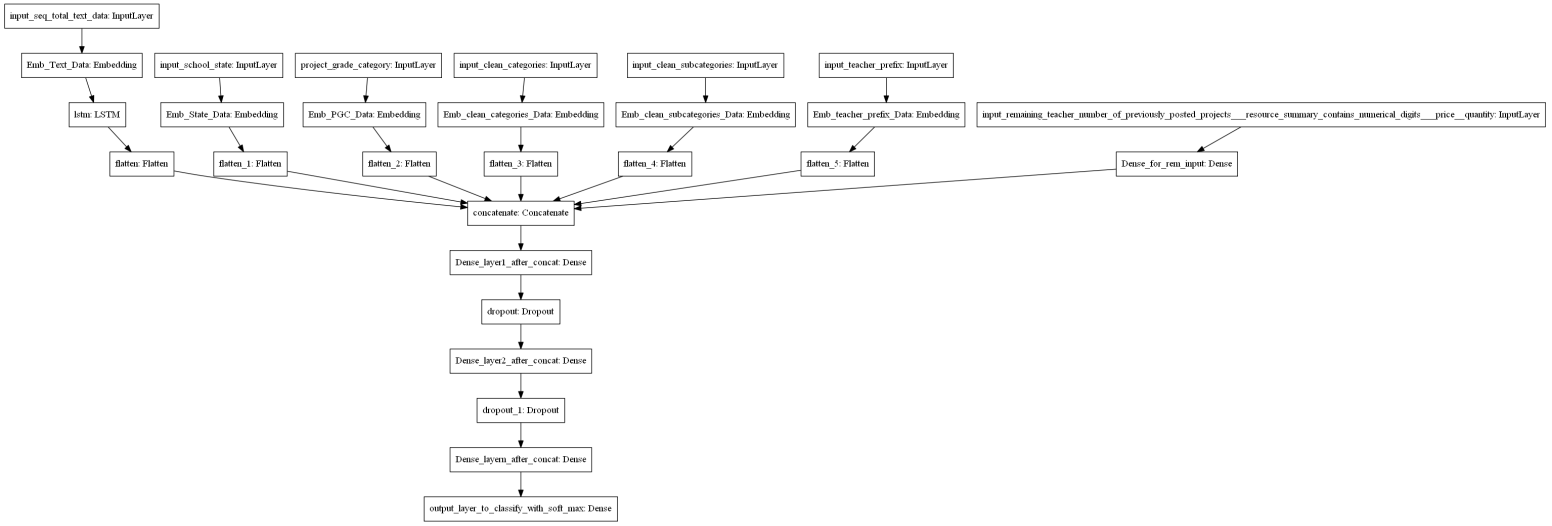
Assignment : 14

1. Preprocess all the Data we have in DonorsChoose Dataset 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' as a metric. check this for using auc as a metric
5. You are free to choose any number of layers/hiddden units but you have to use same type of architectures shown below.
6. You can use any one of the optimizers and choice of Learning rate and momentum, resources: cs231n class notes, cs231n class vide
7. For all the model's use TensorBoard and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots
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

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

- **Input_clean_categories** --- Give 'input_clean_categories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_clean_subcategories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** --- Give 'input_teacher_prefix' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_remaining_teacher_number_of_previously_posted_projects._resource_summary_contains_numerical_digits._price._quantity** --- 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.

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

```
'\n# https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work\ninput_layer = Input(shape=(n,))\nembedding = Embedding(no_1, no_2, input_length=
```

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

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

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 ch
```

```
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 aft
```

Model 1

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%

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

```
with open('/content/drive/My Drive/foo.txt', 'w') as f:
  f.write('Hello Google Drive!')
!cat /content/drive/My\ Drive/foo.txt
```

```
Hello Google Drive!
```

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
```

```
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os


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

```
project_data = pd.read_csv('/content/drive/My Drive/train_data.csv')
resource_data = pd.read_csv('/content/drive/My Drive/resources.csv')
```

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of project_subject_categories

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

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

1.3 preprocessing of project_subject_subcategories

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
```

```
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.3 Text preprocessing

Removing null values from project essay 3 & 4

```
# check if we have any nan values are there in the column
print(project_data['project_essay_3'].isnull().values.any())
print("number of nan values",project_data['project_essay_3'].isnull().values.sum())
```

```
True
number of nan values 105490
```

```
#Replacing the Nan values with most frequent value in the column
project_data['project_essay_3']=project_data['project_essay_3'].fillna(' ')
```

```
# check if we have any nan values are there in the column
print(project_data['project_essay_3'].isnull().values.any())
print("number of nan values",project_data['project_essay_3'].isnull().values.sum())
```

```
False
number of nan values 0
```

```
# check if we have any nan values are there in the column
print(project_data['project_essay_4'].isnull().values.any())
print("number of nan values",project_data['project_essay_4'].isnull().values.sum())
```

```
True
number of nan values 105490
```

```
#Replacing the Nan values with most frequent value in the column
project_data['project_essay_4']=project_data['project_essay_4'].fillna(' ')
```

```
# check if we have any nan values are there in the column
print(project_data['project_essay_4'].isnull().values.any())
print("number of nan values",project_data['project_essay_4'].isnull().values.sum())
```

```
False
number of nan values 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)
```

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

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_essay_1	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educational Support for English Learners at Home	My students are English learners that are work...	\"
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Wanted: Projector for Hungry Learners	Our students arrive to our school eager to lea...	

1.4.2.3 Using Pretrained Models: TFIDF weighted W2V

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[50])
print("="*50)
print(project_data['essay'].values[100])
print("="*50)
print(project_data['essay'].values[200])
print("="*50)
print(project_data['essay'].values[999])
print("="*50)
```

```
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans br
==================================================
The students in our rural NC school come from various backgrounds with many different learning styles and abilities. Many are from military families that have a mother or
==================================================
I teach in a dual immersion 4th grade classroom. We teach 50% of the day in English and 50% in Spanish. My classroom is the English model for two classrooms of 30 students
==================================================
As an inclusion kindergarten teacher, I am constantly looking for materials to help students develop and grow throughout the school year.  This has been challenging with t
==================================================
Welcome to our spectacular 1st and 2nd grade ELL classroom.  I have the most amazing class of motivated second language learners.  These youngsters come from homes with ha
==================================================
```

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

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

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


sent = decontracted(project_data['essay'].values[200])
print(sent)
print("="*50)
```

```
As an inclusion kindergarten teacher, I am constantly looking for materials to help students develop and grow throughout the school year.  This has been challenging with t
==================================================
```

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
As an inclusion kindergarten teacher, I am constantly looking for materials to help students develop and grow throughout the school year.  This has been challenging with t
```

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
As an inclusion kindergarten teacher I am constantly looking for materials to help students develop and grow throughout the school year This has been challenging with the
```

```
# 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're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

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

```
100%|██████████| 109248/109248 [00:55<00:00, 1960.63it/s]
```

```
# after preprocesing
preprocessed_essays[200]
```

```
'as inclusion kindergarten teacher i constantly looking materials help students develop grow throughout school year this challenging school limited funding supplies we cla
```

```
project_data['preprocessed_essays'] = preprocessed_essays
```

1.4 Preprocessing of `project_title`

```
# similarly you can preprocess the titles also
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_essay_1	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educational Support for English Learners at Home	My students are English learners that are work...	\"
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Wanted: Projector for Hungry Learners	Our students arrive to our school eager to lea...	s

```
# printing some random project titles.
print(project_data['project_title'].values[54])
print("="*50)
print(project_data['project_title'].values[89])
print("="*50)
print(project_data['project_title'].values[99])
print("="*50)
print(project_data['project_title'].values[156])
print("="*50)
print(project_data['project_title'].values[846])
print("="*50)
```

```
     Swim For Life At YMCA!
     ==================================================
```

```
#Removing phrases from the title features
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"Gotta", "Got to", phrase)
    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase


#Checkingt titles after removing phrases
sent = decontracted(project_data['project_title'].values[836])
print(sent)
print("="*50)
```

```
Digital Magazine
==================================================
```

```
# Remove \\r \\n \\t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
Digital Magazine
```

```
#Removing numbers & symbols form the titles
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Digital Magazine
```

```
# 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're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
```

```
 she's", "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
"mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
'won', "won't", 'wouldn', "wouldn't"]
```

```
#Combining all the above preprocessed statements
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:02<00:00, 40341.32it/s]
```

```
#checking cleaned text after preprocesing
print(preprocessed_titles[54])
print("="*50)
print(preprocessed_titles[89])
print("="*50)
print(preprocessed_titles[99])
print("="*50)
print(preprocessed_titles[156])
print("="*50)
print(preprocessed_titles[836])
```

```
project_data['preprocessed_titles'] = preprocessed_titles
```

```
     education through technology
```

```
project_data['all_text'] = project_data['preprocessed_essays'] + ' ' + project_data['preprocessed_titles']
```

```
     ================================================
```

```
all_text = project_data["all_text"]
all_text
```

```
0         my students english learners working english s...
1         our students arrive school eager learn they po...
2         true champions not always ones win guts by mia...
3         i work unique school filled esl english second...
4         our second grade classroom next year made arou...
                                ...
109243    welcome mr ramos 2nd grade classroom we title ...
109244    every morning start day core values lead solel...
109245    this great group sharing caring students it mu...
109246    our students live small rural community our cl...
109247    when last time used math probably within last ...
Name: all_text, Length: 109248, dtype: object
```

```
# check if we have any nan values are there in the column
print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

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

```
#Replacing the Nan values with most frequent value in the column
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
```

```
# check if we have any nan values are there in the column
print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
False
number of nan values 0
```

```
#Converting teacher prefix text into smaller case
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix'].value_counts()
```

```
mrs.       57272
ms.        38955
mr.        10648
teacher     2360
dr.           13
Name: teacher_prefix, dtype: int64
```

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

```
X = project_data

y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
project_data.head(1)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_essay_1	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educational Support for English Learners at Home	My students are English learners that are work...	\"Tl la

```
#Splitting data into test & train set
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.33,stratify=y)

from numpy import array
from numpy import asarray
from numpy import zeros
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras import regularizers
from keras.layers import LSTM
from keras.layers import Embedding
from keras.layers import Input
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from keras.preprocessing.text import Tokenizer
from numpy import zeros
```

```
Using TensorFlow backend.
```

```
token = Tokenizer()
token.fit_on_texts(X_train['essay'].tolist())
vocab_size = len(token.word_index) + 1
# integer encode the documents
encoded_train = token.texts_to_sequences(X_train['essay'])
encoded_test = token.texts_to_sequences(X_test['essay'])


# pad documents to a max length of 1000 words
max_length = 1000
padded_train = pad_sequences(encoded_train, maxlen=max_length, padding='post')
padded_test = pad_sequences(encoded_test, maxlen=max_length, padding='post')
padded_essay_train = padded_train
padded_essay_test = padded_test


f = open("/content/drive/My Drive/glove_vectors","rb")
glove = pickle.load(f)


# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size, 300))
for word, i in token.word_index.items():
    embedding_vector = glove.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector


max_len = 1000
emb_layer = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=max_length, trainable=False)
input_lyr = Input(shape=(max_len,))
emb = emb_layer(input_lyr)
x = LSTM(128,return_sequences=True)(emb)
flat_1 = Flatten()(x)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please use tf.compat.v1.plac
```

Embedding Categorical Data

```
# code source - https://stackoverflow.com/questions/21057621/sklearn-labelencoder-with-never-seen-before-values
from sklearn.preprocessing import LabelEncoder
import numpy as np


class LabelEncoderExt(object):
    def __init__(self):
        """
        It differs from LabelEncoder by handling new classes and providing a value for it [Unknown]
        Unknown will be added in fit and transform will take care of new item. It gives unknown class id
        """
        self.label_encoder = LabelEncoder()
        # self.classes_ = self.label_encoder.classes_

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

        return self

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

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

For School State

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

```
enc_school_state_train = vectorizer.transform(X_train['school_state'].values)
enc_school_state_test = vectorizer.transform(X_test['school_state'].values)


unique_states = X_train['school_state'].nunique()
print(unique_states)
```

```
51
```

```
input_state = Input(shape=(1,),name="school_state")
state_emb_size = int(min(np.ceil((unique_states)/2), 50))
embedded_state = Embedding(unique_states, state_emb_size, trainable=True)(input_state)
flatten_state = Flatten()(embedded_state)
```

Embedding Teacher Prefix

```
vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['teacher_prefix'].values)
enc_teacher_prefix_train = vectorizer.transform(X_train['teacher_prefix'].values)
enc_teacher_prefix_test = vectorizer.transform(X_test['teacher_prefix'].values)


unique_tp = X_train['teacher_prefix'].nunique()
print(unique_tp)
```

```
5
```

```
input_tp = Input(shape=(1,),name="teacher_prefix")
tp_emb_size = int(min(np.ceil((unique_tp)/2), 50))
embedded_tp = Embedding(unique_tp, tp_emb_size, trainable=True)(input_tp)
flatten_tp = Flatten()(embedded_tp)
```

Embedding Subject Category

```
vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['clean_categories'])
enc_cat_train = vectorizer.transform(X_train['clean_categories'])
enc_cat_test = vectorizer.transform(X_test['clean_categories'])


unique_cat = X_train['clean_categories'].nunique()
print(unique_cat)
```

```
51
```

```
input_cc = Input(shape=(1,),name="clean_categories")
cat_emb_size = int(min(np.ceil((unique_cat)/2), 50))
```

```
embedded_cat = Embedding(unique_cat, cat_emb_size, trainable=True)(input_cc)
flatten_cat = Flatten()(embedded_cat)
```

Embedding Subject Sub-Category

```
vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['clean_subcategories'])
enc_subcat_train = vectorizer.transform(X_train['clean_subcategories'])
enc_subcat_test = vectorizer.transform(X_test['clean_subcategories'])
```

```
unique_subcat = X_train['clean_subcategories'].nunique()
print(unique_subcat)
```

```
396
```

```
input_subcat = Input(shape=(1,),name="clean_subcategories")
subcat_emb_size = int(min(np.ceil((unique_subcat)/2), 50))
embedded_subcat = Embedding(unique_subcat, subcat_emb_size, trainable=True)(input_subcat)
flatten_subcat = Flatten()(embedded_subcat)
```

Embedding Project Grade Category

```
vectorizer = LabelEncoderExt()
vectorizer.fit(X_train['project_grade_category'])
enc_grade_train = vectorizer.transform(X_train['project_grade_category'])
enc_grade_test = vectorizer.transform(X_test['project_grade_category'])
```

```
unique_grade = X_train['project_grade_category'].nunique()
print(unique_grade)
```

```
4
```

```
input_grade = Input(shape=(1,),name="project_grade_category")
grade_emb_size = int(min(np.ceil((unique_grade)/2), 50))
embedded_grade = Embedding(grade_emb_size, unique_grade, trainable=True)(input_grade)
flatten_grade = Flatten()(embedded_grade)
```

Vectorizing Numerical Features

For Price Feature

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')


# join two dataframes in python:
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')


from sklearn.preprocessing import Normalizer
price_normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
price_normalizer.fit(X_train['price'].values.reshape(1,-1))
X_train_price_norm = price_normalizer.transform(X_train['price'].values.reshape(1,-1))
X_test_price_norm = price_normalizer.transform(X_test['price'].values.reshape(1,-1))
print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
```

```
After vectorizations
(1, 73196) (73196,)
(1, 36052) (36052,)
```

```
X_train_price_norm = X_train_price_norm.T
X_test_price_norm = X_test_price_norm.T
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
(73196, 1) (73196,)
(36052, 1) (36052,)
====================================================================================================
```

For Quantity Feature

```
#Normalizing quantity
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
```

```
# array.reshape(1, -1) if it contains a single sample.
normalizer.fit(X_train['quantity'].values.reshape(1,-1))
X_train_quantity_norm = normalizer.transform(X_train['quantity'].values.reshape(1,-1))
X_test_quantity_norm = normalizer.transform(X_test['quantity'].values.reshape(1,-1))
print("After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 73196) (73196,)
(1, 36052) (36052,)
====================================================================================================
```

```
X_train_quantity_norm = X_train_quantity_norm.T
X_test_quantity_norm = X_test_quantity_norm.T
print("Final Matrix")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(73196, 1) (73196,)
(36052, 1) (36052,)
====================================================================================================
```

For Teacher Previously Posted Project Feature

```
# Normalizing teacher previously posted projects
#Normalizing quantity
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_train_tpp_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_test_tpp_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
print("After vectorizations")
print(X_train_tpp_norm.shape, y_train.shape)
print(X_test_tpp_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
```

```
X_train_tpp_norm = X_train_tpp_norm.T
X_test_tpp_norm = X_test_tpp_norm.T
print(X_train_tpp_norm.shape, y_train.shape)
print(X_test_tpp_norm.shape, y_test.shape)
print("="*100)
```

```
(73196, 1) (73196,)
(36052, 1) (36052,)
====================================================================================================
```

```
numerical_fts_train = np.hstack((X_train_price_norm, X_train_quantity_norm, X_train_tpp_norm))
numerical_fts_test = np.hstack((X_test_price_norm, X_test_quantity_norm, X_test_tpp_norm))
```

```
from keras.regularizers import l2
numerical_input = Input(shape=(3,),name="numerical_fts")
num_input = Dense(100, activation='relu', kernel_initializer="he_normal", kernel_regularizer="l2")(numerical_input)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4479: The name tf.truncated_normal is deprecated. Please use tf.random.t
```

```
from keras.layers import concatenate
concatenated_fts = concatenate([flat_1, flatten_state, flatten_tp, flatten_cat, flatten_subcat, flatten_grade, num_input])
```

```
from keras.models import Sequential
from keras.models import Model, load_model
from keras import regularizers
from keras.initializers import he_normal
from keras.regularizers import l2
from keras.layers import LeakyReLU
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Activation
from keras.layers import Dropout

x_concat = concatenated_fts

z = Dense(256, activation="relu", kernel_initializer="he_normal", kernel_regularizer=regularizers.l2(0.001))(x_concat)
z = (Dropout(0.3))(z)

z = Dense(128, activation="relu", kernel_initializer="he_normal", kernel_regularizer=regularizers.l2(0.001))(z)
z = (Dropout(0.3))(z)

z = Dense(64, activation="relu", kernel_initializer="he_normal", kernel_regularizer=regularizers.l2(0.001))(z)
z = (Dropout(0.3))(z)
z = BatchNormalization()(z)

z = Dense(32, activation="relu", kernel_initializer="he_normal", kernel_regularizer=regularizers.l2(0.001))(z)
```

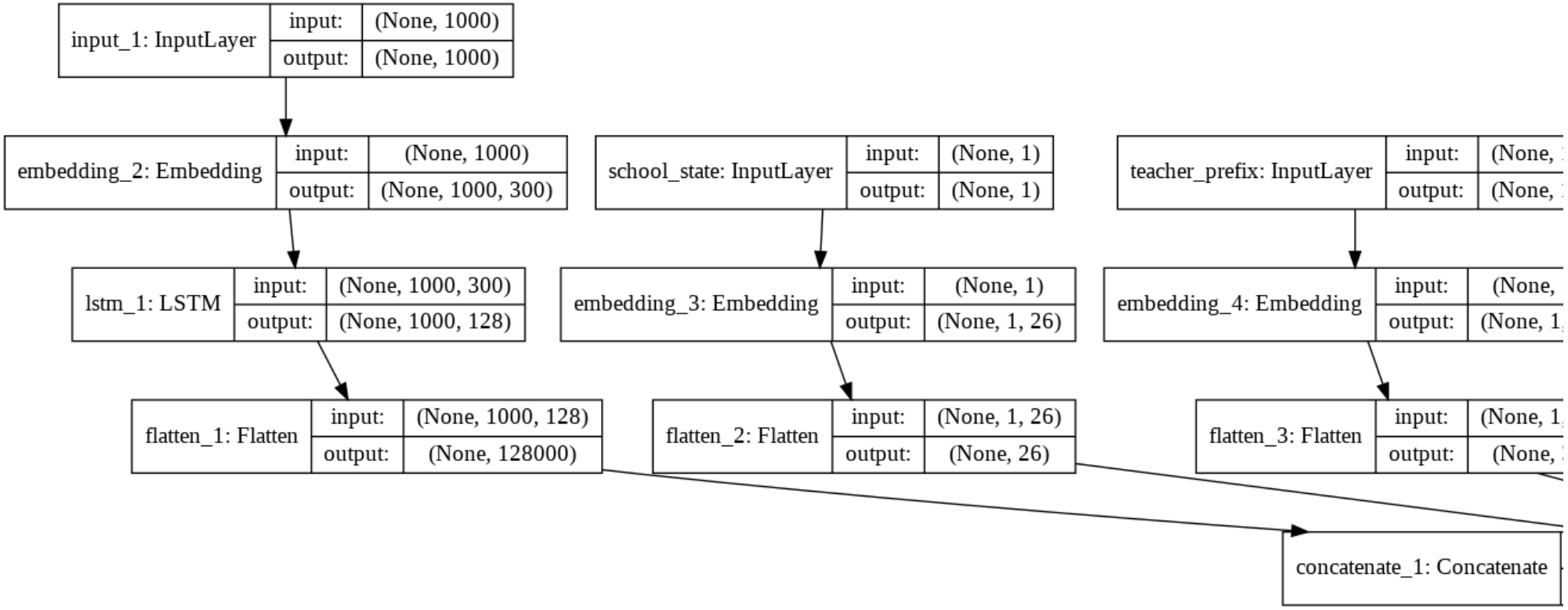
```
    z = (Dropout(0.3))(z)
    z = BatchNormalization()(z)

    output = Dense(2, activation = "softmax", name="output")(z)
    model_two = Model(inputs=[input_lyr, input_state, input_tp, input_cc, input_subcat, input_grade, numerical_input],outputs=[output])
    model_two.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148: The name tf.placeholder_with_default is deprecated. Please use tf.c

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
Model: "model_1"
```

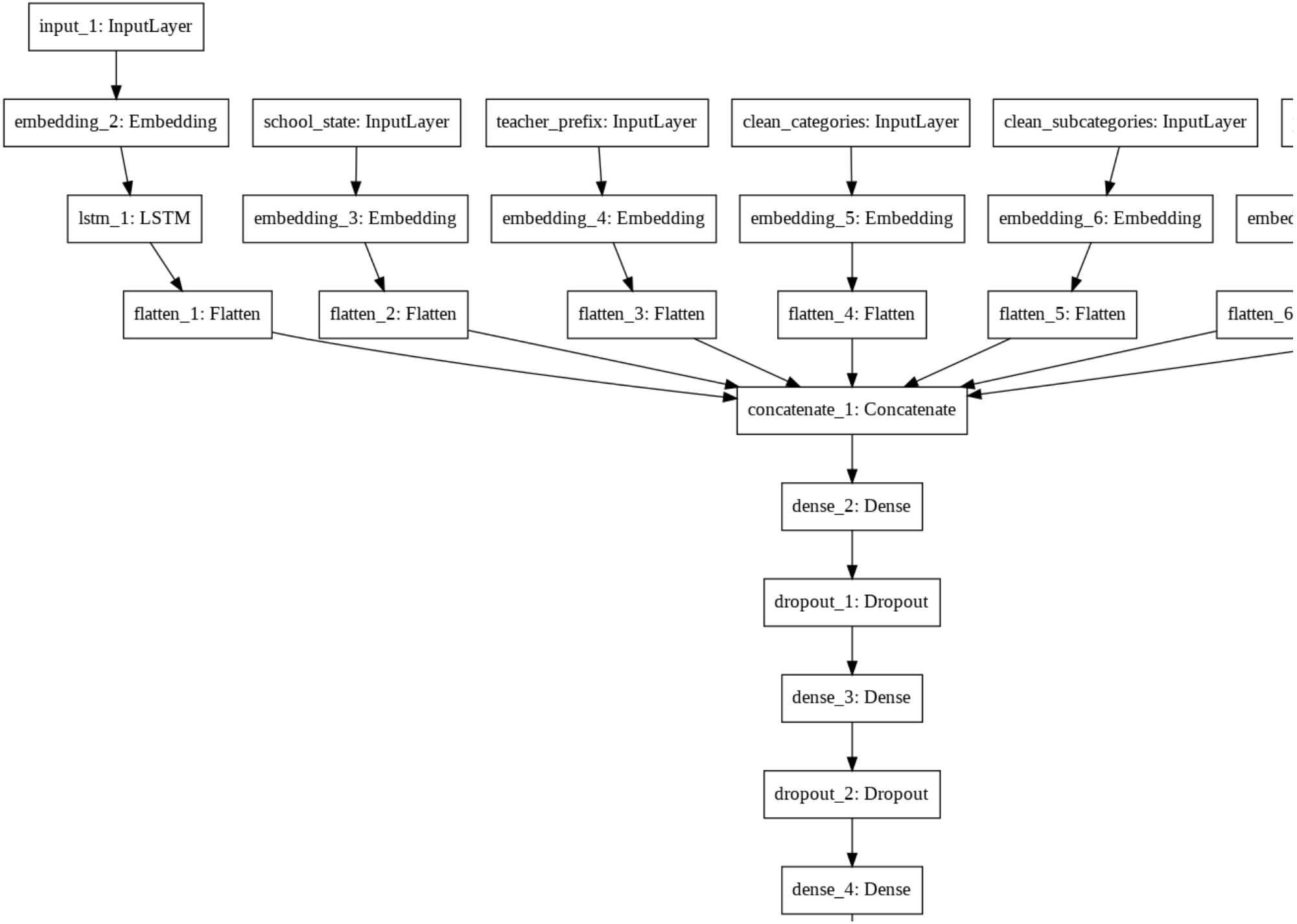
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 1000)	0	
embedding_2 (Embedding)	(None, 1000, 300)	17105100	input_1[0][0]
school_state (InputLayer)	(None, 1)	0	
teacher_prefix (InputLayer)	(None, 1)	0	
clean_categories (InputLayer)	(None, 1)	0	
clean_subcategories (InputLayer	(None, 1)	0	
project_grade_category (InputLa	(None, 1)	0	
lstm_1 (LSTM)	(None, 1000, 128)	219648	embedding_2[0][0]
embedding_3 (Embedding)	(None, 1, 26)	1326	school_state[0][0]
embedding_4 (Embedding)	(None, 1, 3)	15	teacher_prefix[0][0]
embedding_5 (Embedding)	(None, 1, 26)	1326	clean_categories[0][0]
embedding_6 (Embedding)	(None, 1, 50)	19800	clean_subcategories[0][0]
embedding_7 (Embedding)	(None, 1, 4)	8	project_grade_category[0][0]
numerical_fts (InputLayer)	(None, 3)	0	
flatten_1 (Flatten)	(None, 128000)	0	lstm_1[0][0]
flatten_2 (Flatten)	(None, 26)	0	embedding_3[0][0]
flatten_3 (Flatten)	(None, 3)	0	embedding_4[0][0]
flatten_4 (Flatten)	(None, 26)	0	embedding_5[0][0]
flatten_5 (Flatten)	(None, 50)	0	embedding_6[0][0]
flatten_6 (Flatten)	(None, 4)	0	embedding_7[0][0]
dense_1 (Dense)	(None, 100)	400	numerical_fts[0][0]
concatenate_1 (Concatenate)	(None, 128209)	0	flatten_1[0][0] flatten_2[0][0] flatten_3[0][0] flatten_4[0][0]

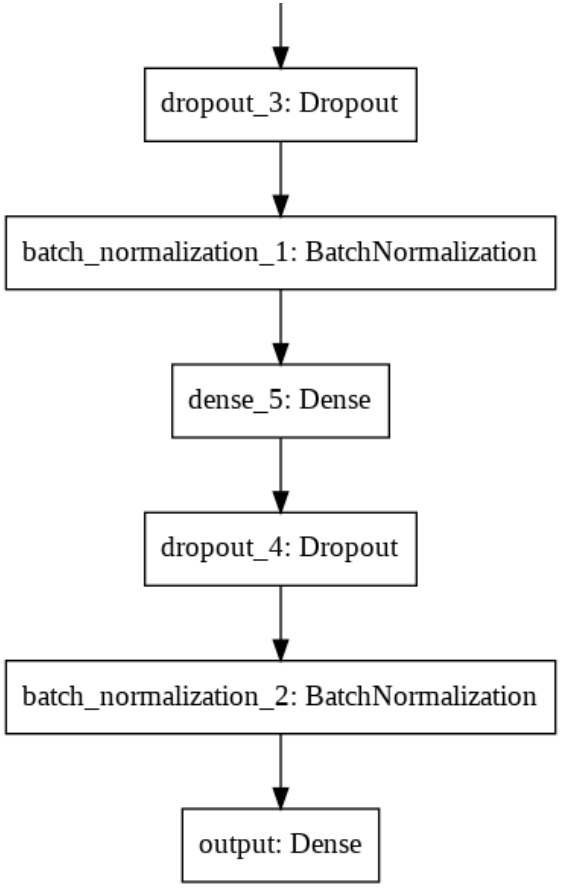
```
                                                                 flatten_5[0][0]
                                                                 flatten_6[0][0]
                                                                 dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          32821760    concatenate_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 256)          0           dense_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 128)          32896       dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 64)           8256        dropout_2[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 64)           0           dense_4[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 64)           256         dropout_3[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 32)           2080        batch_normalization_1[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 32)           0           dense_5[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 32)           128         dropout_4[0][0]
__________________________________________________________________________________________________
output (Dense)                  (None, 2)            66          batch_normalization_2[0][0]
==================================================================================================
Total params: 50,213,065
Trainable params: 33,107,773
Non-trainable params: 17,105,292
__________________________________________________________________________________________________
```

```
# code source - https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/
from keras.utils.vis_utils import plot_model
plot_model(model_two, to_file='model_one.png', show_shapes=True, show_layer_names=True)
```


input_1: InputLayer
input: (None, 1000)
output: (None, 1000)
embedding_2: Embedding
input: (None, 1000)
output: (None, 1000, 300)
school_state: InputLayer
input: (None, 1)
output: (None, 1)
teacher_prefix: InputLayer
input: (None,
output: (None,
lstm_1: LSTM
input: (None, 1000, 300)
output: (None, 1000, 128)
embedding_3: Embedding
input: (None, 1)
output: (None, 1, 26)
embedding_4: Embedding
input: (None,
output: (None, 1
flatten_1: Flatten
input: (None, 1000, 128)
output: (None, 128000)
flatten_2: Flatten
input: (None, 1, 26)
output: (None, 26)
flatten_3: Flatten
input: (None, 1
output: (None,
concatenate_1: Concatenate

```
#https://www.tensorflow.org/tensorboard/scalars_and_keras
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
checkpoint_1 = ModelCheckpoint("model_one.h5",
                               monitor="val_auroc",
                               mode="max",
                               save_best_only = True,
                               verbose=1)
earlystop_1 = EarlyStopping(monitor = 'val_auroc',
                            mode="max",
                            min_delta = 0,
                            patience = 20,
                            verbose = 1)
tensorboard_1 = TensorBoard(log_dir='graph_two', batch_size=512,update_freq='epoch')
callbacks_1 = [checkpoint_1,earlystop_1,tensorboard_1]


from keras.utils import plot_model
plot_model(model_two, to_file='model_two.png')
```


input_1: InputLayer
embedding_2: Embedding
school_state: InputLayer
teacher_prefix: InputLayer
clean_categories: InputLayer
clean_subcategories: InputLayer
lstm_1: LSTM
embedding_3: Embedding
embedding_4: Embedding
embedding_5: Embedding
embedding_6: Embedding
flatten_1: Flatten
flatten_2: Flatten
flatten_3: Flatten
flatten_4: Flatten
flatten_5: Flatten
concatenate_1: Concatenate
dense_2: Dense
dropout_1: Dropout
dense_3: Dense
dropout_2: Dropout
dense_4: Dense




```
# code source - https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-roc-and-auc-in-keras
import tensorflow as tf
from sklearn.metrics import roc_auc_score

def auroc(y_true, y_pred):
    return tf.py_function(roc_auc_score, (y_true, y_pred), tf.double)


train_one = [padded_essay_train, enc_school_state_train, enc_teacher_prefix_train, enc_cat_train, enc_subcat_train, enc_grade_train, numerical_fts_train]
test_one = [padded_essay_test, enc_school_state_test, enc_teacher_prefix_test, enc_cat_test, enc_subcat_test, enc_grade_test, numerical_fts_test]


from keras.utils import np_utils
```

```
from keras.utils import np_utils
y_train = np_utils.to_categorical(y_train, 2)
y_test = np_utils.to_categorical(y_test, 2)


model_two.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[auroc])
```

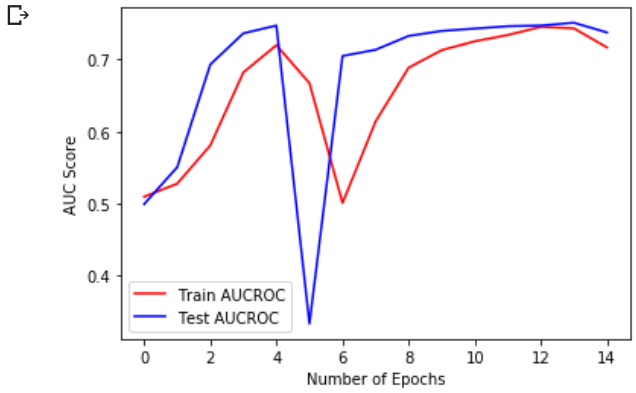
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer

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

```
h2 = model_two.fit(train_one, y_train, batch_size=512, epochs=15, validation_data=(test_one, y_test),verbose=1, callbacks=callbacks_1)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
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.assi

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.compat.v1.assign i

Train on 73196 samples, validate on 36052 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:1122: The name tf.summary.merge_all is deprecated. Please use tf.compat.v1.summary.merge_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:1125: The name tf.summary.FileWriter is deprecated. Please use tf.compat.v1.summary.FileW

Epoch 1/15
73196/73196 [==============================] - 310s 4ms/step - loss: 3.1843 - auroc: 0.5094 - val_loss: 2.4934 - val_auroc: 0.4992

Epoch 00001: val_auroc improved from -inf to 0.49915, saving model to model_one.h5
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:1265: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.

Epoch 2/15
73196/73196 [==============================] - 303s 4ms/step - loss: 2.1738 - auroc: 0.5276 - val_loss: 1.8887 - val_auroc: 0.5505

Epoch 00002: val_auroc improved from 0.49915 to 0.55049, saving model to model_one.h5
Epoch 3/15
73196/73196 [==============================] - 301s 4ms/step - loss: 1.6893 - auroc: 0.5807 - val_loss: 1.4961 - val_auroc: 0.6931

Epoch 00003: val_auroc improved from 0.55049 to 0.69308, saving model to model_one.h5
Epoch 4/15
73196/73196 [==============================] - 304s 4ms/step - loss: 1.3524 - auroc: 0.6822 - val_loss: 1.2153 - val_auroc: 0.7365

Epoch 00004: val_auroc improved from 0.69308 to 0.73655, saving model to model_one.h5
Epoch 5/15
73196/73196 [==============================] - 304s 4ms/step - loss: 1.1017 - auroc: 0.7200 - val_loss: 0.9963 - val_auroc: 0.7473

Epoch 00005: val_auroc improved from 0.73655 to 0.74733, saving model to model_one.h5
Epoch 6/15
73196/73196 [==============================] - 303s 4ms/step - loss: 0.9559 - auroc: 0.6673 - val_loss: 0.9537 - val_auroc: 0.3329

Epoch 00006: val_auroc did not improve from 0.74733
Epoch 7/15
73196/73196 [==============================] - 301s 4ms/step - loss: 0.8493 - auroc: 0.5007 - val_loss: 0.7763 - val_auroc: 0.7050

Epoch 00007: val_auroc did not improve from 0.74733
Epoch 8/15
73196/73196 [==============================] - 305s 4ms/step - loss: 0.7267 - auroc: 0.6141 - val_loss: 0.7033 - val_auroc: 0.7136

Epoch 00008: val_auroc did not improve from 0.74733
Epoch 9/15
73196/73196 [==============================] - 310s 4ms/step - loss: 0.6265 - auroc: 0.6888 - val_loss: 0.5873 - val_auroc: 0.7329

Epoch 00009: val_auroc did not improve from 0.74733
Epoch 10/15
73196/73196 [==============================] - 310s 4ms/step - loss: 0.5523 - auroc: 0.7131 - val_loss: 0.5420 - val_auroc: 0.7398

Epoch 00010: val_auroc did not improve from 0.74733
Epoch 11/15
```

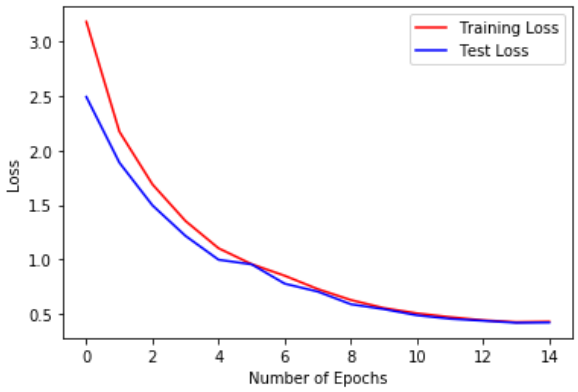
```
73196/73196 [==============================] - 305s 4ms/step - loss: 0.5040 - auroc: 0.7254 - val_loss: 0.4855 - val_auroc: 0.7431

Epoch 00011: val_auroc did not improve from 0.74733
Epoch 12/15
73196/73196 [==============================] - 309s 4ms/step - loss: 0.4691 - auroc: 0.7342 - val_loss: 0.4534 - val_auroc: 0.7464

Epoch 00012: val_auroc did not improve from 0.74733
Epoch 13/15
73196/73196 [==============================] - 313s 4ms/step - loss: 0.4403 - auroc: 0.7455 - val_loss: 0.4354 - val_auroc: 0.7474

Epoch 00013: val_auroc improved from 0.74733 to 0.74738, saving model to model_one.h5
Epoch 14/15
73196/73196 [==============================] - 301s 4ms/step - loss: 0.4231 - auroc: 0.7434 - val_loss: 0.4168 - val_auroc: 0.7512

Epoch 00014: val_auroc improved from 0.74738 to 0.75123, saving model to model_one.h5
Epoch 15/15
73196/73196 [==============================] - 303s 4ms/step - loss: 0.4274 - auroc: 0.7169 - val_loss: 0.4194 - val_auroc: 0.7377

Epoch 00015: val_auroc did not improve from 0.75123
```

```
fig,a = plt.subplots(1,1)
a.set_xlabel('Number of Epochs') ;
a.set_ylabel('AUC Score')

plt.plot(h2.history['auroc'], 'r')
plt.plot(h2.history['val_auroc'], 'b')
plt.legend({'Train AUCROC': 'r', 'Test AUCROC':'b'})
plt.show()
```




```
fig,a = plt.subplots(1,1)
a.set_xlabel('Number of Epochs') ;
```

```
a.set_ylabel('Loss')

plt.plot(h2.history['loss'], 'r')
plt.plot(h2.history['val_loss'], 'b')
plt.legend({'Training Loss': 'r', 'Test Loss':'b'})
plt.show()
```



The best score we could obtain for Model 1 is 0.751 with a loss of 0.416