num_1 = Input(shape = (train_ppp.shape[1],),name = 'input_previously_po
```

Quantity

In [0]:
```
norm = Normalizer()
train_quantity = norm.fit_transform(X_train.quantity.values.reshape(-1,1))
val_quantity = norm.transform(X_cv.quantity.values.reshape(-1,1))
test_quantity = norm.transform(X_test.quantity.values.reshape(-1,1))

train_quantity = train_quantity.reshape(len(X_train),1)
val_quantity = val_quantity.reshape(len(X_cv),1)
test_quantity = test_quantity.reshape(len(X_test),1)

```

In [0]:
```
input_num_2 = Input(shape = (train_quantity.shape[1],),name = 'input_quantity
```

Price

In [0]:
```
norm = Normalizer()
train_price = norm.fit_transform(X_train.price.values.reshape(-1,1))
val_price = norm.fit_transform(X_cv.price.values.reshape(-1,1))
test_price = norm.fit_transform(X_test.price.values.reshape(-1,1))

train_price = train_price.reshape(len(X_train),1)
val_price = val_price.reshape(len(X_cv),1)
test_price = test_price.reshape(len(X_test),1)
```

In [0]:
```
input_num_3 = Input(shape = (train_price.shape[1],),name = 'input_price')
```

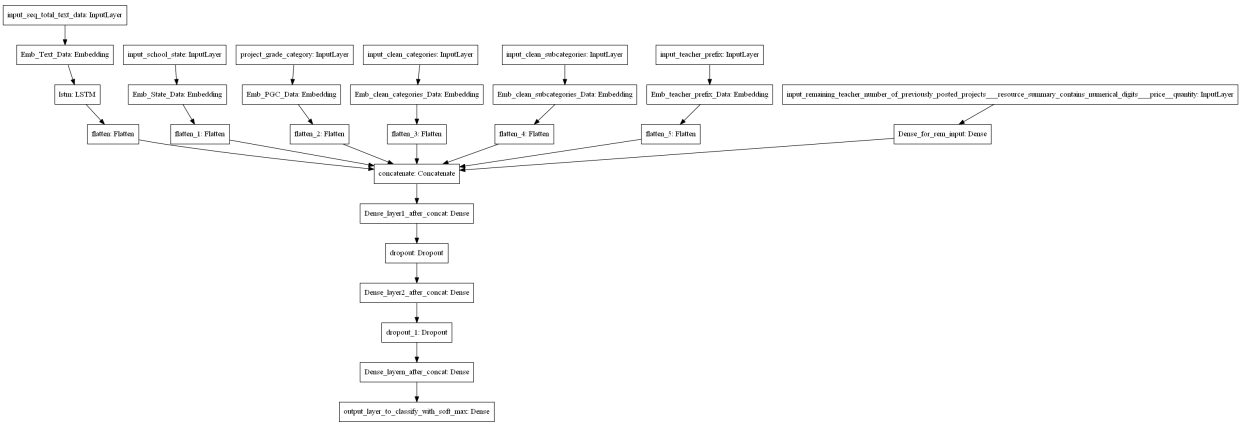
In [0]:
```
#concatenating all
num_final_concatenate = concatenate(inputs = [input_num_1,input_num_2,input_n
#dense layer for numerical features as described by the model architecture
dense_num_layer = Dense(16,activation = 'relu',kernel_initializer = 'he_norma
```

Assignment : 14

1. Preprocess all the Data we have in DonorsChoose Dataset (https://drive.google.com/drive/folders/1MIwK7BQMev8f5CbDDVNLPaFGB32pFN60) use train.csv
2. Combine 4 essay's into one column named - 'preprocessed_essays'.
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' (https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics) as a metric. check this (https://datascience.stackexchange.com/a/20192) for using auc as a metric
5. You are free to choose any number of layers/hiddden units but you have to use same type of architectures shown below.
6. You can use any one of the optimizers and choice of Learning rate and momentum, resources: cs231n class notes (http://cs231n.github.io/neural-networks-3/), cs231n class video (https://www.youtube.com/watch?v=hd_KFJ5ktUc).
7. For all the model's use TensorBoard (https://www.youtube.com/watch?v=2U6Jl7oqRkM) and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots and include those images in .ipynb notebook and PDF.
8. Use Categorical Cross Entropy as Loss to minimize.

Model-1

Build and Train deep neural network as shown below



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

- **Input_seq_total_text_data** --- You have to give Total text data columns. After this use the Embedding layer to get word vectors. Use given predefined glove word vectors, don't train any word vectors. After this use LSTM and get the LSTM output and Flatten that output.
- **Input_school_state** --- Give 'school_state' column as input to embedding layer and Train the Keras Embedding layer.
- **Project_grade_category** --- Give 'project_grade_category' column as input to embedding layer and Train the Keras Embedding layer.

- **Input_clean_categories** --- Give 'input_clean_categories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_clean_subcategories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_teacher_prefix' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_remaining_teacher_number_of_previously_posted_projects._resource_summary_c** ---concatenate remaining columns and add a Dense layer after that.

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

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

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

1. Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/)

2. Please go through this link https://keras.io/getting-started/functional-api-guide/ (https://keras.io/getting-started/functional-api-guide/) and check the 'Multi-input and multi-output models' then you will get to know how to give multiple inputs.

In [0]:

```
x = concatenate(inputs = [flatten_text,flatten_prefix,flatten_clean_categorie
                name = 'final_concatenated_layer')

x = Dense(8,activation = 'relu',kernel_initializer = 'he_normal',name = 'dens
x = Dropout(0.3,name = 'dropout_1')(x)
x = Dense(8,activation= 'relu',kernel_initializer = 'he_normal',name = 'dense
x = Dropout(0.3,name = 'dropout_2')(x)

output_layer = Dense(1,activation = 'sigmoid',name = 'output_layer')(x)#for b

model = Model(inputs = [input_text_layer,input_layer_prefix,input_layer_clean
                        input_num_1,input_num_2,input_num_3],outputs = [output
import warnings
warnings.filterwarnings('ignore')
model.summary()
```

```
__________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
==================
input_total_text_sequence (Inpu (None, 300)          0
__________________________________________________________________________________
__________________
embedding_19 (Embedding)        (None, 300, 300)     15345900    input_total_te
xt_sequence[0][0]
__________________________________________________________________________________
__________________
input_prefix (InputLayer)       (None, 5)            0
__________________________________________________________________________________
__________________
input_categories (InputLayer)   (None, 51)           0
__________________________________________________________________________________
__________________
input_subcategories (InputLayer (None, 401)          0
__________________________________________________________________________________
__________________
input_state (InputLayer)        (None, 51)           0
__________________________________________________________________________________
__________________
input_grade_category (InputLaye (None, 4)            0
__________________________________________________________________________________
__________________
input_previously_posted_project (None, 1)            0
__________________________________________________________________________________
__________________
input_quantity (InputLayer)     (None, 1)            0
__________________________________________________________________________________
__________________
input_price (InputLayer)        (None, 1)            0
__________________________________________________________________________________
__________________
lstm_2 (LSTM)                   (None, 300, 32)      42624       embedding_19
[0][0]
__________________________________________________________________________________
```

```
__________________
embedding_16 (Embedding)        (None, 5, 4)         20          input_prefix
[0][0]
____________________________________________________________________________
__________________
embedding_14 (Embedding)        (None, 51, 4)        204         input_categori
es[0][0]
____________________________________________________________________________
__________________
embedding_15 (Embedding)        (None, 401, 4)       1604        input_subcateg
ories[0][0]
____________________________________________________________________________
__________________
embedding_17 (Embedding)        (None, 51, 4)        204         input_state[0]
[0]
____________________________________________________________________________
__________________
embedding_18 (Embedding)        (None, 4, 4)         16          input_grade_ca
tegory[0][0]
____________________________________________________________________________
__________________
numerical_input_final (Concaten (None, 3)            0           input_previous
ly_posted_projects[
                                                                 input_quantity
[0][0]
                                                                 input_price[0]
[0]
____________________________________________________________________________
__________________
flatten_18 (Flatten)            (None, 9600)         0           lstm_2[0][0]
____________________________________________________________________________
__________________
flatten_15 (Flatten)            (None, 20)           0           embedding_16
[0][0]
____________________________________________________________________________
__________________
flatten_13 (Flatten)            (None, 204)          0           embedding_14
[0][0]
____________________________________________________________________________
__________________
flatten_14 (Flatten)            (None, 1604)         0           embedding_15
[0][0]
____________________________________________________________________________
__________________
flatten_16 (Flatten)            (None, 204)          0           embedding_17
[0][0]
____________________________________________________________________________
__________________
flatten_17 (Flatten)            (None, 16)           0           embedding_18
[0][0]
____________________________________________________________________________
__________________
dense_layer_1 (Dense)           (None, 16)           64          numerical_inpu
t_final[0][0]
____________________________________________________________________________
__________________
final_concatenated_layer (Conca (None, 11664)        0           flatten_18[0]
```

```
[0]                                                                    flatten_15[0]
[0]                                                                    flatten_13[0]
[0]                                                                    flatten_14[0]
[0]                                                                    flatten_16[0]
[0]                                                                    flatten_17[0]
[0]                                                                    dense_layer_1
[0][0]
__________________________________________________________________________________
__________________
dense_layer_concatenate_1 (Dens (None, 8)            93320       final_concaten
ated_layer[0][0]
__________________________________________________________________________________
__________________
dropout_1 (Dropout)             (None, 8)            0           dense_layer_co
ncatenate_1[0][0]
__________________________________________________________________________________
__________________
dense_layer_concatenate_2 (Dens (None, 8)            72          dropout_1[0]
[0]
__________________________________________________________________________________
__________________
dropout_2 (Dropout)             (None, 8)            0           dense_layer_co
ncatenate_2[0][0]
__________________________________________________________________________________
__________________
output_layer (Dense)            (None, 1)            9           dropout_2[0]
[0]
==================================================================================
==================
Total params: 15,484,037
Trainable params: 138,137
Non-trainable params: 15,345,900
__________________________________________________________________________________
__________________
```

Defining the custom metric for AUC

In [0]:

```
import tensorflow as tf
from keras import backend as K
from sklearn.metrics import roc_auc_score

#https://stackoverflow.com/questions/51922500/tf-metrics-auc-yielding-very-di
def roc_auc(y_true, y_pred):
    auc = tf.py_func(roc_auc_score, (y_true, y_pred), tf.double)
    #auc = tf.metrics.auc(y_true, y_pred, num_thresholds=200)[1]
    K.get_session().run(tf.local_variables_initializer())
    return auc
```

Defining Callbacks

In [0]:

```
import datetime
from tensorflow.python.keras.callbacks import TensorBoard
from keras.callbacks import ModelCheckpoint

tensorboard = TensorBoard("logs\\fit\\" + datetime.datetime.now().strftime("%
filepath="weights_best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_roc_auc', verbose=1, save
```

In [0]:
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[roc_auc]
model.fit(x=[padded_docs_train,train_prefix,train_categories,train_subcategor
          y=[np.array(Y_train)],
          validation_data=([padded_docs_val,val_prefix,val_categories,val_sub
          epochs=7,
          batch_size=1024,
          callbacks=[tensorboard,checkpoint])
```

```
Train on 69918 samples, validate on 17480 samples
Epoch 1/10
69918/69918 [==============================] - ETA: 8:21 - loss: 0.6649 - roc
_auc: 0.520 - ETA: 6:40 - loss: 0.6117 - roc_auc: 0.514 - ETA: 6:02 - loss:
0.5718 - roc_auc: 0.518 - ETA: 5:43 - loss: 0.5482 - roc_auc: 0.525 - ETA: 5:
26 - loss: 0.5364 - roc_auc: 0.522 - ETA: 5:19 - loss: 0.5336 - roc_auc: 0.51
1 - ETA: 5:23 - loss: 0.5316 - roc_auc: 0.512 - ETA: 5:12 - loss: 0.5278 - ro
c_auc: 0.513 - ETA: 5:04 - loss: 0.5250 - roc_auc: 0.512 - ETA: 4:59 - loss:
0.5223 - roc_auc: 0.511 - ETA: 4:52 - loss: 0.5185 - roc_auc: 0.513 - ETA: 4:
45 - loss: 0.5137 - roc_auc: 0.517 - ETA: 4:39 - loss: 0.5109 - roc_auc: 0.51
7 - ETA: 4:32 - loss: 0.5105 - roc_auc: 0.518 - ETA: 4:26 - loss: 0.5102 - ro
c_auc: 0.516 - ETA: 4:21 - loss: 0.5092 - roc_auc: 0.514 - ETA: 4:15 - loss:
0.5052 - roc_auc: 0.516 - ETA: 4:09 - loss: 0.5045 - roc_auc: 0.519 - ETA: 4:
04 - loss: 0.5031 - roc_auc: 0.521 - ETA: 3:59 - loss: 0.5009 - roc_auc: 0.52
1 - ETA: 3:54 - loss: 0.5000 - roc_auc: 0.521 - ETA: 3:49 - loss: 0.4988 - ro
c_auc: 0.522 - ETA: 3:45 - loss: 0.4970 - roc_auc: 0.525 - ETA: 3:40 - loss:
0.4962 - roc_auc: 0.524 - ETA: 3:35 - loss: 0.4952 - roc_auc: 0.524 - ETA: 3:
29 - loss: 0.4935 - roc_auc: 0.526 - ETA: 3:25 - loss: 0.4928 - roc_auc: 0.52
5 - ETA: 3:20 - loss: 0.4931 - roc_auc: 0.526 - ETA: 3:15 - loss: 0.4928 - ro
c_auc: 0.526 - ETA: 3:09 - loss: 0.4909 - roc_auc: 0.528 - ETA: 3:04 - loss:
0.4894 - roc_auc: 0.532 - ETA: 2:59 - loss: 0.4887 - roc_auc: 0.534 - ETA: 2:
54 - loss: 0.4877 - roc_auc: 0.536 - ETA: 2:49 - loss: 0.4868 - roc_auc: 0.53
8 - ETA: 2:44 - loss: 0.4860 - roc_auc: 0.539 - ETA: 2:39 - loss: 0.4843 - ro
c_auc: 0.540 - ETA: 2:34 - loss: 0.4841 - roc_auc: 0.541 - ETA: 2:29 - loss:
0.4837 - roc_auc: 0.543 - ETA: 2:24 - loss: 0.4829 - roc_auc: 0.545 - ETA: 2:
19 - loss: 0.4817 - roc_auc: 0.547 - ETA: 2:15 - loss: 0.4802 - roc_auc: 0.55
0 - ETA: 2:10 - loss: 0.4803 - roc_auc: 0.551 - ETA: 2:05 - loss: 0.4794 - ro
c_auc: 0.552 - ETA: 2:00 - loss: 0.4787 - roc_auc: 0.553 - ETA: 1:55 - loss:
0.4774 - roc_auc: 0.555 - ETA: 1:50 - loss: 0.4764 - roc_auc: 0.557 - ETA: 1:
45 - loss: 0.4751 - roc_auc: 0.559 - ETA: 1:40 - loss: 0.4750 - roc_auc: 0.55
9 - ETA: 1:35 - loss: 0.4743 - roc_auc: 0.561 - ETA: 1:30 - loss: 0.4744 - ro
c_auc: 0.561 - ETA: 1:26 - loss: 0.4734 - roc_auc: 0.562 - ETA: 1:21 - loss:
0.4729 - roc_auc: 0.563 - ETA: 1:16 - loss: 0.4723 - roc_auc: 0.564 - ETA: 1:
11 - loss: 0.4717 - roc_auc: 0.566 - ETA: 1:06 - loss: 0.4709 - roc_auc: 0.56
7 - ETA: 1:01 - loss: 0.4701 - roc_auc: 0.569 - ETA: 56s - loss: 0.4694 - roc
_auc: 0.570 - ETA: 51s - loss: 0.4684 - roc_auc: 0.57 - ETA: 46s - loss: 0.46
85 - roc_auc: 0.57 - ETA: 41s - loss: 0.4682 - roc_auc: 0.57 - ETA: 36s - los
s: 0.4674 - roc_auc: 0.57 - ETA: 31s - loss: 0.4662 - roc_auc: 0.57 - ETA: 26
s - loss: 0.4657 - roc_auc: 0.57 - ETA: 21s - loss: 0.4652 - roc_auc: 0.57 -
ETA: 16s - loss: 0.4645 - roc_auc: 0.57 - ETA: 11s - loss: 0.4642 - roc_auc:
0.57 - ETA: 6s - loss: 0.4639 - roc_auc: 0.5798 - ETA: 1s - loss: 0.4635 - ro
c_auc: 0.580 - 400s 6ms/step - loss: 0.4634 - roc_auc: 0.5813 - val_loss: 0.3
907 - val_roc_auc: 0.7212

Epoch 00001: val_roc_auc improved from -inf to 0.72119, saving model to weigh
ts_best.hdf5
Epoch 2/10
```

```
69918/69918 [==============================] - ETA: 5:38 - loss: 0.4490 - roc
_auc: 0.593 - ETA: 5:51 - loss: 0.4414 - roc_auc: 0.624 - ETA: 5:40 - loss:
0.4385 - roc_auc: 0.632 - ETA: 5:31 - loss: 0.4340 - roc_auc: 0.639 - ETA: 5:
24 - loss: 0.4313 - roc_auc: 0.640 - ETA: 5:16 - loss: 0.4306 - roc_auc: 0.64
1 - ETA: 5:10 - loss: 0.4236 - roc_auc: 0.645 - ETA: 5:03 - loss: 0.4241 - ro
c_auc: 0.643 - ETA: 4:59 - loss: 0.4288 - roc_auc: 0.644 - ETA: 4:53 - loss:
0.4347 - roc_auc: 0.641 - ETA: 4:48 - loss: 0.4375 - roc_auc: 0.641 - ETA: 4:
44 - loss: 0.4403 - roc_auc: 0.643 - ETA: 4:39 - loss: 0.4415 - roc_auc: 0.64
4 - ETA: 4:36 - loss: 0.4435 - roc_auc: 0.643 - ETA: 4:32 - loss: 0.4435 - ro
c_auc: 0.644 - ETA: 4:27 - loss: 0.4429 - roc_auc: 0.647 - ETA: 4:23 - loss:
0.4420 - roc_auc: 0.646 - ETA: 4:17 - loss: 0.4425 - roc_auc: 0.649 - ETA: 4:
12 - loss: 0.4405 - roc_auc: 0.649 - ETA: 4:07 - loss: 0.4424 - roc_auc: 0.64
6 - ETA: 4:02 - loss: 0.4423 - roc_auc: 0.646 - ETA: 3:56 - loss: 0.4426 - ro
c_auc: 0.646 - ETA: 3:51 - loss: 0.4424 - roc_auc: 0.645 - ETA: 3:45 - loss:
0.4420 - roc_auc: 0.645 - ETA: 3:41 - loss: 0.4412 - roc_auc: 0.645 - ETA: 3:
36 - loss: 0.4404 - roc_auc: 0.645 - ETA: 3:31 - loss: 0.4392 - roc_auc: 0.64
6 - ETA: 3:26 - loss: 0.4389 - roc_auc: 0.648 - ETA: 3:21 - loss: 0.4380 - ro
c_auc: 0.648 - ETA: 3:16 - loss: 0.4377 - roc_auc: 0.647 - ETA: 3:10 - loss:
0.4375 - roc_auc: 0.647 - ETA: 3:05 - loss: 0.4371 - roc_auc: 0.646 - ETA: 3:
00 - loss: 0.4374 - roc_auc: 0.647 - ETA: 2:54 - loss: 0.4383 - roc_auc: 0.64
6 - ETA: 2:49 - loss: 0.4384 - roc_auc: 0.646 - ETA: 2:44 - loss: 0.4381 - ro
c_auc: 0.646 - ETA: 2:39 - loss: 0.4384 - roc_auc: 0.647 - ETA: 2:34 - loss:
0.4385 - roc_auc: 0.647 - ETA: 2:29 - loss: 0.4380 - roc_auc: 0.647 - ETA: 2:
24 - loss: 0.4379 - roc_auc: 0.648 - ETA: 2:18 - loss: 0.4370 - roc_auc: 0.64
8 - ETA: 2:14 - loss: 0.4363 - roc_auc: 0.650 - ETA: 2:08 - loss: 0.4355 - ro
c_auc: 0.650 - ETA: 2:03 - loss: 0.4351 - roc_auc: 0.651 - ETA: 1:58 - loss:
0.4352 - roc_auc: 0.651 - ETA: 1:53 - loss: 0.4346 - roc_auc: 0.651 - ETA: 1:
48 - loss: 0.4343 - roc_auc: 0.652 - ETA: 1:43 - loss: 0.4353 - roc_auc: 0.65
1 - ETA: 1:38 - loss: 0.4348 - roc_auc: 0.651 - ETA: 1:33 - loss: 0.4345 - ro
c_auc: 0.652 - ETA: 1:28 - loss: 0.4344 - roc_auc: 0.652 - ETA: 1:22 - loss:
0.4340 - roc_auc: 0.653 - ETA: 1:17 - loss: 0.4340 - roc_auc: 0.653 - ETA: 1:
12 - loss: 0.4338 - roc_auc: 0.653 - ETA: 1:07 - loss: 0.4334 - roc_auc: 0.65
3 - ETA: 1:02 - loss: 0.4334 - roc_auc: 0.654 - ETA: 57s - loss: 0.4330 - roc
_auc: 0.655 - ETA: 52s - loss: 0.4332 - roc_auc: 0.65 - ETA: 47s - loss: 0.43
28 - roc_auc: 0.65 - ETA: 42s - loss: 0.4318 - roc_auc: 0.65 - ETA: 37s - los
s: 0.4311 - roc_auc: 0.65 - ETA: 32s - loss: 0.4310 - roc_auc: 0.65 - ETA: 27
s - loss: 0.4301 - roc_auc: 0.65 - ETA: 22s - loss: 0.4307 - roc_auc: 0.65 -
ETA: 17s - loss: 0.4304 - roc_auc: 0.65 - ETA: 11s - loss: 0.4305 - roc_auc:
0.65 - ETA: 6s - loss: 0.4303 - roc_auc: 0.6588 - ETA: 1s - loss: 0.4302 - ro
c_auc: 0.659 - 407s 6ms/step - loss: 0.4301 - roc_auc: 0.6591 - val_loss: 0.3
856 - val_roc_auc: 0.7309

Epoch 00002: val_roc_auc improved from 0.72119 to 0.73091, saving model to we
ights_best.hdf5
Epoch 3/10
69918/69918 [==============================] - ETA: 6:04 - loss: 0.4189 - roc
_auc: 0.696 - ETA: 6:11 - loss: 0.4158 - roc_auc: 0.698 - ETA: 6:14 - loss:
0.4168 - roc_auc: 0.693 - ETA: 6:14 - loss: 0.4149 - roc_auc: 0.687 - ETA: 6:
04 - loss: 0.4167 - roc_auc: 0.682 - ETA: 5:54 - loss: 0.4168 - roc_auc: 0.68
4 - ETA: 5:48 - loss: 0.4194 - roc_auc: 0.681 - ETA: 5:44 - loss: 0.4210 - ro
c_auc: 0.685 - ETA: 5:39 - loss: 0.4209 - roc_auc: 0.681 - ETA: 5:34 - loss:
0.4208 - roc_auc: 0.679 - ETA: 5:28 - loss: 0.4198 - roc_auc: 0.678 - ETA: 5:
24 - loss: 0.4180 - roc_auc: 0.680 - ETA: 5:18 - loss: 0.4154 - roc_auc: 0.68
0 - ETA: 5:14 - loss: 0.4156 - roc_auc: 0.683 - ETA: 5:09 - loss: 0.4150 - ro
c_auc: 0.685 - ETA: 5:04 - loss: 0.4160 - roc_auc: 0.684 - ETA: 4:57 - loss:
0.4136 - roc_auc: 0.685 - ETA: 4:53 - loss: 0.4133 - roc_auc: 0.686 - ETA: 4:
46 - loss: 0.4116 - roc_auc: 0.688 - ETA: 4:41 - loss: 0.4117 - roc_auc: 0.68
```

```
7 - ETA: 4:35 - loss: 0.4114 - roc_auc: 0.689 - ETA: 4:31 - loss: 0.4114 - ro
c_auc: 0.688 - ETA: 4:25 - loss: 0.4112 - roc_auc: 0.688 - ETA: 4:22 - loss:
0.4111 - roc_auc: 0.688 - ETA: 4:16 - loss: 0.4113 - roc_auc: 0.686 - ETA: 4:
09 - loss: 0.4117 - roc_auc: 0.686 - ETA: 4:03 - loss: 0.4116 - roc_auc: 0.68
6 - ETA: 3:57 - loss: 0.4122 - roc_auc: 0.685 - ETA: 3:51 - loss: 0.4123 - ro
c_auc: 0.685 - ETA: 3:45 - loss: 0.4117 - roc_auc: 0.686 - ETA: 3:41 - loss:
0.4113 - roc_auc: 0.685 - ETA: 3:35 - loss: 0.4109 - roc_auc: 0.685 - ETA: 3:
30 - loss: 0.4106 - roc_auc: 0.685 - ETA: 3:24 - loss: 0.4106 - roc_auc: 0.68
6 - ETA: 3:18 - loss: 0.4103 - roc_auc: 0.687 - ETA: 3:12 - loss: 0.4095 - ro
c_auc: 0.686 - ETA: 3:07 - loss: 0.4100 - roc_auc: 0.686 - ETA: 3:01 - loss:
0.4100 - roc_auc: 0.686 - ETA: 2:55 - loss: 0.4107 - roc_auc: 0.686 - ETA: 2:
49 - loss: 0.4112 - roc_auc: 0.685 - ETA: 2:43 - loss: 0.4111 - roc_auc: 0.68
7 - ETA: 2:37 - loss: 0.4104 - roc_auc: 0.688 - ETA: 2:31 - loss: 0.4103 - ro
c_auc: 0.688 - ETA: 2:26 - loss: 0.4103 - roc_auc: 0.688 - ETA: 2:20 - loss:
0.4102 - roc_auc: 0.689 - ETA: 2:13 - loss: 0.4109 - roc_auc: 0.688 - ETA: 2:
07 - loss: 0.4105 - roc_auc: 0.688 - ETA: 2:01 - loss: 0.4102 - roc_auc: 0.68
8 - ETA: 1:55 - loss: 0.4102 - roc_auc: 0.688 - ETA: 1:50 - loss: 0.4111 - ro
c_auc: 0.688 - ETA: 1:44 - loss: 0.4110 - roc_auc: 0.688 - ETA: 1:38 - loss:
0.4112 - roc_auc: 0.688 - ETA: 1:32 - loss: 0.4115 - roc_auc: 0.688 - ETA: 1:
27 - loss: 0.4113 - roc_auc: 0.689 - ETA: 1:20 - loss: 0.4114 - roc_auc: 0.68
9 - ETA: 1:14 - loss: 0.4115 - roc_auc: 0.688 - ETA: 1:09 - loss: 0.4118 - ro
c_auc: 0.688 - ETA: 1:02 - loss: 0.4124 - roc_auc: 0.688 - ETA: 56s - loss:
0.4120 - roc_auc: 0.688 - ETA: 50s - loss: 0.4115 - roc_auc: 0.68 - ETA: 44s
- loss: 0.4122 - roc_auc: 0.68 - ETA: 38s - loss: 0.4118 - roc_auc: 0.68 - ET
A: 32s - loss: 0.4116 - roc_auc: 0.68 - ETA: 26s - loss: 0.4115 - roc_auc: 0.
69 - ETA: 20s - loss: 0.4115 - roc_auc: 0.69 - ETA: 14s - loss: 0.4115 - roc_
auc: 0.69 - ETA: 7s - loss: 0.4115 - roc_auc: 0.6909 - ETA: 1s - loss: 0.4120
- roc_auc: 0.690 - 472s 7ms/step - loss: 0.4122 - roc_auc: 0.6908 - val_loss:
0.3820 - val_roc_auc: 0.7361

Epoch 00003: val_roc_auc improved from 0.73091 to 0.73612, saving model to we
ights_best.hdf5
Epoch 4/10
69918/69918 [==============================] - ETA: 7:47 - loss: 0.4019 - roc
_auc: 0.692 - ETA: 7:21 - loss: 0.4000 - roc_auc: 0.709 - ETA: 6:44 - loss:
0.4052 - roc_auc: 0.702 - ETA: 6:39 - loss: 0.4098 - roc_auc: 0.696 - ETA: 6:
22 - loss: 0.4125 - roc_auc: 0.697 - ETA: 6:16 - loss: 0.4113 - roc_auc: 0.69
9 - ETA: 6:08 - loss: 0.4075 - roc_auc: 0.698 - ETA: 5:57 - loss: 0.4081 - ro
c_auc: 0.698 - ETA: 5:50 - loss: 0.4082 - roc_auc: 0.699 - ETA: 5:43 - loss:
0.4052 - roc_auc: 0.701 - ETA: 5:39 - loss: 0.4038 - roc_auc: 0.699 - ETA: 5:
31 - loss: 0.4037 - roc_auc: 0.700 - ETA: 5:25 - loss: 0.4036 - roc_auc: 0.70
0 - ETA: 5:20 - loss: 0.4020 - roc_auc: 0.703 - ETA: 5:17 - loss: 0.4020 - ro
c_auc: 0.702 - ETA: 5:10 - loss: 0.4010 - roc_auc: 0.705 - ETA: 5:05 - loss:
0.4016 - roc_auc: 0.704 - ETA: 5:00 - loss: 0.4021 - roc_auc: 0.703 - ETA: 4:
55 - loss: 0.4029 - roc_auc: 0.702 - ETA: 4:51 - loss: 0.4032 - roc_auc: 0.70
2 - ETA: 4:46 - loss: 0.4039 - roc_auc: 0.700 - ETA: 4:39 - loss: 0.4044 - ro
c_auc: 0.700 - ETA: 4:32 - loss: 0.4053 - roc_auc: 0.700 - ETA: 4:26 - loss:
0.4050 - roc_auc: 0.701 - ETA: 4:19 - loss: 0.4049 - roc_auc: 0.701 - ETA: 4:
13 - loss: 0.4048 - roc_auc: 0.703 - ETA: 4:07 - loss: 0.4039 - roc_auc: 0.70
4 - ETA: 4:01 - loss: 0.4038 - roc_auc: 0.705 - ETA: 3:56 - loss: 0.4025 - ro
c_auc: 0.707 - ETA: 3:50 - loss: 0.4034 - roc_auc: 0.707 - ETA: 3:44 - loss:
0.4025 - roc_auc: 0.708 - ETA: 3:37 - loss: 0.4009 - roc_auc: 0.709 - ETA: 3:
31 - loss: 0.4018 - roc_auc: 0.708 - ETA: 3:25 - loss: 0.4020 - roc_auc: 0.70
7 - ETA: 3:19 - loss: 0.4019 - roc_auc: 0.708 - ETA: 3:13 - loss: 0.4013 - ro
c_auc: 0.709 - ETA: 3:07 - loss: 0.4013 - roc_auc: 0.710 - ETA: 3:01 - loss:
0.4013 - roc_auc: 0.710 - ETA: 2:55 - loss: 0.4003 - roc_auc: 0.711 - ETA: 2:
49 - loss: 0.3996 - roc_auc: 0.712 - ETA: 2:43 - loss: 0.3992 - roc_auc: 0.71
```

```
2 - ETA: 2:36 - loss: 0.3985 - roc_auc: 0.712 - ETA: 2:30 - loss: 0.3989 - ro
c_auc: 0.713 - ETA: 2:24 - loss: 0.3983 - roc_auc: 0.713 - ETA: 2:19 - loss:
0.3979 - roc_auc: 0.714 - ETA: 2:13 - loss: 0.3976 - roc_auc: 0.714 - ETA: 2:
07 - loss: 0.3978 - roc_auc: 0.714 - ETA: 2:01 - loss: 0.3977 - roc_auc: 0.71
5 - ETA: 1:55 - loss: 0.3976 - roc_auc: 0.714 - ETA: 1:49 - loss: 0.3974 - ro
c_auc: 0.715 - ETA: 1:43 - loss: 0.3966 - roc_auc: 0.716 - ETA: 1:37 - loss:
0.3969 - roc_auc: 0.716 - ETA: 1:31 - loss: 0.3967 - roc_auc: 0.716 - ETA: 1:
25 - loss: 0.3971 - roc_auc: 0.716 - ETA: 1:19 - loss: 0.3964 - roc_auc: 0.71
6 - ETA: 1:13 - loss: 0.3961 - roc_auc: 0.716 - ETA: 1:07 - loss: 0.3961 - ro
c_auc: 0.716 - ETA: 1:02 - loss: 0.3961 - roc_auc: 0.717 - ETA: 56s - loss:
0.3961 - roc_auc: 0.718 - ETA: 50s - loss: 0.3967 - roc_auc: 0.71 - ETA: 44s
- loss: 0.3964 - roc_auc: 0.71 - ETA: 38s - loss: 0.3963 - roc_auc: 0.71 - ET
A: 32s - loss: 0.3960 - roc_auc: 0.71 - ETA: 26s - loss: 0.3961 - roc_auc: 0.
71 - ETA: 19s - loss: 0.3958 - roc_auc: 0.71 - ETA: 13s - loss: 0.3965 - roc_
auc: 0.71 - ETA: 7s - loss: 0.3971 - roc_auc: 0.7175 - ETA: 1s - loss: 0.3968
- roc_auc: 0.717 - 468s 7ms/step - loss: 0.3968 - roc_auc: 0.7177 - val_loss:
0.3848 - val_roc_auc: 0.7336

Epoch 00004: val_roc_auc did not improve from 0.73612
Epoch 5/10

69918/69918 [==============================] - ETA: 7:26 - loss: 0.3734 - roc_a
uc: 0.736 - ETA: 7:07 - loss: 0.3825 - roc_auc: 0.739 - ETA: 6:53 - loss: 0.394
3 - roc_auc: 0.725 - ETA: 6:47 - loss: 0.3895 - roc_auc: 0.729 - ETA: 6:56 - lo
ss: 0.3914 - roc_auc: 0.731 - ETA: 6:50 - loss: 0.3960 - roc_auc: 0.729 - ETA:
6:45 - loss: 0.3987 - roc_auc: 0.727 - ETA: 6:39 - loss: 0.3994 - roc_auc: 0.72
9 - ETA: 6:35 - loss: 0.4012 - roc_auc: 0.723 - ETA: 6:33 - loss: 0.3992 - roc_
auc: 0.724 - ETA: 6:31 - loss: 0.3984 - roc_auc: 0.723 - ETA: 6:30 - loss: 0.39
83 - roc_auc: 0.724 - ETA: 6:27 - loss: 0.3992 - roc_auc: 0.721 - ETA: 6:20 - l
oss: 0.3979 - roc_auc: 0.719 - ETA: 6:14 - loss: 0.3983 - roc_auc: 0.721 - ETA:
6:05 - loss: 0.3963 - roc_auc: 0.723 - ETA: 5:58 - loss: 0.3946 - roc_auc: 0.72
4 - ETA: 5:51 - loss: 0.3933 - roc_auc: 0.726 - ETA: 5:44 - loss: 0.3944 - roc_
auc: 0.725 - ETA: 5:36 - loss: 0.3951 - roc_auc: 0.724 - ETA: 5:30 - loss: 0.39
42 - roc_auc: 0.725 - ETA: 5:26 - loss: 0.3929 - roc_auc: 0.725 - ETA: 5:21 - l
oss: 0.3916 - roc_auc: 0.726 - ETA: 5:17 - loss: 0.3900 - roc_auc: 0.726 - ETA:
5:09 - loss: 0.3898 - roc_auc: 0.727 - ETA: 5:02 - loss: 0.3906 - roc_auc: 0.72
6 - ETA: 4:56 - loss: 0.3892 - roc_auc: 0.726 - ETA: 4:50 - loss: 0.3909 - roc_
auc: 0.725 - ETA: 4:42 - loss: 0.3907 - roc_auc: 0.724 - ETA: 4:34 - loss: 0.39
05 - roc_auc: 0.725 - ETA: 4:26 - loss: 0.3901 - roc_auc: 0.726 - ETA: 4:17 - l
oss: 0.3895 - roc_auc: 0.727 - ETA: 4:09 - loss: 0.3887 - roc_auc: 0.727 - ETA:
4:02 - loss: 0.3888 - roc_auc: 0.728 - ETA: 3:54 - loss: 0.3872 - roc_auc: 0.72
9 - ETA: 3:45 - loss: 0.3881 - roc_auc: 0.729 - ETA: 3:38 - loss: 0.3890 - roc_
auc: 0.728 - ETA: 3:30 - loss: 0.3885 - roc_auc: 0.729 - ETA: 3:23 - loss: 0.38
82 - roc_auc: 0.730 - ETA: 3:16 - loss: 0.3882 - roc_auc: 0.729 - ETA: 3:09 - l
oss: 0.3880 - roc_auc: 0.729 - ETA: 3:02 - loss: 0.3874 - roc_auc: 0.729 - ETA:
2:56 - loss: 0.3876 - roc_auc: 0.729 - ETA: 2:50 - loss: 0.3879 - roc_auc: 0.72
8 - ETA: 2:44 - loss: 0.3880 - roc_auc: 0.728 - ETA: 2:37 - loss: 0.3879 - roc_
auc: 0.729 - ETA: 2:30 - loss: 0.3878 - roc_auc: 0.730 - ETA: 2:22 - loss: 0.38
77 - roc_auc: 0.730 - ETA: 2:16 - loss: 0.3883 - roc_auc: 0.730 - ETA: 2:08 - l
oss: 0.3879 - roc_auc: 0.731 - ETA: 2:01 - loss: 0.3881 - roc_auc: 0.731 - ETA:
1:54 - loss: 0.3883 - roc_auc: 0.731 - ETA: 1:47 - loss: 0.3875 - roc_auc: 0.73
1 - ETA: 1:40 - loss: 0.3874 - roc_auc: 0.732 - ETA: 1:33 - loss: 0.3879 - roc_
auc: 0.731 - ETA: 1:26 - loss: 0.3885 - roc_auc: 0.731 - ETA: 1:19 - loss: 0.38
85 - roc_auc: 0.731 - ETA: 1:12 - loss: 0.3888 - roc_auc: 0.732 - ETA: 1:04 - l
oss: 0.3892 - roc_auc: 0.731 - ETA: 57s - loss: 0.3893 - roc_auc: 0.732 - ETA:
50s - loss: 0.3888 - roc_auc: 0.73 - ETA: 43s - loss: 0.3886 - roc_auc: 0.73 -
ETA: 36s - loss: 0.3889 - roc_auc: 0.73 - ETA: 30s - loss: 0.3892 - roc_auc: 0.
```

```
73 - ETA: 23s - loss: 0.3895 - roc_auc: 0.73 - ETA: 16s - loss: 0.3894 - roc_au
c: 0.73 - ETA: 9s - loss: 0.3899 - roc_auc: 0.7327 - ETA: 1s - loss: 0.3898 - r
oc_auc: 0.733 - 542s 8ms/step - loss: 0.3898 - roc_auc: 0.7329 - val_loss: 0.38
53 - val_roc_auc: 0.7324

Epoch 00005: val_roc_auc did not improve from 0.73612
Epoch 6/10
69918/69918 [==============================] - ETA: 7:50 - loss: 0.3892 - roc_a
uc: 0.772 - ETA: 8:22 - loss: 0.3791 - roc_auc: 0.773 - ETA: 8:42 - loss: 0.393
2 - roc_auc: 0.757 - ETA: 8:11 - loss: 0.3954 - roc_auc: 0.745 - ETA: 8:11 - lo
ss: 0.3923 - roc_auc: 0.743 - ETA: 7:55 - loss: 0.3928 - roc_auc: 0.748 - ETA:
7:45 - loss: 0.3922 - roc_auc: 0.749 - ETA: 7:33 - loss: 0.3885 - roc_auc: 0.75
1 - ETA: 7:19 - loss: 0.3882 - roc_auc: 0.751 - ETA: 7:04 - loss: 0.3849 - roc_
auc: 0.751 - ETA: 7:00 - loss: 0.3829 - roc_auc: 0.752 - ETA: 6:54 - loss: 0.38
15 - roc_auc: 0.751 - ETA: 6:53 - loss: 0.3815 - roc_auc: 0.748 - ETA: 6:48 - l
oss: 0.3804 - roc_auc: 0.750 - ETA: 6:43 - loss: 0.3803 - roc_auc: 0.750 - ETA:
6:32 - loss: 0.3820 - roc_auc: 0.748 - ETA: 6:23 - loss: 0.3819 - roc_auc: 0.75
0 - ETA: 6:18 - loss: 0.3818 - roc_auc: 0.751 - ETA: 6:12 - loss: 0.3818 - roc_
auc: 0.752 - ETA: 6:06 - loss: 0.3808 - roc_auc: 0.752 - ETA: 6:02 - loss: 0.38
02 - roc_auc: 0.751 - ETA: 5:54 - loss: 0.3814 - roc_auc: 0.749 - ETA: 5:46 - l
oss: 0.3814 - roc_auc: 0.749 - ETA: 5:38 - loss: 0.3831 - roc_auc: 0.748 - ETA:
5:28 - loss: 0.3821 - roc_auc: 0.749 - ETA: 5:20 - loss: 0.3821 - roc_auc: 0.75
0 - ETA: 5:12 - loss: 0.3821 - roc_auc: 0.750 - ETA: 5:05 - loss: 0.3821 - roc_
auc: 0.751 - ETA: 5:00 - loss: 0.3822 - roc_auc: 0.751 - ETA: 4:53 - loss: 0.38
16 - roc_auc: 0.752 - ETA: 4:46 - loss: 0.3813 - roc_auc: 0.752 - ETA: 4:39 - l
oss: 0.3807 - roc_auc: 0.752 - ETA: 4:32 - loss: 0.3811 - roc_auc: 0.752 - ETA:
4:25 - loss: 0.3800 - roc_auc: 0.752 - ETA: 4:17 - loss: 0.3802 - roc_auc: 0.75
2 - ETA: 4:09 - loss: 0.3798 - roc_auc: 0.752 - ETA: 4:01 - loss: 0.3796 - roc_
auc: 0.752 - ETA: 3:53 - loss: 0.3795 - roc_auc: 0.752 - ETA: 3:46 - loss: 0.37
95 - roc_auc: 0.752 - ETA: 3:39 - loss: 0.3790 - roc_auc: 0.752 - ETA: 3:32 - l
oss: 0.3803 - roc_auc: 0.751 - ETA: 3:25 - loss: 0.3797 - roc_auc: 0.752 - ETA:
3:19 - loss: 0.3791 - roc_auc: 0.752 - ETA: 3:12 - loss: 0.3795 - roc_auc: 0.75
1 - ETA: 3:04 - loss: 0.3798 - roc_auc: 0.751 - ETA: 2:56 - loss: 0.3801 - roc_
auc: 0.751 - ETA: 2:48 - loss: 0.3806 - roc_auc: 0.750 - ETA: 2:39 - loss: 0.38
04 - roc_auc: 0.751 - ETA: 2:30 - loss: 0.3805 - roc_auc: 0.750 - ETA: 2:22 - l
oss: 0.3799 - roc_auc: 0.751 - ETA: 2:14 - loss: 0.3799 - roc_auc: 0.751 - ETA:
2:05 - loss: 0.3800 - roc_auc: 0.751 - ETA: 1:57 - loss: 0.3796 - roc_auc: 0.75
1 - ETA: 1:49 - loss: 0.3800 - roc_auc: 0.751 - ETA: 1:41 - loss: 0.3794 - roc_
auc: 0.751 - ETA: 1:33 - loss: 0.3788 - roc_auc: 0.752 - ETA: 1:25 - loss: 0.37
84 - roc_auc: 0.752 - ETA: 1:18 - loss: 0.3785 - roc_auc: 0.752 - ETA: 1:10 - l
oss: 0.3786 - roc_auc: 0.752 - ETA: 1:02 - loss: 0.3786 - roc_auc: 0.752 - ETA:
55s - loss: 0.3793 - roc_auc: 0.751 - ETA: 47s - loss: 0.3796 - roc_auc: 0.75 -
ETA: 39s - loss: 0.3796 - roc_auc: 0.75 - ETA: 32s - loss: 0.3800 - roc_auc: 0.
75 - ETA: 24s - loss: 0.3801 - roc_auc: 0.75 - ETA: 17s - loss: 0.3803 - roc_au
c: 0.75 - ETA: 9s - loss: 0.3808 - roc_auc: 0.7514 - ETA: 2s - loss: 0.3808 - r
oc_auc: 0.751 - 554s 8ms/step - loss: 0.3806 - roc_auc: 0.7514 - val_loss: 0.38
63 - val_roc_auc: 0.7327

Epoch 00006: val_roc_auc did not improve from 0.73612
Epoch 7/10
69918/69918 [==============================] - ETA: 6:39 - loss: 0.4008 - roc_a
uc: 0.726 - ETA: 6:41 - loss: 0.3919 - roc_auc: 0.744 - ETA: 7:06 - loss: 0.386
0 - roc_auc: 0.747 - ETA: 7:16 - loss: 0.3797 - roc_auc: 0.756 - ETA: 7:17 - lo
ss: 0.3823 - roc_auc: 0.757 - ETA: 7:13 - loss: 0.3789 - roc_auc: 0.759 - ETA:
7:12 - loss: 0.3744 - roc_auc: 0.763 - ETA: 7:00 - loss: 0.3764 - roc_auc: 0.75
9 - ETA: 6:48 - loss: 0.3744 - roc_auc: 0.763 - ETA: 6:39 - loss: 0.3730 - roc_
auc: 0.764 - ETA: 6:28 - loss: 0.3725 - roc_auc: 0.767 - ETA: 6:27 - loss: 0.37
```

```
21 - roc_auc: 0.767 - ETA: 6:22 - loss: 0.3726 - roc_auc: 0.764 - ETA: 6:11 - l
oss: 0.3736 - roc_auc: 0.765 - ETA: 6:03 - loss: 0.3732 - roc_auc: 0.766 - ETA:
5:55 - loss: 0.3726 - roc_auc: 0.765 - ETA: 5:47 - loss: 0.3736 - roc_auc: 0.76
6 - ETA: 5:38 - loss: 0.3728 - roc_auc: 0.767 - ETA: 5:34 - loss: 0.3716 - roc_
auc: 0.768 - ETA: 5:26 - loss: 0.3720 - roc_auc: 0.769 - ETA: 5:19 - loss: 0.37
25 - roc_auc: 0.769 - ETA: 5:11 - loss: 0.3723 - roc_auc: 0.770 - ETA: 5:03 - l
oss: 0.3730 - roc_auc: 0.770 - ETA: 4:55 - loss: 0.3734 - roc_auc: 0.769 - ETA:
4:48 - loss: 0.3727 - roc_auc: 0.770 - ETA: 4:40 - loss: 0.3727 - roc_auc: 0.77
1 - ETA: 4:33 - loss: 0.3735 - roc_auc: 0.771 - ETA: 4:25 - loss: 0.3729 - roc_
auc: 0.772 - ETA: 4:18 - loss: 0.3743 - roc_auc: 0.771 - ETA: 4:11 - loss: 0.37
35 - roc_auc: 0.772 - ETA: 4:05 - loss: 0.3735 - roc_auc: 0.772 - ETA: 3:58 - l
oss: 0.3731 - roc_auc: 0.773 - ETA: 3:51 - loss: 0.3722 - roc_auc: 0.773 - ETA:
3:44 - loss: 0.3721 - roc_auc: 0.773 - ETA: 3:37 - loss: 0.3729 - roc_auc: 0.77
2 - ETA: 3:30 - loss: 0.3715 - roc_auc: 0.773 - ETA: 3:24 - loss: 0.3715 - roc_
auc: 0.772 - ETA: 3:17 - loss: 0.3718 - roc_auc: 0.772 - ETA: 3:10 - loss: 0.37
15 - roc_auc: 0.772 - ETA: 3:04 - loss: 0.3709 - roc_auc: 0.773 - ETA: 2:57 - l
oss: 0.3712 - roc_auc: 0.772 - ETA: 2:50 - loss: 0.3709 - roc_auc: 0.773 - ETA:
2:44 - loss: 0.3715 - roc_auc: 0.772 - ETA: 2:38 - loss: 0.3714 - roc_auc: 0.77
2 - ETA: 2:32 - loss: 0.3718 - roc_auc: 0.772 - ETA: 2:27 - loss: 0.3718 - roc_
auc: 0.772 - ETA: 2:20 - loss: 0.3714 - roc_auc: 0.772 - ETA: 2:15 - loss: 0.37
19 - roc_auc: 0.772 - ETA: 2:08 - loss: 0.3716 - roc_auc: 0.772 - ETA: 2:02 - l
oss: 0.3709 - roc_auc: 0.773 - ETA: 1:55 - loss: 0.3710 - roc_auc: 0.772 - ETA:
1:48 - loss: 0.3711 - roc_auc: 0.772 - ETA: 1:42 - loss: 0.3711 - roc_auc: 0.77
1 - ETA: 1:35 - loss: 0.3710 - roc_auc: 0.771 - ETA: 1:28 - loss: 0.3706 - roc_
auc: 0.772 - ETA: 1:21 - loss: 0.3710 - roc_auc: 0.772 - ETA: 1:15 - loss: 0.37
08 - roc_auc: 0.772 - ETA: 1:08 - loss: 0.3701 - roc_auc: 0.772 - ETA: 1:01 - l
oss: 0.3703 - roc_auc: 0.772 - ETA: 55s - loss: 0.3702 - roc_auc: 0.772 - ETA:
48s - loss: 0.3698 - roc_auc: 0.77 - ETA: 41s - loss: 0.3699 - roc_auc: 0.77 -
ETA: 35s - loss: 0.3698 - roc_auc: 0.77 - ETA: 28s - loss: 0.3700 - roc_auc: 0.
77 - ETA: 21s - loss: 0.3699 - roc_auc: 0.77 - ETA: 15s - loss: 0.3700 - roc_au
c: 0.77 - ETA: 8s - loss: 0.3698 - roc_auc: 0.7709 - ETA: 1s - loss: 0.3702 - r
oc_auc: 0.770 - 502s 7ms/step - loss: 0.3704 - roc_auc: 0.7703 - val_loss: 0.39
36 - val_roc_auc: 0.7214

Epoch 00007: val_roc_auc did not improve from 0.73612
Epoch 8/10
69918/69918 [==============================] - ETA: 7:01 - loss: 0.3861 - roc_a
uc: 0.770 - ETA: 6:51 - loss: 0.3641 - roc_auc: 0.792 - ETA: 6:52 - loss: 0.365
1 - roc_auc: 0.793 - ETA: 6:46 - loss: 0.3729 - roc_auc: 0.791 - ETA: 6:39 - lo
ss: 0.3658 - roc_auc: 0.792 - ETA: 6:46 - loss: 0.3586 - roc_auc: 0.795 - ETA:
6:47 - loss: 0.3599 - roc_auc: 0.796 - ETA: 6:43 - loss: 0.3599 - roc_auc: 0.79
5 - ETA: 6:46 - loss: 0.3583 - roc_auc: 0.798 - ETA: 6:36 - loss: 0.3561 - roc_
auc: 0.800 - ETA: 6:27 - loss: 0.3571 - roc_auc: 0.799 - ETA: 6:20 - loss: 0.35
75 - roc_auc: 0.796 - ETA: 6:11 - loss: 0.3544 - roc_auc: 0.796 - ETA: 6:03 - l
oss: 0.3558 - roc_auc: 0.794 - ETA: 5:55 - loss: 0.3562 - roc_auc: 0.794 - ETA:
5:47 - loss: 0.3572 - roc_auc: 0.794 - ETA: 5:41 - loss: 0.3578 - roc_auc: 0.79
3 - ETA: 5:35 - loss: 0.3580 - roc_auc: 0.793 - ETA: 5:28 - loss: 0.3593 - roc_
auc: 0.792 - ETA: 5:20 - loss: 0.3578 - roc_auc: 0.792 - ETA: 5:12 - loss: 0.35
79 - roc_auc: 0.790 - ETA: 5:05 - loss: 0.3583 - roc_auc: 0.789 - ETA: 4:57 - l
oss: 0.3576 - roc_auc: 0.790 - ETA: 4:51 - loss: 0.3573 - roc_auc: 0.790 - ETA:
4:44 - loss: 0.3569 - roc_auc: 0.790 - ETA: 4:37 - loss: 0.3568 - roc_auc: 0.79
0 - ETA: 4:30 - loss: 0.3572 - roc_auc: 0.790 - ETA: 4:23 - loss: 0.3577 - roc_
auc: 0.790 - ETA: 4:16 - loss: 0.3581 - roc_auc: 0.790 - ETA: 4:09 - loss: 0.35
87 - roc_auc: 0.789 - ETA: 4:02 - loss: 0.3590 - roc_auc: 0.787 - ETA: 3:56 - l
oss: 0.3595 - roc_auc: 0.787 - ETA: 3:49 - loss: 0.3591 - roc_auc: 0.787 - ETA:
3:44 - loss: 0.3597 - roc_auc: 0.788 - ETA: 3:37 - loss: 0.3591 - roc_auc: 0.78
9 - ETA: 3:30 - loss: 0.3588 - roc_auc: 0.789 - ETA: 3:23 - loss: 0.3597 - roc_
```

```
auc: 0.789 - ETA: 3:18 - loss: 0.3602 - roc_auc: 0.788 - ETA: 3:12 - loss: 0.36
10 - roc_auc: 0.788 - ETA: 3:05 - loss: 0.3617 - roc_auc: 0.787 - ETA: 2:58 - l
oss: 0.3622 - roc_auc: 0.787 - ETA: 2:52 - loss: 0.3618 - roc_auc: 0.787 - ETA:
2:45 - loss: 0.3612 - roc_auc: 0.787 - ETA: 2:38 - loss: 0.3617 - roc_auc: 0.78
7 - ETA: 2:32 - loss: 0.3619 - roc_auc: 0.787 - ETA: 2:26 - loss: 0.3616 - roc_
auc: 0.787 - ETA: 2:20 - loss: 0.3612 - roc_auc: 0.787 - ETA: 2:13 - loss: 0.36
24 - roc_auc: 0.786 - ETA: 2:07 - loss: 0.3620 - roc_auc: 0.787 - ETA: 2:00 - l
oss: 0.3620 - roc_auc: 0.786 - ETA: 1:54 - loss: 0.3622 - roc_auc: 0.786 - ETA:
1:47 - loss: 0.3618 - roc_auc: 0.787 - ETA: 1:41 - loss: 0.3622 - roc_auc: 0.78
6 - ETA: 1:34 - loss: 0.3621 - roc_auc: 0.786 - ETA: 1:28 - loss: 0.3622 - roc_
auc: 0.786 - ETA: 1:21 - loss: 0.3623 - roc_auc: 0.786 - ETA: 1:14 - loss: 0.36
21 - roc_auc: 0.786 - ETA: 1:08 - loss: 0.3624 - roc_auc: 0.786 - ETA: 1:01 - l
oss: 0.3624 - roc_auc: 0.786 - ETA: 55s - loss: 0.3622 - roc_auc: 0.787 - ETA:
48s - loss: 0.3623 - roc_auc: 0.78 - ETA: 41s - loss: 0.3617 - roc_auc: 0.78 -
ETA: 35s - loss: 0.3616 - roc_auc: 0.78 - ETA: 28s - loss: 0.3615 - roc_auc: 0.
78 - ETA: 21s - loss: 0.3619 - roc_auc: 0.78 - ETA: 15s - loss: 0.3623 - roc_au
c: 0.78 - ETA: 8s - loss: 0.3626 - roc_auc: 0.7870 - ETA: 1s - loss: 0.3625 - r
oc_auc: 0.787 - 500s 7ms/step - loss: 0.3623 - roc_auc: 0.7871 - val_loss: 0.39
72 - val_roc_auc: 0.7206

Epoch 00008: val_roc_auc did not improve from 0.73612
Epoch 9/10

69918/69918 [==============================] - ETA: 6:58 - loss: 0.3156 - roc_a
uc: 0.813 - ETA: 6:47 - loss: 0.3246 - roc_auc: 0.808 - ETA: 6:41 - loss: 0.345
0 - roc_auc: 0.798 - ETA: 6:35 - loss: 0.3522 - roc_auc: 0.795 - ETA: 6:29 - lo
ss: 0.3485 - roc_auc: 0.798 - ETA: 6:30 - loss: 0.3490 - roc_auc: 0.799 - ETA:
6:24 - loss: 0.3518 - roc_auc: 0.804 - ETA: 6:22 - loss: 0.3525 - roc_auc: 0.80
4 - ETA: 6:20 - loss: 0.3510 - roc_auc: 0.804 - ETA: 6:14 - loss: 0.3541 - roc_
auc: 0.802 - ETA: 6:12 - loss: 0.3528 - roc_auc: 0.802 - ETA: 6:14 - loss: 0.35
36 - roc_auc: 0.803 - ETA: 6:08 - loss: 0.3523 - roc_auc: 0.804 - ETA: 6:03 - l
oss: 0.3545 - roc_auc: 0.801 - ETA: 5:59 - loss: 0.3563 - roc_auc: 0.801 - ETA:
5:59 - loss: 0.3564 - roc_auc: 0.800 - ETA: 5:51 - loss: 0.3555 - roc_auc: 0.80
0 - ETA: 5:43 - loss: 0.3548 - roc_auc: 0.801 - ETA: 5:38 - loss: 0.3549 - roc_
auc: 0.801 - ETA: 5:30 - loss: 0.3547 - roc_auc: 0.802 - ETA: 5:24 - loss: 0.35
63 - roc_auc: 0.801 - ETA: 5:16 - loss: 0.3565 - roc_auc: 0.800 - ETA: 5:13 - l
oss: 0.3572 - roc_auc: 0.800 - ETA: 5:07 - loss: 0.3563 - roc_auc: 0.801 - ETA:
5:00 - loss: 0.3558 - roc_auc: 0.801 - ETA: 4:52 - loss: 0.3547 - roc_auc: 0.80
0 - ETA: 4:44 - loss: 0.3549 - roc_auc: 0.800 - ETA: 4:36 - loss: 0.3547 - roc_
auc: 0.800 - ETA: 4:28 - loss: 0.3541 - roc_auc: 0.800 - ETA: 4:20 - loss: 0.35
36 - roc_auc: 0.801 - ETA: 4:13 - loss: 0.3530 - roc_auc: 0.801 - ETA: 4:06 - l
oss: 0.3524 - roc_auc: 0.801 - ETA: 3:58 - loss: 0.3511 - roc_auc: 0.802 - ETA:
3:51 - loss: 0.3526 - roc_auc: 0.801 - ETA: 3:44 - loss: 0.3519 - roc_auc: 0.80
2 - ETA: 3:37 - loss: 0.3518 - roc_auc: 0.802 - ETA: 3:30 - loss: 0.3524 - roc_
auc: 0.802 - ETA: 3:22 - loss: 0.3522 - roc_auc: 0.802 - ETA: 3:15 - loss: 0.35
27 - roc_auc: 0.801 - ETA: 3:08 - loss: 0.3530 - roc_auc: 0.801 - ETA: 3:01 - l
oss: 0.3525 - roc_auc: 0.801 - ETA: 2:55 - loss: 0.3523 - roc_auc: 0.801 - ETA:
2:48 - loss: 0.3530 - roc_auc: 0.801 - ETA: 2:41 - loss: 0.3533 - roc_auc: 0.80
1 - ETA: 2:34 - loss: 0.3529 - roc_auc: 0.802 - ETA: 2:29 - loss: 0.3527 - roc_
auc: 0.801 - ETA: 2:22 - loss: 0.3527 - roc_auc: 0.801 - ETA: 2:15 - loss: 0.35
23 - roc_auc: 0.802 - ETA: 2:09 - loss: 0.3524 - roc_auc: 0.802 - ETA: 2:02 - l
oss: 0.3520 - roc_auc: 0.802 - ETA: 1:56 - loss: 0.3518 - roc_auc: 0.802 - ETA:
1:49 - loss: 0.3518 - roc_auc: 0.802 - ETA: 1:42 - loss: 0.3523 - roc_auc: 0.80
1 - ETA: 1:35 - loss: 0.3524 - roc_auc: 0.802 - ETA: 1:29 - loss: 0.3524 - roc_
auc: 0.802 - ETA: 1:22 - loss: 0.3524 - roc_auc: 0.802 - ETA: 1:15 - loss: 0.35
23 - roc_auc: 0.802 - ETA: 1:08 - loss: 0.3525 - roc_auc: 0.801 - ETA: 1:02 - l
oss: 0.3527 - roc_auc: 0.801 - ETA: 55s - loss: 0.3530 - roc_auc: 0.802 - ETA:
49s - loss: 0.3528 - roc_auc: 0.80 - ETA: 42s - loss: 0.3528 - roc_auc: 0.80 -
```

```
ETA: 35s - loss: 0.3529 - roc_auc: 0.80 - ETA: 28s - loss: 0.3529 - roc_auc: 0.
80 - ETA: 22s - loss: 0.3530 - roc_auc: 0.80 - ETA: 15s - loss: 0.3531 - roc_au
c: 0.80 - ETA: 8s - loss: 0.3535 - roc_auc: 0.8024 - ETA: 1s - loss: 0.3536 - r
oc_auc: 0.802 - 514s 7ms/step - loss: 0.3537 - roc_auc: 0.8022 - val_loss: 0.40
79 - val_roc_auc: 0.7116

Epoch 00009: val_roc_auc did not improve from 0.73612
Epoch 10/10
69918/69918 [==============================] - ETA: 8:58 - loss: 0.3294 - roc_a
uc: 0.838 - ETA: 8:18 - loss: 0.3302 - roc_auc: 0.832 - ETA: 8:19 - loss: 0.329
5 - roc_auc: 0.831 - ETA: 8:06 - loss: 0.3252 - roc_auc: 0.830 - ETA: 7:43 - lo
ss: 0.3253 - roc_auc: 0.827 - ETA: 7:34 - loss: 0.3295 - roc_auc: 0.829 - ETA:
7:16 - loss: 0.3276 - roc_auc: 0.832 - ETA: 7:03 - loss: 0.3285 - roc_auc: 0.82
9 - ETA: 6:52 - loss: 0.3287 - roc_auc: 0.831 - ETA: 6:42 - loss: 0.3310 - roc_
auc: 0.833 - ETA: 6:31 - loss: 0.3350 - roc_auc: 0.829 - ETA: 6:21 - loss: 0.33
60 - roc_auc: 0.827 - ETA: 6:16 - loss: 0.3353 - roc_auc: 0.827 - ETA: 6:11 - l
oss: 0.3368 - roc_auc: 0.826 - ETA: 6:01 - loss: 0.3372 - roc_auc: 0.827 - ETA:
5:53 - loss: 0.3379 - roc_auc: 0.827 - ETA: 5:45 - loss: 0.3392 - roc_auc: 0.82
7 - ETA: 5:39 - loss: 0.3404 - roc_auc: 0.826 - ETA: 5:34 - loss: 0.3396 - roc_
auc: 0.826 - ETA: 5:26 - loss: 0.3400 - roc_auc: 0.826 - ETA: 5:18 - loss: 0.33
89 - roc_auc: 0.827 - ETA: 5:10 - loss: 0.3388 - roc_auc: 0.827 - ETA: 5:02 - l
oss: 0.3380 - roc_auc: 0.826 - ETA: 4:55 - loss: 0.3381 - roc_auc: 0.826 - ETA:
4:47 - loss: 0.3371 - roc_auc: 0.826 - ETA: 4:44 - loss: 0.3380 - roc_auc: 0.82
6 - ETA: 4:42 - loss: 0.3377 - roc_auc: 0.827 - ETA: 4:36 - loss: 0.3383 - roc_
auc: 0.826 - ETA: 4:30 - loss: 0.3400 - roc_auc: 0.825 - ETA: 4:22 - loss: 0.34
05 - roc_auc: 0.824 - ETA: 4:16 - loss: 0.3403 - roc_auc: 0.824 - ETA: 4:08 - l
oss: 0.3409 - roc_auc: 0.823 - ETA: 4:03 - loss: 0.3407 - roc_auc: 0.823 - ETA:
3:57 - loss: 0.3408 - roc_auc: 0.823 - ETA: 3:50 - loss: 0.3397 - roc_auc: 0.82
4 - ETA: 3:43 - loss: 0.3400 - roc_auc: 0.824 - ETA: 3:36 - loss: 0.3398 - roc_
auc: 0.825 - ETA: 3:30 - loss: 0.3399 - roc_auc: 0.825 - ETA: 3:25 - loss: 0.34
00 - roc_auc: 0.824 - ETA: 3:19 - loss: 0.3397 - roc_auc: 0.824 - ETA: 3:13 - l
oss: 0.3407 - roc_auc: 0.824 - ETA: 3:06 - loss: 0.3405 - roc_auc: 0.824 - ETA:
2:59 - loss: 0.3410 - roc_auc: 0.824 - ETA: 2:53 - loss: 0.3412 - roc_auc: 0.82
4 - ETA: 2:46 - loss: 0.3406 - roc_auc: 0.824 - ETA: 2:38 - loss: 0.3405 - roc_
auc: 0.824 - ETA: 2:31 - loss: 0.3409 - roc_auc: 0.824 - ETA: 2:24 - loss: 0.34
20 - roc_auc: 0.822 - ETA: 2:18 - loss: 0.3424 - roc_auc: 0.822 - ETA: 2:12 - l
oss: 0.3427 - roc_auc: 0.821 - ETA: 2:05 - loss: 0.3432 - roc_auc: 0.821 - ETA:
1:58 - loss: 0.3433 - roc_auc: 0.821 - ETA: 1:51 - loss: 0.3436 - roc_auc: 0.82
1 - ETA: 1:44 - loss: 0.3436 - roc_auc: 0.820 - ETA: 1:37 - loss: 0.3431 - roc_
auc: 0.821 - ETA: 1:30 - loss: 0.3438 - roc_auc: 0.820 - ETA: 1:23 - loss: 0.34
34 - roc_auc: 0.821 - ETA: 1:15 - loss: 0.3433 - roc_auc: 0.820 - ETA: 1:08 - l
oss: 0.3430 - roc_auc: 0.820 - ETA: 1:01 - loss: 0.3427 - roc_auc: 0.821 - ETA:
53s - loss: 0.3428 - roc_auc: 0.820 - ETA: 46s - loss: 0.3434 - roc_auc: 0.82 -
ETA: 38s - loss: 0.3434 - roc_auc: 0.82 - ETA: 31s - loss: 0.3434 - roc_auc: 0.
82 - ETA: 24s - loss: 0.3438 - roc_auc: 0.82 - ETA: 16s - loss: 0.3436 - roc_au
c: 0.81 - ETA: 9s - loss: 0.3436 - roc_auc: 0.8192 - ETA: 2s - loss: 0.3439 - r
oc_auc: 0.819 - 553s 8ms/step - loss: 0.3437 - roc_auc: 0.8194 - val_loss: 0.42
19 - val_roc_auc: 0.7169

Epoch 00010: val_roc_auc did not improve from 0.73612
```

Out[142]: <keras.callbacks.History at 0x28255b75e48>

In [0]:
```
!pip -- upgrade tensorflow
```

```
ERROR: unknown command "upgrade"
```

In [0]:
```
import tensorflow as tf
from tensorflow.python.keras.callbacks import TensorBoard
import datetime
from keras.callbacks import ModelCheckpoint
```

In [0]:
```
tensorboard = TensorBoard("logs\\fit\\" + datetime.datetime.now().strftime("%
```

In [0]:
```
% load_ext tensorboard
% tensorboard --logdir logs/fit
```

```
The tensorboard module is not an IPython extension.
```

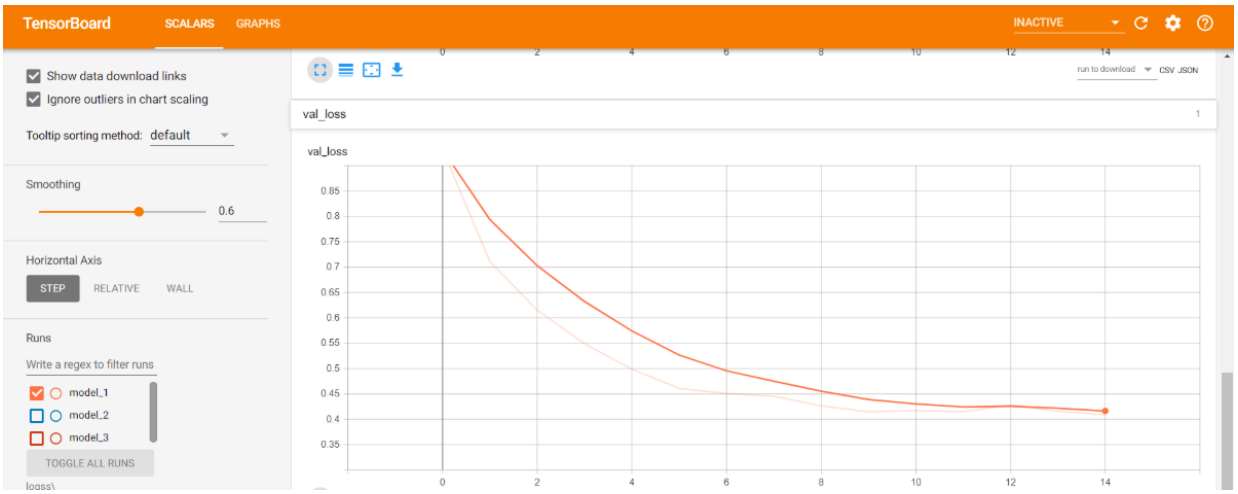
```
UsageError: Line magic function `%tensorboard` not found.
```

In [0]:
```
tf.__version__
```

Out[22]:
```
'1.11.0'
```

In [2]:
```
from IPython.display import Image
Image("Capture2.PNG")
```

Out[2]:



In [0]:
```

```

Model-2

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

1. Train the TF-IDF on the Train data

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

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

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

In [0]:
```
import tensorflow as tf
tf.__version__
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

Out[3]:
```
'1.15.0'
```

In [0]:
```
project_data = pd.read_csv('drive/My Drive/lstm_donor/processed_data.csv')
#target = project_data['']
```

In [0]:
```
project_data.columns
target = project_data['project_is_approved']
#project_data.drop(['Unnamed :0','id','teacher_id','project_is_approved'],axi
```

In [0]:
```
project_data.drop(['Unnamed: 0','id','teacher_id','project_is_approved'],axis
```

In [7]:
```
#splitting the data in train,validation and test
from sklearn.model_selection import train_test_split
#80% training data and 20% test data
X_train,X_test,Y_train,Y_test = train_test_split(project_data,target,test_siz
X_train,X_cv,Y_train,Y_cv = train_test_split(X_train,Y_train,test_size = 0.2,

print('shape of training data after vectorization is:',X_train.shape)
print('shape of validation data after vectorization is:',X_cv.shape)
print('shape of test data after vectorization is:',X_test.shape)
```

```
shape of training data after vectorization is: (69918, 14)
shape of validation data after vectorization is: (17480, 14)
shape of test data after vectorization is: (21850, 14)
```

In [0]:
```

```

TFIDF Model

In [8]:
```
project_data.columns
```

Out[8]:
```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'cleaned_project_titles', 'cleaned_resource_summary', 'essay',
       'cleaned_essay', 'nrm_price'],
      dtype='object')
```

In [0]:
```
#utility function
def final_text(df):
    """Returns the final concatenated text"""
    df['final_Text'] = df['cleaned_essay'].map(str) + " " + df['cleaned_resou
        " " + df['cleaned_project_titles'].map(str)
    return df

#getting the train,val and test data
X_train = final_text(X_train)
X_cv= final_text(X_cv)
X_test = final_text(X_test)
```

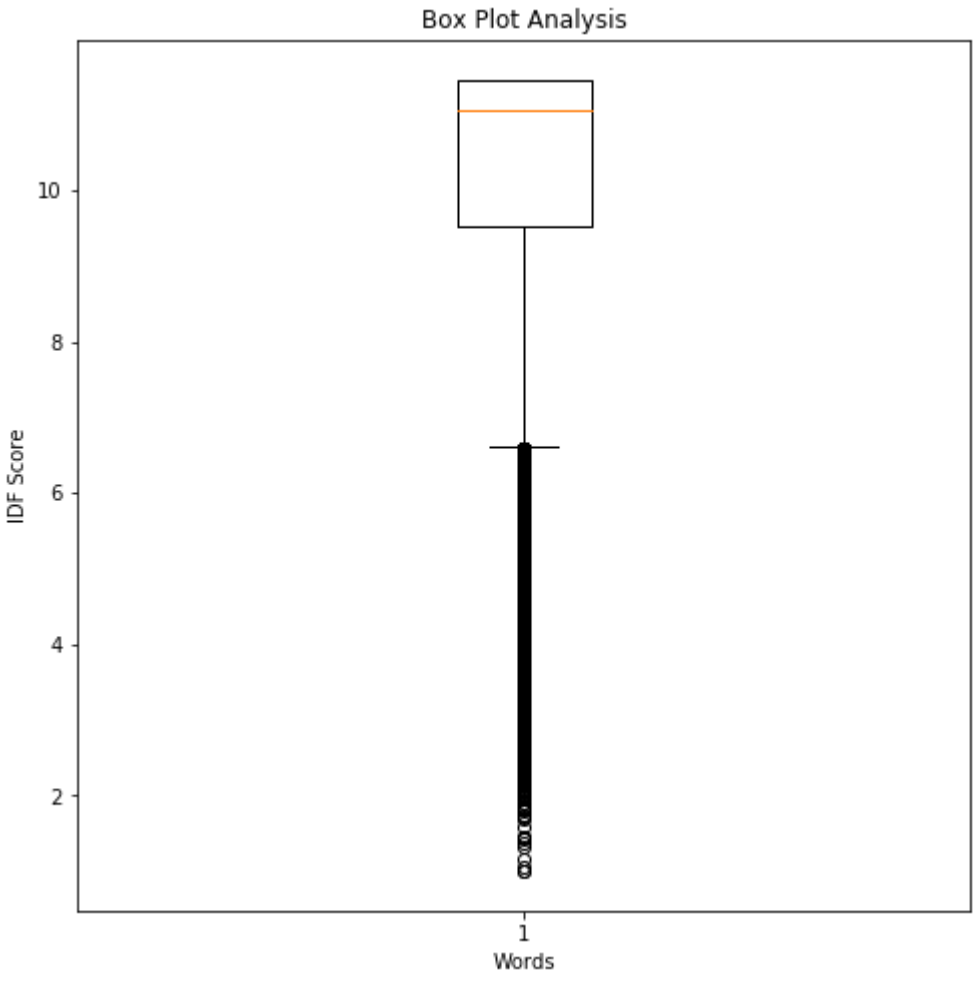
In [10]:
```
X_train.columns
```

Out[10]:
```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'cleaned_project_titles', 'cleaned_resource_summary', 'essay',
       'cleaned_essay', 'nrm_price', 'final_Text'],
      dtype='object')
```

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()

tfidf.fit(X_train['final_Text'])
corpus = tfidf.get_feature_names() #getting names of all the features
idf_corpus = tfidf.idf_ #getting the  idf scores

dict_corpus_idf = dict(zip(corpus,idf_corpus))
print('Number of wordds in the corpus are:',len(dict_corpus_idf))
```

```
Number of wordds in the corpus are: 51124
```

In [0]:
```
p_75th = np.percentile(idf_corpus,59.81896)
p_75th
import matplotlib.pyplot as plt
plt.figure(figsize=(8,8))
plt.boxplot(idf_corpus)
plt.title('Box Plot Analysis')
plt.xlabel('Words')
plt.ylabel('IDF Score')
plt.show()

p_25th = 3
p_75th = np.percentile(idf_corpus,75)
#p_75th = 11.555

print("The lowest significant value of TF-IDF Scores: ",p_25th)
print("The highest significant value of TF-IDF Scores: ",p_75th)
```



```
The lowest significant value of TF-IDF Scores:  3
The highest significant value of TF-IDF Scores:  11.4619455276077
```

In [0]:
```
#creating the list of words to be removed from the corpus
removed_wordlist = []
for word in list(dict_corpus_idf.keys()):
    if(dict_corpus_idf[word] < p_25th or dict_corpus_idf[word] > p_75th):
        removed_wordlist.append(word)
    else:
        continue

print("Number of words to be removed: ",len(removed_wordlist))
```

```
Number of words to be removed:  141
```

In [0]:
```
def remove_from_text(list_of_sentences):
    """This function will be used to remove words from text data"""
    processed_text = []
    for sentence in tqdm(list_of_sentences):
        sent = ' '.join(word for word in sentence.split() if word not in remo
        processed_text.append(sent)
    return processed_text

X_train['total_text'] = remove_from_text(X_train.final_Text.values)
X_cv['total_text'] = remove_from_text(X_cv.final_Text.values)
X_test['total_text'] = remove_from_text(X_test.final_Text.values)
```

```
100%|██████████| 69918/69918 [00:15<00:00, 4432.91it/s]
100%|██████████| 17480/17480 [00:03<00:00, 4512.44it/s]
100%|██████████| 21850/21850 [00:04<00:00, 4496.13it/s]
```

In [0]:
```
X_train.to_csv("X_train_removed.csv", index=False)
X_cv.to_csv("X_val_removed.csv", index=False)
X_test.to_csv("X_test_removed.csv", index=False)
```

In [0]:
```
X_train = pd.read_csv("X_train_removed.csv")
X_val = pd.read_csv("X_val_removed.csv")
X_test = pd.read_csv("X_test_removed.csv")
```

Total Text data

In [0]:
```
#Get the total_text values in list
docs_text_train=list(X_train.total_text.values)
docs_text_val=list(X_val.total_text.values)
docs_text_test=list(X_test.total_text.values)
labels_train=np.array(Y_train)
labels_val=np.array(Y_cv)
labels_test=np.array(Y_test)

#Initializing the keras tokenizer and fitting it on train data
tokens = Tokenizer()
tokens.fit_on_texts(docs_text_train)

#Convert the texts to sequences using the tokenizer
sequences_text_train = tokens.texts_to_sequences(docs_text_train)
sequences_text_val = tokens.texts_to_sequences(docs_text_val)
sequences_text_test = tokens.texts_to_sequences(docs_text_test)
vocab_size_text = len(tokens.word_index) + 1

#Add padding
padded_text_train = pad_sequences(sequences_text_train, maxlen=300, padding='
padded_text_val = pad_sequences(sequences_text_val, maxlen=300, padding='post
padded_text_test = pad_sequences(sequences_text_test, maxlen=300, padding='po
```

In [0]:
```
pickle_in = open("drive/My Drive/lstm_donor/glove_vectors","rb")
glove_words = pickle.load(pickle_in)
embedding_matrix = zeros((vocab_size_text, 300))
for word, i in tokens.word_index.items(): #enumerating all unique words
    embedding_vector = glove_words.get(word)#locating for loaded glove model
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [0]:
```
#Get the flattened LSTM output for input text
import warnings
warnings.filterwarnings('ignore')
input_layer_total_text = Input(shape=(300,), name = "total_text_sequence")
embedding_layer_total_text = Embedding(input_dim=vocab_size_text, output_dim=
lstm_total_text  = LSTM(16, activation="relu", return_sequences=True)(embeddi
flatten_total_text = Flatten()(lstm_total_text)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:541: The name tf.placeholder is deprecated. Please use tf.c
ompat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:66: The name tf.get_default_graph is deprecated. Please use
tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:4432: The name tf.random_uniform is deprecated. Please use
tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:190: The name tf.get_default_session is deprecated. Please
use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please use tf.c
ompat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:203: The name tf.Session is deprecated. Please use tf.compa
t.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:207: The name tf.global_variables is deprecated. Please use
tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:216: The name tf.is_variable_initialized is deprecated. Ple
ase use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:223: The name tf.variables_initializer is deprecated. Pleas
e use tf.compat.v1.variables_initializer instead.
```

In [0]:
```
# teacher prefix
#Get the teacher_prefix values
docs_teacher_prefix_train=list(X_train.teacher_prefix.values)
docs_teacher_prefix_val=list(X_val.teacher_prefix.values)
docs_teacher_prefix_test=list(X_test.teacher_prefix.values)

#Initializing the keras tokenizer and fitting it on train data
tokens = Tokenizer()
tokens.fit_on_texts(docs_teacher_prefix_train)

#Convert the school_state to sequences using the tokenizer
sequences_teacher_prefix_train = np.array(tokens.texts_to_sequences(docs_teac
sequences_teacher_prefix_val = np.array(tokens.texts_to_sequences(docs_teache
sequences_teacher_prefix_test = np.array(tokens.texts_to_sequences(docs_teach
vocab_size_teacher_prefix = len(tokens.word_index) + 1
#Get the flattened output for teacher_prefix
input_layer_teacher_prefix = Input(shape=(1,), name = "teacher_prefix")
embedding_layer_teacher_prefix = Embedding(input_dim=vocab_size_teacher_prefi
flatten_teacher_prefix = Flatten()(embedding_layer_teacher_prefix)
```

Categorical data : school_state

In [0]:
```

#Get the school state values
docs_school_state_train=list(X_train.school_state.values)
docs_school_state_val=list(X_val.school_state.values)
docs_school_state_test=list(X_test.school_state.values)

#Initializing the keras tokenizer and fitting it on train data
tokens = Tokenizer()
tokens.fit_on_texts(docs_school_state_train)

#Convert the school_state to sequences using the tokenizer
sequences_school_train = np.array(tokens.texts_to_sequences(docs_school_state
sequences_school_val = np.array(tokens.texts_to_sequences(docs_school_state_v
sequences_school_test = np.array(tokens.texts_to_sequences(docs_school_state_
vocab_size_school_state = len(tokens.word_index) + 1
#Get the flattened output for school_state
input_layer_school_state = Input(shape=(1,), name = "encoded_school_state")
embedding_layer_school_state = Embedding(input_dim=vocab_size_school_state, o
flatten_school_state = Flatten()(embedding_layer_school_state)

```

Categorical data : project_grade_category

In [0]:
```
#Get the project_grade_category values
docs_project_grade_category_train=list(X_train.project_grade_category.values)
docs_project_grade_category_val=list(X_val.project_grade_category.values)
docs_project_grade_category_test=list(X_test.project_grade_category.values)

#Initializing the keras tokenizer and fitting it on train data
tokens = Tokenizer()
tokens.fit_on_texts(docs_project_grade_category_train)

#Convert the school_state to sequences using the tokenizer
sequences_project_grade_category_train = tokens.texts_to_sequences(docs_proje
sequences_project_grade_category_val = tokens.texts_to_sequences(docs_project
sequences_project_grade_category_test = tokens.texts_to_sequences(docs_projec
vocab_size_project_grade_category= len(tokens.word_index) + 1

#Add padding
padded_project_grade_category_train = pad_sequences(sequences_project_grade_c
padded_project_grade_category_val = pad_sequences(sequences_project_grade_cat
padded_project_grade_category_test = pad_sequences(sequences_project_grade_ca
#Get the flattened output for project_grade_category
input_layer_project_grade = Input(shape=(3,), name = "project_grade_category"
embedding_layer_project_grade = Embedding(input_dim=vocab_size_project_grade_
flatten_project_grade = Flatten()(embedding_layer_project_grade)
```

Categorical data : clean categories

In [0]:
```
X_train.columns
```

Out[28]:
```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'cleaned_project_titles', 'cleaned_resource_summary', 'essay',
       'cleaned_essay', 'nrm_price', 'final_Text', 'total_text'],
      dtype='object')
```

In [0]:
```
#Get the clean_categories values
docs_clean_categories_train=list(X_train.project_subject_categories.values)
docs_clean_categories_val=list(X_val.project_subject_categories.values)
docs_clean_categories_test=list(X_test.project_subject_categories.values)

#Initializing the keras tokenizer and fitting it on train data
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_categories_train)

#Convert the school_state to sequences using the tokenizer
sequences_clean_categories_train = tokens.texts_to_sequences(docs_clean_categ
sequences_clean_categories_val = tokens.texts_to_sequences(docs_clean_categor
sequences_clean_categories_test = tokens.texts_to_sequences(docs_clean_catego
vocab_size_clean_categories = len(tokens.word_index) + 1

#Add padding
padded_clean_categories_train = pad_sequences(sequences_clean_categories_trai
padded_clean_categories_val = pad_sequences(sequences_clean_categories_val, m
padded_clean_categories_test = pad_sequences(sequences_clean_categories_test,
#Get the flattened output for clean_categories
input_layer_clean_categories = Input(shape=(3,), name = "clean_categories")
embedding_layer_clean_categories = Embedding(input_dim=vocab_size_clean_categ
flatten_clean_categories = Flatten()(embedding_layer_clean_categories)
```

Categorical data : project subcategories

In [0]:
```
#Get the clean_subcategories values
docs_clean_subcategories_train=list(X_train.project_subject_subcategories.val
docs_clean_subcategories_val=list(X_val.project_subject_subcategories.values)
docs_clean_subcategories_test=list(X_test.project_subject_subcategories.value

#Initializing the keras tokenizer and fitting it on train data
tokens = Tokenizer()
tokens.fit_on_texts(docs_clean_subcategories_train)

#Convert the school_state to sequences using the tokenizer
sequences_clean_subcategories_train = tokens.texts_to_sequences(docs_clean_su
sequences_clean_subcategories_val = tokens.texts_to_sequences(docs_clean_subc
sequences_clean_subcategories_test = tokens.texts_to_sequences(docs_clean_sub
vocab_size_clean_subcategories = len(tokens.word_index) + 1

padded_clean_subcategories_train = pad_sequences(sequences_clean_subcategorie
padded_clean_subcategories_val = pad_sequences(sequences_clean_subcategories_
padded_clean_subcategories_test = pad_sequences(sequences_clean_subcategories
#Get the flattened output for clean_subcategories
input_layer_clean_subcategories = Input(shape=(3,), name = "clean_subcategori
embedding_layer_clean_subcategories = Embedding(input_dim=vocab_size_clean_su
flatten_clean_subcategories = Flatten()(embedding_layer_clean_subcategories)
```

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

def normalize_vars(data):
    """This function is used to normalize all the input datas between 0 and 1
    normalizer = Normalizer()
    data_normalized = normalizer.fit_transform(data.reshape(1, -1))
    return data_normalized, normalizer
```

In [0]:
```
previous_projects_train = X_train.teacher_number_of_previously_posted_project
previous_projects_val = X_val.teacher_number_of_previously_posted_projects.va
previous_projects_test = X_test.teacher_number_of_previously_posted_projects.

norm_previous_projects_train, normalizer = normalize_vars(previous_projects_t
norm_previous_projects_val = normalizer.transform(previous_projects_val.resha
norm_previous_projects_test = normalizer.transform(previous_projects_test.res

norm_previous_projects_train = norm_previous_projects_train.reshape(len(X_tra
norm_previous_projects_val = norm_previous_projects_val.reshape(len(X_val),1)
norm_previous_projects_test = norm_previous_projects_test.reshape(len(X_test)
#Input layer for teacher_number_of_previously_posted_projects
input_layer_previous_projects = Input(shape=(1,), name = "previous_projects")
#price
#Building train, test and validation data
price_train = X_train.price.values
price_val = X_val.price.values
price_test = X_test.price.values

norm_price_train, normalizer = normalize_vars(price_train.reshape(1,-1))
norm_price_val = normalizer.transform(price_val.reshape(1,-1))
norm_price_test = normalizer.transform(price_test.reshape(1,-1))

norm_price_train = norm_price_train.reshape(len(X_train),1)
norm_price_val = norm_price_val.reshape(len(X_val),1)
norm_price_test = norm_price_test.reshape(len(X_test),1)
#Input layer for price
input_layer_price = Input(shape=(1,), name = "price")
#quantity
#Building train and validation data
quantity_train = X_train.quantity.values
quantity_val = X_val.quantity.values
quantity_test = X_test.quantity.values

norm_quantity_train, normalizer = normalize_vars(quantity_train.reshape(1,-1)
norm_quantity_val = normalizer.transform(quantity_val.reshape(1,-1))
norm_quantity_test = normalizer.transform(quantity_test.reshape(1,-1))

norm_quantity_train = norm_quantity_train.reshape(len(X_train),1)
norm_quantity_val = norm_quantity_val.reshape(len(X_val),1)
norm_quantity_test = norm_quantity_test.reshape(len(X_test),1)
#Input layer for quantity
input_layer_quantity = Input(shape=(1,), name = "quantity")
```

In [0]:
```
# concatenating all
numerical_features_layers_concat = concatenate([input_layer_previous_projects
dense_layer_numerical = Dense(6, activation='relu',kernel_initializer='he_nor
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:4479: The name tf.truncated_normal is deprecated. Please us
e tf.random.truncated_normal instead.
```

In [0]:
```
# Merge all the layers according to the architecture diagram
x = concatenate([flatten_total_text, flatten_teacher_prefix, flatten_school_s
x = Dense(10, activation='relu',kernel_initializer='he_normal',name='dense_la
x = Dropout(0.3, name='dropout_1')(x)
x = Dense(10, activation='relu',kernel_initializer='he_normal',name='dense_la
x = Dropout(0.3, name='dropout_2')(x)
output_layer = Dense(1, activation='sigmoid', name='output_layer')(x)

# Final model
model = Model(inputs=[input_layer_total_text,input_layer_teacher_prefix,input
                      input_layer_clean_subcategories,input_layer_previous_pr
model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:148: The name tf.placeholder_with_default is deprecated. Pl
ease use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:3733: calling dropout (from tensorflow.python.ops.nn_ops) w
ith keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_prob`.
Model: "model_1"
__________________________________________________________________________________
_________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
==================
total_text_sequence (InputLayer (None, 300)          0
__________________________________________________________________________________
_________________
embedding_1 (Embedding)         (None, 300, 300)     15303600    total_text_seq
uence[0][0]
__________________________________________________________________________________
_________________
teacher_prefix (InputLayer)     (None, 1)            0
__________________________________________________________________________________
_________________
encoded_school_state (InputLaye (None, 1)            0
__________________________________________________________________________________
_________________
project_grade_category (InputLa (None, 3)            0
__________________________________________________________________________________
_________________
clean_categories (InputLayer)   (None, 3)            0
__________________________________________________________________________________
_________________
clean_subcategories (InputLayer (None, 3)            0
__________________________________________________________________________________
_________________
previous_projects (InputLayer)  (None, 1)            0
__________________________________________________________________________________
_________________
price (InputLayer)              (None, 1)            0
```

```
__________________________________________________________________________________________________
quantity (InputLayer)           (None, 1)            0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 300, 16)      20288       embedding_1[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 1, 4)         24          teacher_prefix[0][0]
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 1, 4)         208         encoded_school_state[0][0]
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, 3, 4)         40          project_grade_category[0][0]
__________________________________________________________________________________________________
embedding_5 (Embedding)         (None, 3, 4)         64          clean_categories[0][0]
__________________________________________________________________________________________________
embedding_6 (Embedding)         (None, 3, 4)         152         clean_subcategories[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 3)            0           previous_projects[0][0]
                                                                 price[0][0]
                                                                 quantity[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 4800)         0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 4)            0           embedding_2[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 4)            0           embedding_3[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 12)           0           embedding_4[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 12)           0           embedding_5[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 12)           0           embedding_6[0][0]
__________________________________________________________________________________________________
```

```
_____________________
dense_1 (Dense)                 (None, 6)            24          concatenate_1
[0][0]
__________________________________________________________________________________
_____________________
concatenate_2 (Concatenate)     (None, 4850)         0           flatten_1[0]
[0]
                                                                 flatten_2[0]
[0]
                                                                 flatten_3[0]
[0]
                                                                 flatten_4[0]
[0]
                                                                 flatten_5[0]
[0]
                                                                 flatten_6[0]
[0]
                                                                 dense_1[0][0]
__________________________________________________________________________________
_____________________
dense_layer_1 (Dense)           (None, 10)           48510       concatenate_2
[0][0]
__________________________________________________________________________________
_____________________
dropout_1 (Dropout)             (None, 10)           0           dense_layer_1
[0][0]
__________________________________________________________________________________
_____________________
dense_layer_2 (Dense)           (None, 10)           110         dropout_1[0]
[0]
__________________________________________________________________________________
_____________________
dropout_2 (Dropout)             (None, 10)           0           dense_layer_2
[0][0]
__________________________________________________________________________________
_____________________
output_layer (Dense)            (None, 1)            11          dropout_2[0]
[0]
==================================================================================
==================
Total params: 15,373,031
Trainable params: 69,431
Non-trainable params: 15,303,600
__________________________________________________________________________________
_____________________
```

In [0]:
```
from keras import backend as K
from sklearn.metrics import roc_auc_score
#https://stackoverflow.com/questions/51922500/tf-metrics-auc-yielding-very-di
def roc_auc(y_true, y_pred):
    auc = tf.py_func(roc_auc_score, (y_true, y_pred), tf.double)
    #auc = tf.metrics.auc(y_true, y_pred, num_thresholds=200)[1]
    K.get_session().run(tf.local_variables_initializer())
    return auc
#Defining callbacks
from time import time
#from tensorflow.python.keras.callbacks import TensorBoard
from keras.callbacks import ModelCheckpoint
from keras.callbacks import TensorBoard

tensorboard = TensorBoard(log_dir="logs".format(time))
filepath="weights_best.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_roc_auc', verbose=1, save
import gc
gc.collect()
```

Out[35]: 5

In [0]:
```
import tensorflow as tf
```

In [0]:
```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[roc_auc]
model.fit(x=[padded_text_train,sequences_teacher_prefix_train,sequences_schoo
             norm_previous_projects_train,norm_price_train,norm_quantity_trai
          y=[labels_train],
          validation_data=([padded_text_val,sequences_teacher_prefix_val,sequ
                            norm_previous_projects_val,norm_price_val,norm_qu
          epochs=7,
          batch_size=1024,
          callbacks=[tensorboard])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimize
rs.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v
1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:3657: The name tf.log is deprecated. Please use tf.mat
h.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_cor
e/python/ops/nn_impl.py:183: where (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From <ipython-input-35-b70f758d7380>:5: py_func (from tens
orflow.python.ops.script_ops) is deprecated and will be removed in a future v
ersion.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor t
o
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use
tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.c
ompat.v1.assign instead.

Train on 69918 samples, validate on 17480 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callback
s.py:1122: The name tf.summary.merge_all is deprecated. Please use tf.compat.
v1.summary.merge_all instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callback
s.py:1125: The name tf.summary.FileWriter is deprecated. Please use tf.compa
t.v1.summary.FileWriter instead.
```

```
Epoch 1/7
69918/69918 [==============================] - 39s 557us/step - loss: 0.4618
- roc_auc: 0.5689 - val_loss: 0.4135 - val_roc_auc: 0.7006
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callback
s.py:1265: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary
instead.

Epoch 2/7
69918/69918 [==============================] - 36s 516us/step - loss: 0.4259
- roc_auc: 0.6409 - val_loss: 0.3959 - val_roc_auc: 0.7145
Epoch 3/7
69918/69918 [==============================] - 36s 518us/step - loss: 0.4117
- roc_auc: 0.6753 - val_loss: 0.3884 - val_roc_auc: 0.7197
Epoch 4/7
69918/69918 [==============================] - 37s 524us/step - loss: 0.4012
- roc_auc: 0.6970 - val_loss: 0.3836 - val_roc_auc: 0.7267
Epoch 5/7
69918/69918 [==============================] - 36s 513us/step - loss: 0.3962
- roc_auc: 0.7081 - val_loss: 0.3843 - val_roc_auc: 0.7276
Epoch 6/7
69918/69918 [==============================] - 36s 516us/step - loss: 0.3916
- roc_auc: 0.7194 - val_loss: 0.3862 - val_roc_auc: 0.7255
Epoch 7/7
69918/69918 [==============================] - 36s 508us/step - loss: 0.3871
- roc_auc: 0.7287 - val_loss: 0.3866 - val_roc_auc: 0.7262
```

Out[37]: <keras.callbacks.History at 0x7f6f48f5f1d0>

In [0]:
```
!tensorboard --logdir=logs/ --host=127.0.0.1
```

```
TensorBoard 1.15.0 at http://127.0.0.1:6006/ (http://127.0.0.1:6006/) (Press CT
RL+C to quit)
```

Auc on test data

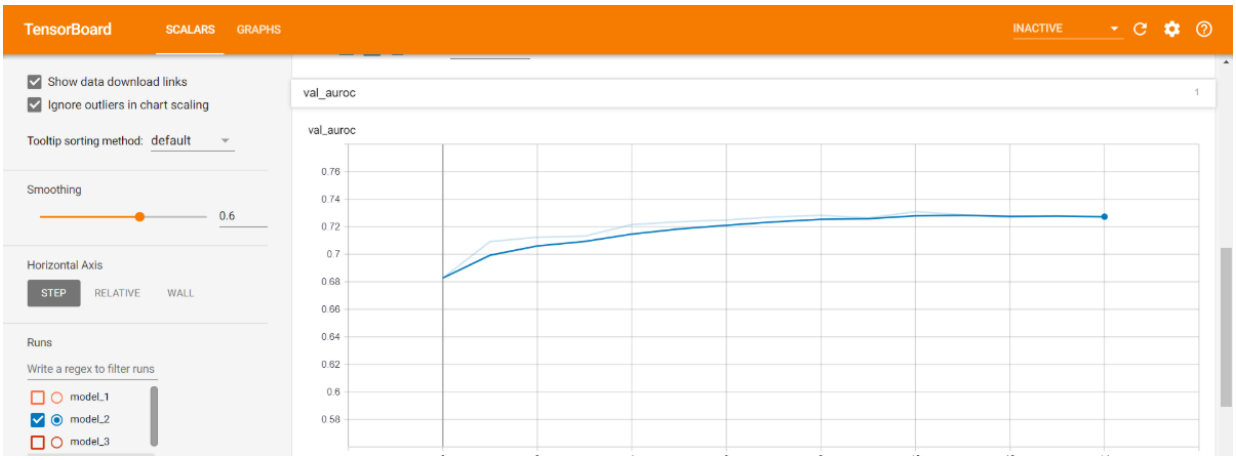
In [0]:
```
test_data=[padded_text_test,sequences_teacher_prefix_test,sequences_school_te
                  padded_clean_categories_test,padded_clean_subcategories_tes
```

In [0]:
```
#Test AUC
y_pred= model.predict(test_data)
print("AUC on unseen test data: ",roc_auc_score(Y_test,y_pred))

```

```
AUC on unseen test data:  0.7275554325048124
```

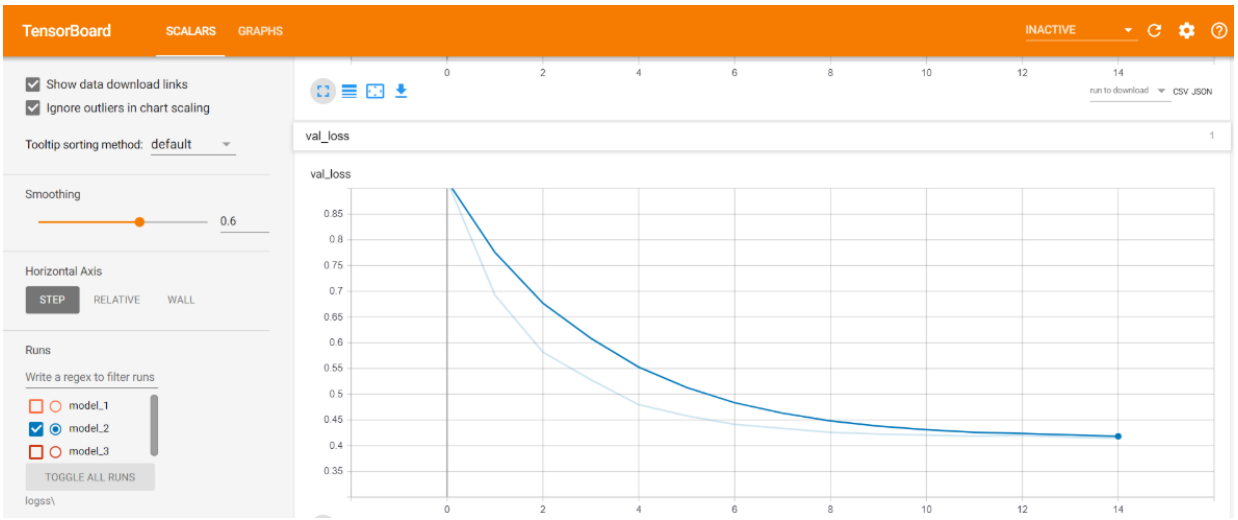
In [3]:
```
from IPython.display import Image
Image("Capture3.PNG")
```

Out[3]:

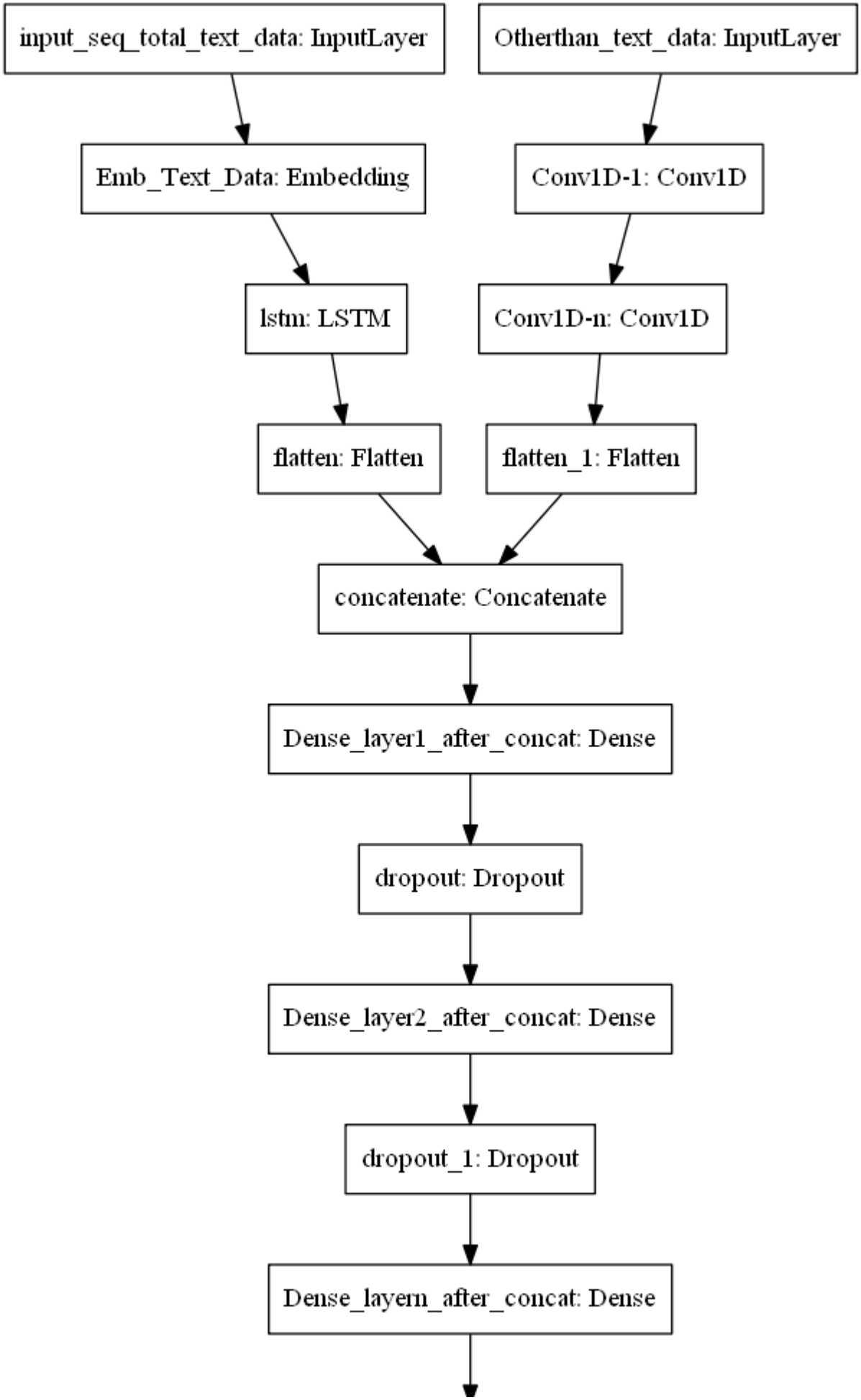


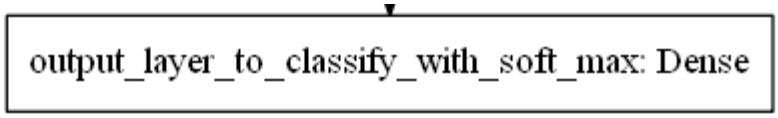
In [4]:
```
from IPython.display import Image
Image("Capture4.PNG")
```

Out[4]:



Model-3


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



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

- **input_seq_total_text_data**:

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

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

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

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

- **Other_than_text_data**:

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

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

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

In [0]:
```
1 X_train.columns
```

Out[11]:
```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'cleaned_project_titles', 'cleaned_resource_summary', 'essay',
       'cleaned_essay', 'nrm_price', 'final_Text'],
      dtype='object')
```

In [0]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(fil_words)
seq_train = tokenizer.texts_to_sequences(X_train["final_Text"])
seq_cv = tokenizer.texts_to_sequences(X_cv['final_Text'])
seq_test = tokenizer.texts_to_sequences(X_test["final_Text"])

vocab_size_text = len(tokenizer.word_index)+1 #vocablury size of the data

#Add padding
padded_text_train = pad_sequences(seq_train, maxlen=300, padding='post')
padded_text_val = pad_sequences(seq_cv, maxlen=300, padding='post')
padded_text_test = pad_sequences(seq_test, maxlen=300, padding='post')
```

In [15]:
```
pickle_in = open("drive/My Drive/lstm_donor/glove_vectors","rb")
glove_words = pickle.load(pickle_in)
embedding_matrix = zeros((vocab_size_text, 300))
for word, i in tokenizer.word_index.items(): #enumerating all unique words
    embedding_vector = glove_words.get(word)#locating for loaded glove model
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

print('the shappe of embedding matrix is:',embedding_matrix.shape)
```

```
the shappe of embedding matrix is: (24730, 300)
```

In [0]:
```
# input_text = Input(shape=(800,),name="input_text")
import warnings
warnings.filterwarnings('ignore')
from keras import regularizers
input_text = Input(shape=(300,),name="input_text")
embedding_layer = Embedding(vocab_size_text,output_dim = 300,weights=[embeddi
x = embedding_layer(input_text)
x = LSTM(256,recurrent_dropout=0.5,kernel_regularizer=regularizers.l2(0.001),
flat_1 = Flatten()(x)
```

In [0]:
```
X_train.columns
```

Out[19]:
```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category', 'project_subject_categories',
       'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'cleaned_project_titles', 'cleaned_resource_summary', 'essay',
       'cleaned_essay', 'nrm_price', 'final_Text'],
      dtype='object')
```

In [0]:

```
vect = CountVectorizer(binary=True)
vect.fit(X_train["teacher_prefix"])
train_prefix = vect.transform(X_train["teacher_prefix"])
val_prefix = vect.transform(X_cv['teacher_prefix'])
test_prefix = vect.transform(X_test["teacher_prefix"])

vect = CountVectorizer(binary=True)
vect.fit(X_train["school_state"])
train_state = vect.transform(X_train["school_state"])
val_state = vect.transform(X_cv["school_state"])
test_state = vect.transform(X_test["school_state"])

vect = CountVectorizer(binary=True)
vect.fit(X_train["school_state"])
train_grade = vect.transform(X_train["project_grade_category"])
val_grade = vect.transform(X_cv["project_grade_category"])
test_grade = vect.transform(X_test["project_grade_category"])

vect = CountVectorizer(binary=True)
vect.fit(X_train["project_subject_categories"])
train_subcat = vect.transform(X_train["project_subject_categories"])
val_subcat = vect.transform(X_cv["project_subject_categories"])
test_subcat = vect.transform(X_test["project_subject_categories"])

vect = CountVectorizer(binary=True)
vect.fit(X_train["project_subject_subcategories"])
train_subcat_1 = vect.transform(X_train["project_subject_subcategories"])
val_subcat_1 = vect.transform(X_cv["project_subject_subcategories"])
test_subcat_1 = vect.transform(X_test["project_subject_subcategories"])
```

In [0]:

```
#for all the numerical features

#numerical_train_a=X_train['digits_in_summary'].values.reshape(-1, 1)
num_train_1=X_train['price'].values.reshape(-1, 1)
num_train_2=X_train['quantity'].values.reshape(-1, 1)
num_train_3=X_train['teacher_number_of_previously_posted_projects'].values.re


num_val_1=X_cv['price'].values.reshape(-1, 1)
num_val_2=X_cv['quantity'].values.reshape(-1, 1)
num_val_3=X_cv['teacher_number_of_previously_posted_projects'].values.reshape

#numerical_test_a=X_test['digits_in_summary'].values.reshape(-1, 1)
num_test_1=X_test['price'].values.reshape(-1, 1)
num_test_2=X_test['quantity'].values.reshape(-1, 1)
num_test_3=X_test['teacher_number_of_previously_posted_projects'].values.resh
```

In [ ]:
```
numerical_train=np.concatenate((num_train_1,num_train_2,num_train_3),axis=1)
numerical_val=np.concatenate((num_val_1,num_val_2,num_val_3),axis=1)
numerical_test=np.concatenate((num_test_1,num_test_2,num_test_3),axis=1)

from sklearn.preprocessing import StandardScaler
normal=StandardScaler()
normal_train=normal.fit_transform(numerical_train)
normal_val=normal.fit_transform(numerical_val)
normal_test=normal.transform(numerical_test)
```

In [0]:
```
from scipy.sparse import hstack
other_train = hstack([train_prefix,train_state,train_grade,train_subcat,train
other_val = hstack([val_prefix,val_state,val_grade,val_subcat,val_subcat_1]).
other_test = hstack([test_prefix,test_state,test_grade,test_subcat,test_subca
other_train = np.hstack((ot_train,normal_train))
other_val = np.hstack((ot_val,normal_val))
other_test = np.hstack((ot_test,normal_test))
other_train_all = np.expand_dims(ot_all_train,2)
other_val_all = np.expand_dims(ot_all_val,2)
other_test_all = np.expand_dims(ot_all_test,2)
```

In [22]:
```
from keras.layers import Input, Embedding, LSTM, Dropout, BatchNormalization,
inp_conv=Input(shape=(other_all_train.shape[1], 1))
x1 = Conv1D(filters=128, kernel_size=3, activation='relu',kernel_initializer=
x1 = Conv1D(filters=128, kernel_size=3, activation='relu',kernel_initializer=
x1 = Flatten()(x1)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:4479: The name tf.truncated_normal is deprecated. Please us
e tf.random.truncated_normal instead.
```

In [0]:
```

x_concatenate = concatenate([flat_1,x1])
x = Dense(128,activation="relu",kernel_initializer="he_normal",kernel_regular
x=Dropout(0.5)(x)
x = Dense(64,activation="relu",kernel_initializer="he_normal",kernel_regulari
x=Dropout(0.3)(x)
x = Dense(32,activation="relu",kernel_initializer="he_normal",kernel_regulari
output = Dense(2, activation='softmax', name='output')(x)
model_1 = Model(inputs=[input_text,inp_conv],outputs=[output])
```

In [0]:
```
train_data_3 = [padded_text_train,ot_train_all]
val_data_3 = [padded_text_val,ot_val_all]
test_data_3 = [padded_text_test,ot_test_all]
from keras.utils import np_utils
Y_train = np_utils.to_categorical(Y_train, 2)
Y_val = np_utils.to_categorical(Y_cv, 2)
Y_test = np_utils.to_categorical(Y_test, 2)
```

In [0]:
```
from sklearn.metrics import roc_auc_score
def auc1(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score(y_true, y_pred)
def auroc(y_true, y_pred):
    return tf.py_func(auc1, (y_true, y_pred), tf.double)
```

In [0]:
```
from keras.optimizers import Adam
import tensorflow as tf
```

In [27]:
```
model_1.compile(optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=No
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizer
s.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.tr
ain.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
nsorflow_backend.py:3576: The name tf.log is deprecated. Please use tf.math.log
instead.

WARNING:tensorflow:From <ipython-input-25-b3d027b8e5dd>:8: py_func (from tensor
flow.python.ops.script_ops) is deprecated and will be removed in a future versi
on.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.
```

In [0]:
```

checkpoint_3 = ModelCheckpoint("model_3.h5",monitor="val_auroc",mode="max",sa
NAME = 'model_3'
tensorboard_2 = TensorBoard(log_dir='logs')
callbacks_2 = [tensorboard_2,checkpoint_3]
```

In [37]:
```
history_1 = model_1.fit(train_data_3,Y_train,batch_size=512,epochs=15,validat
```

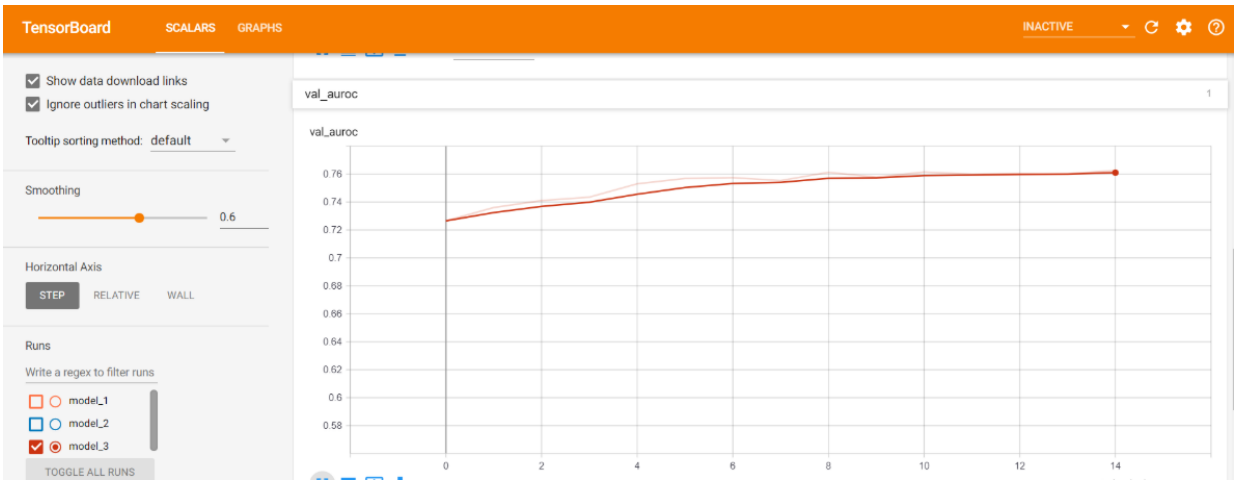
```
Train on 69918 samples, validate on 17480 samples
Epoch 1/15
69918/69918 [==============================] - 76s 1ms/step - loss: 0.8016 - au
roc: 0.6520 - val_loss: 0.6379 - val_auroc: 0.7352
Epoch 2/15
69918/69918 [==============================] - 75s 1ms/step - loss: 0.5536 - au
roc: 0.7216 - val_loss: 0.5570 - val_auroc: 0.7476
Epoch 3/15
69918/69918 [==============================] - 74s 1ms/step - loss: 0.4930 - au
roc: 0.7324 - val_loss: 0.5011 - val_auroc: 0.7417
Epoch 4/15
69918/69918 [==============================] - 75s 1ms/step - loss: 0.4623 - au
roc: 0.7346 - val_loss: 0.4821 - val_auroc: 0.7493
Epoch 5/15
69918/69918 [==============================] - 75s 1ms/step - loss: 0.4403 - au
roc: 0.7460 - val_loss: 0.4691 - val_auroc: 0.7615
Epoch 6/15
69918/69918 [==============================] - 75s 1ms/step - loss: 0.4241 - au
roc: 0.7544 - val_loss: 0.4552 - val_auroc: 0.7610
Epoch 7/15
69918/69918 [==============================] - 75s 1ms/step - loss: 0.4128 - au
roc: 0.7587 - val_loss: 0.4182 - val_auroc: 0.7620
Epoch 8/15
69918/69918 [==============================] - 75s 1ms/step - loss: 0.4047 - au
roc: 0.7627 - val_loss: 0.4197 - val_auroc: 0.7631
Epoch 9/15
69918/69918 [==============================] - 76s 1ms/step - loss: 0.4015 - au
roc: 0.7617 - val_loss: 0.4119 - val_auroc: 0.7602
Epoch 10/15
69918/69918 [==============================] - 76s 1ms/step - loss: 0.3976 - au
roc: 0.7625 - val_loss: 0.4145 - val_auroc: 0.7628
Epoch 11/15
69918/69918 [==============================] - 76s 1ms/step - loss: 0.3933 - au
roc: 0.7635 - val_loss: 0.4053 - val_auroc: 0.7654
Epoch 12/15
69918/69918 [==============================] - 78s 1ms/step - loss: 0.3916 - au
roc: 0.7661 - val_loss: 0.4001 - val_auroc: 0.7633
Epoch 13/15
69918/69918 [==============================] - 78s 1ms/step - loss: 0.3892 - au
roc: 0.7655 - val_loss: 0.4177 - val_auroc: 0.7619
Epoch 14/15
69918/69918 [==============================] - 77s 1ms/step - loss: 0.3866 - au
roc: 0.7691 - val_loss: 0.3923 - val_auroc: 0.7681
Epoch 15/15
69918/69918 [==============================] - 78s 1ms/step - loss: 0.3855 - au
roc: 0.7678 - val_loss: 0.3983 - val_auroc: 0.7677
```

In [40]:
```
#Test AUC
y_pred= model_1.predict(test_data_3)
print("AUC on unseen test data: ",roc_auc_score(Y_test,y_pred))
```

```
AUC on unseen test data:  0.7633947879300654
```

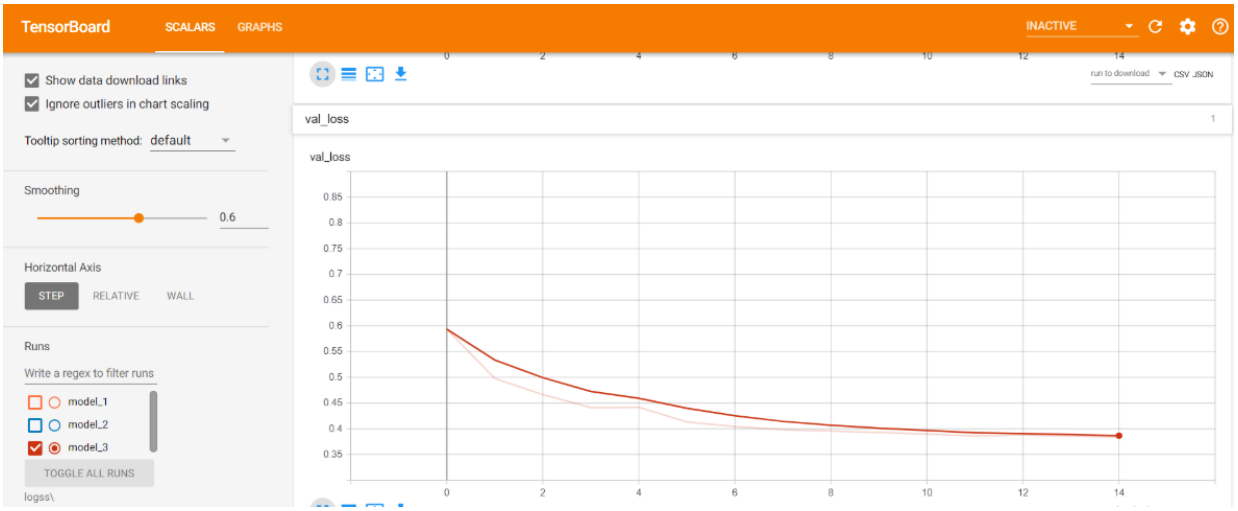
In [5]:
```
from IPython.display import Image
Image("Capture5.PNG")
```

Out[5]:



In [6]:
```
from IPython.display import Image
Image("Capture6.PNG")
```

Out[6]:



In [12]:
```
from prettytable import PrettyTable
table =PrettyTable()
table.field_names = ["Model No", "Trained for Epochs", "Train ROC-AUC", "Vali
table.add_row(["Model 1",10,0.7988,0.7169,0.7148])
table.add_row(["Model 2",7,0.7287,0.7262,0.7275])
table.add_row(["Model 3",15,0.7678,0.7677,0.7633])
print(table)
```

```
+----------+--------------------+--------------+--------------------+---------
-----+
| Model No | Trained for Epochs | Train ROC-AUC | Validation ROC-AUC | Test ROC
-AUC |
+----------+--------------------+--------------+--------------------+---------
-----+
| Model 1  |         10         |     0.7988    |       0.7169       |    0.714
8    |
| Model 2  |         7          |     0.7287    |       0.7262       |    0.727
5    |
| Model 3  |         15         |     0.7678    |       0.7677       |    0.763
3    |
+----------+--------------------+--------------+--------------------+---------
-----+
```

What was the case study all about and what we learnt

So this case study was an application of machine learning for social good where we we were able to automate the process of approving the projects being submitted for evaluation based on lots of different factors.

We tokenized the text data and one hot encoded the categorical variables and also performed normalization on the numerical features primarily ,along we that we performed the feature engineering with digits present in the text or not .

We also used pre embedded glove vectors as the main embedding layer for the text data.

Amongst all the three models, the 3rd model with 1D CNN layers seems to perform the best as we have got the maximum value of ROC-AUC for it.

Using RSM Prop with proper weight initialization was also resulting in exploding gradients for Model 1. Changing the optimizer to adam has changed this problem. Got the loss curves and score curve using Tensorboard. Used a custome metric function for training the model with a custom roc-auc score. For TFIDF analysis, I first tried with the 25 percentile threshold for tfidf scores and the model performed very poorly despiting trying my best to optimize it. The lowest threshold I have considered for this assignment is 6.5 approximately. Model 3 has given us the best value of ROC-AUC - just under 0.75.

For TFIDF analysis, I had previously tried with IDF score between 7 and max value, ut it resulted in huge loss of data. Hence I took IDF values which were greater than 3 and the model has improved a lot.