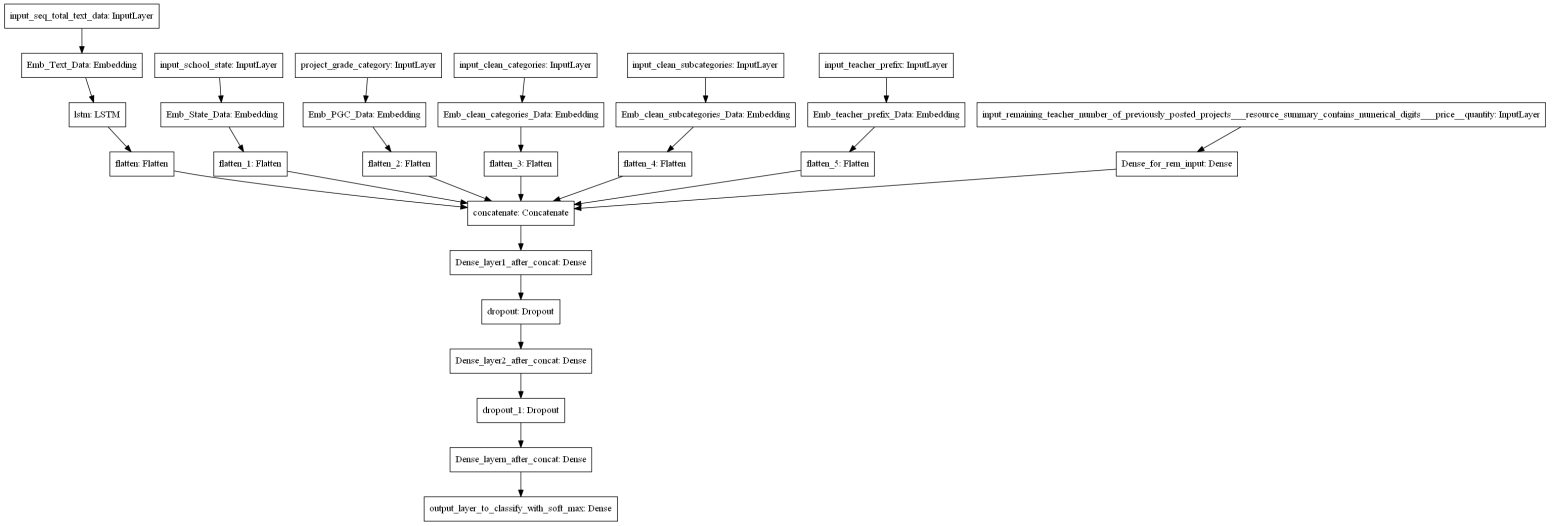
Assignment : 14

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)
```

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/

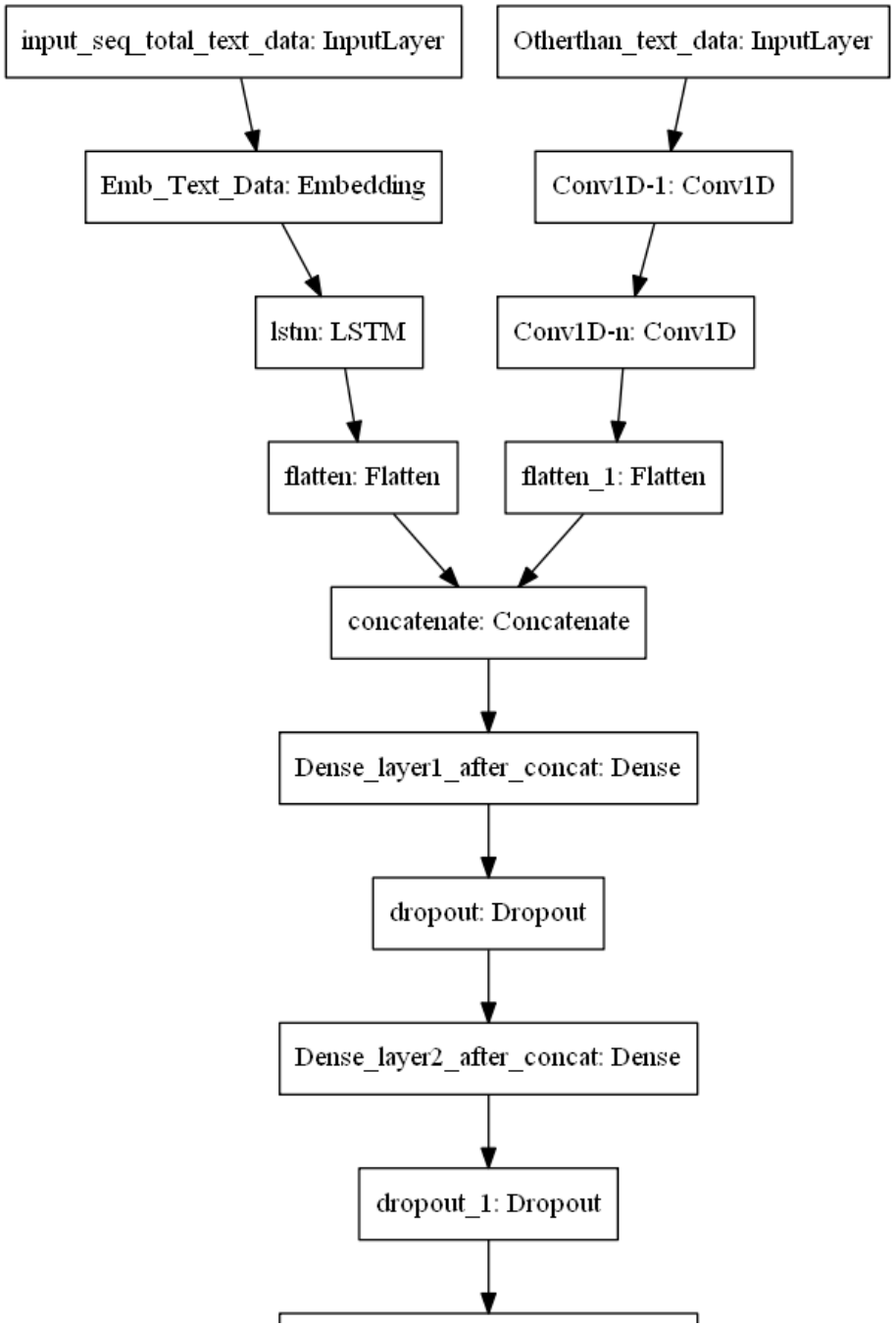
2. Please go through this link https://keras.io/getting-started/functional-api-guide/ and check the 'Multi-input and multi-output models' then you will get to know how to give multiple inputs.

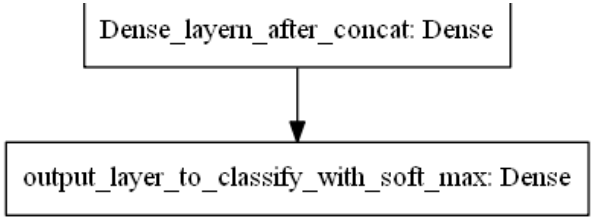
Model-2

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-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



ref: 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 as shown in above figure.

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

Model 3

```
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%
```

```
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
```

1.1 Reading Data

```
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)
```

```
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 [00:59<00:00, 1847.15it/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...	

```
# 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!
==================================================
Education Through Technology
==================================================
Teaching Math With Manipulatives
==================================================
Getting Our MOVE On!
==================================================
21st Century Skills and Technology Optimized to Improve OUR World!!!
==================================================
```

```
#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',\
            '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', \
```

```
'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, 38039.76it/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])
```

```
swim for life at ymca
==================================================
education through technology
==================================================
teaching math with manipulatives
==================================================
getting our move on
==================================================
digital magazine
```

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

1.5.1 Vectorizing Categorical data

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

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Ca
Shape of matrix after one hot encodig  (109248, 30)
```

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
vectorizer = CountVectorizer(binary=True)
school_state_count = vectorizer.fit_transform(project_data['school_state'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",school_state_count.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'n
Shape of matrix after one hot encodig  (109248, 51)
```

```
#Replacing spaces & hyphens in the text of project grade category with underscore
#converting Capital letters in the string to smaller letters
#Performing avalue count of project grade category
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-onp
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

```
#One hot encoding project grade category feature
vectorizer = CountVectorizer(binary=True)
project_grade_one = vectorizer.fit_transform(project_data['project_grade_category'].values)
```

```
project_grade_one = vectorizer.fit_transform(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encoding ",project_grade_one.shape)
```

```
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix after one hot encoding  (109248, 4)
```

```
# 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
```

```
#One hot encoding the teacher prefix column
vectorizer = CountVectorizer(binary=True)
teacher_prefix_one = vectorizer.fit_transform(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",teacher_prefix_one.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encodig  (109248, 5)
```

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

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
# a. Title, that describes your plot, this will be very helpful to the reader
# b. Legends if needed
# c. X-axis label
# d. Y-axis label
#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)

#Splitting training data into training & cross validation sets
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,
                                                stratify= y_train,
                                                test_size = 0.33)
```

Using Pre-trained Glove model for Embedding Text Data

```
from keras.preprocessing.text import Tokenizer
from numpy import zeros
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info.

```
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


#code source - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
token = Tokenizer()
token.fit_on_texts(X_train["essay"])
vocab_size = len(token.word_index) + 1
# integer encode the documents
seq_train = token.texts_to_sequences(X_train["essay"])
seq_test = token.texts_to_sequences(X_test["essay"])


# pad documents to a max length of 600 words
max_length = 600
padded_train = pad_sequences(seq_train, maxlen=max_length, padding='post')
padded_test = pad_sequences(seq_test, maxlen=max_length, padding='post')
padded_essay_train = padded_train
padded_essay_test = padded_test


# load the whole embedding into memory
embeddings_index = dict()
f = open('/content/drive/My Drive/glove.6B.300d.txt')
for line in f:
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

```
# create a weight matrix for words in training docs
embedding_matrix = zeros((vocab_size, 300))
for word, i in token.word_index.items():
```

```
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector


input_text = Input(shape=(600,),name="input_text")
emb_layer = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=max_length, trainable=False)
x = emb_layer(input_text)
x = LSTM(128, return_sequences=True)(x)
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

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

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

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.Conf

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is deprecated. Please use tf.compat.v1.Session

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initialized is deprecated. Please use tf.co

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer is deprecated. Please use tf.comp
```

One Hot Encoding Categorical Data

One Hot Encoding School State

```
#We use fit only for train data
vectorizer_state = CountVectorizer(binary=True)
vectorizer_state.fit(X_train['school_state'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer_state.transform(X_train['school_state'].values)
X_test_state_ohe = vectorizer_state.transform(X_test['school_state'].values)
print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer_state.get_feature_names())
print("="*75)
```

```
After vectorizations
(49041, 51) (49041,)
(36052, 51) (36052,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'n
==========================================================================
```

One Hot Encoding Teacher Prefix

```
# we use count vectorizer to convert the values into one
#We use fit only for train data
vectorizer_tp = CountVectorizer(binary=True)
vectorizer_tp.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer_tp.transform(X_train['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer_tp.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer_tp.get_feature_names())
print("="*50)
```

```
After vectorizations
(49041, 5) (49041,)
(36052, 5) (36052,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
==================================================
```

One Hot Encoding Subject Category

```
#We use fit only for train data
vectorizer_category = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), binary=True)
vectorizer_category.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cat_ohe = vectorizer_category.transform(X_train['clean_categories'].values)
X_test_cat_ohe = vectorizer_category.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_cat_ohe.shape, y_train.shape)
print(X_test_cat_ohe.shape, y_test.shape)
print(vectorizer_category.get_feature_names())
```

```
print("="*70)
```

```
After vectorizations
(49041, 9) (49041,)
(36052, 9) (36052,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
======================================================================
```

One Hot Encoding Subject Sub-Category

```
#We use fit only for train data
vectorizer_subcat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), binary=True)
vectorizer_subcat.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_subcat_ohe = vectorizer_subcat.transform(X_train['clean_subcategories'].values)
X_test_subcat_ohe = vectorizer_subcat.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_subcat_ohe.shape, y_train.shape)
print(X_test_subcat_ohe.shape, y_test.shape)
print(vectorizer_subcat.get_feature_names())
print("="*70)
```

```
After vectorizations
(49041, 30) (49041,)
(36052, 30) (36052,)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Ca
======================================================================
```

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

```
377
```

```
input_subcat = Input(shape=(1,),name="clean_subcategories")
embedded_subcat = Embedding(371, 5, trainable=True)(input_subcat)
flatten_subcat = Flatten()(embedded_subcat)
```

One Hot Encoding Project Grade Category

```
#Replacing spaces & hyphens in the text of project grade category with underscore
#converting Capital letters in the string to smaller letters
#Performing avalue count of project grade category
```

```
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-onproject_grade_category
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

```
#We use fit only for train data
vectorizer_grade = CountVectorizer()
vectorizer_grade.fit(X_train['project_grade_category'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer_grade.transform(X_train['project_grade_category'].values)
X_test_grade_ohe = vectorizer_grade.transform(X_test['project_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer_grade.get_feature_names())
print("="*70)
```

```
After vectorizations
(49041, 4) (49041,)
(36052, 4) (36052,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
======================================================================
```

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:
```

```
# 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, 49041) (49041,)
(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)
```

```
(49041, 1) (49041,)
(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.
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, 49041) (49041,)
(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
(49041, 1) (49041,)
(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
(1, 49041) (49041,)
(1, 36052) (36052,)
====================================================================================================
```

```
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)
```

```
(49041, 1) (49041,)
(36052, 1) (36052,)
====================================================================================================
```

```
num_fts_train = np.concatenate((X_train_price_norm, X_train_quantity_norm, X_train_tpp_norm))
```

```
num_fts_test = np.concatenate((X_test_price_norm, X_test_quantity_norm, X_test_tpp_norm))


from scipy.sparse import hstack
other_fts_train = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_cat_ohe, X_train_subcat_ohe, X_train_grade_ohe, X_train_price_norm, X_train_quantity_norm, X_train_tpp
other_fts_test = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_cat_ohe, X_test_subcat_ohe, X_test_grade_ohe, X_test_price_norm, X_test_quantity_norm, X_test_tpp_norm)).t
print("After stacking")
print(other_fts_train.shape, y_train.shape)
print(other_fts_test.shape, y_test.shape)
print("="*100)
```

```
After stacking
(49041, 102) (49041,)
(36052, 102) (36052,)
====================================================================================================
```

```
other_fts_test.shape
```

```
(36052, 102)
```

```
oth_fts_train = np.array(other_fts_train).reshape(49041,102,1)
oth_fts_test = np.array(other_fts_test).reshape(36052,102,1)


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.normalization import BatchNormalization
from keras.layers import SpatialDropout1D, LSTM, concatenate,Flatten,Embedding,MaxPooling2D,Reshape,Conv1D
from keras.layers import Dense, Activation
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from keras.layers import Dropout
input_oth=Input(shape=(oth_fts_train.shape[1],1), name="oth")
conv_one = Conv1D(128,kernel_size=3,activation='relu',kernel_initializer="he_normal")(input_oth)
conv_two = Conv1D(128,kernel_size=3,activation='relu',kernel_initializer="he_normal")(conv_one)
flatten1 = Flatten()(conv_two)
```

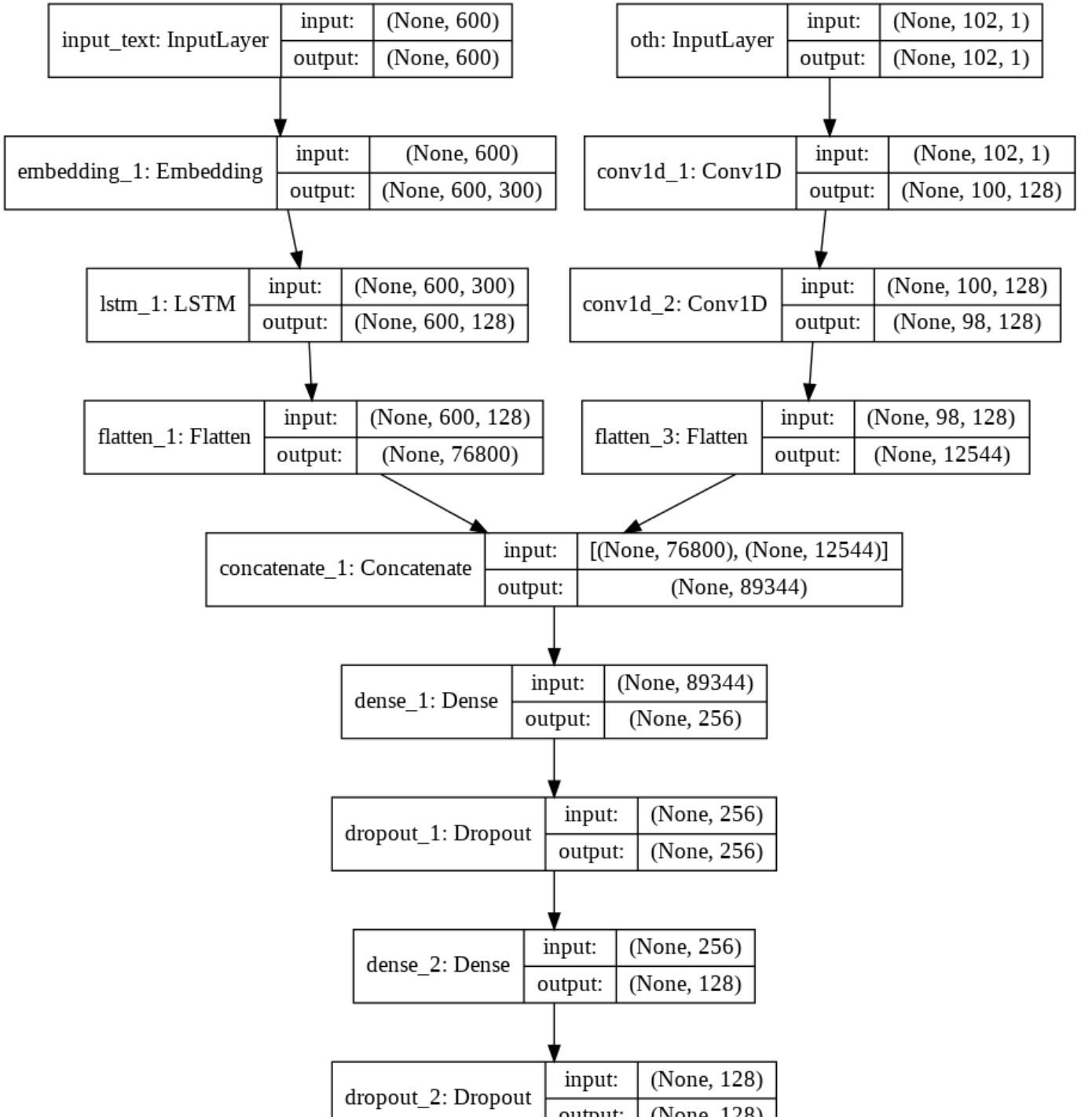
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4479: The name tf.truncated_normal is deprecated. Please use tf.random.t
```

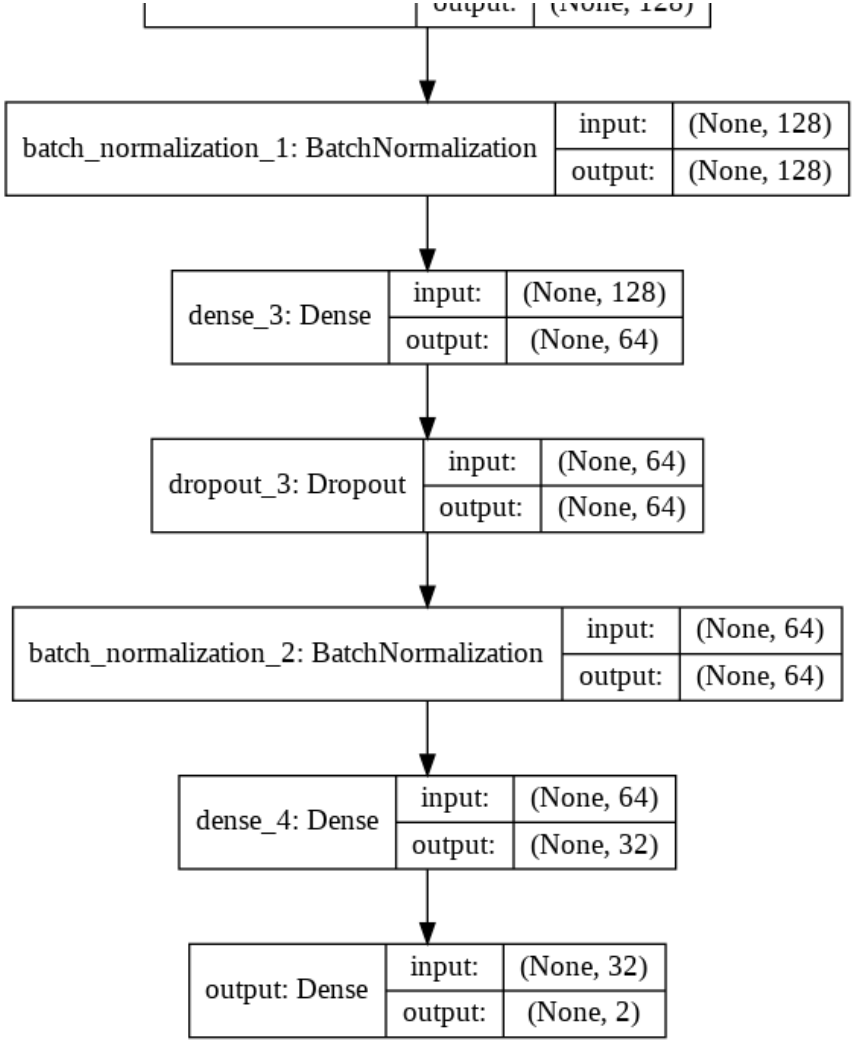
```
concatenated_fts=concatenate([flat_1,flatten1])
z1 = Dense(256,activation='relu')(concatenated_fts)
z2 = Dropout(0.3)(z1)

y = Dense(128,activation='relu',kernel_initializer="he_normal")(z2)
y = Dropout(0.3)(y)
y = BatchNormalization()(y)
```

```
z = Dense(64, activation='relu',kernel_initializer="he_normal")(y)
z = Dropout(0.3)(z)
z = BatchNormalization()(z)

z3 = Dense(32, activation='relu',kernel_initializer="he_normal")(z)

output = Dense(2, activation='softmax', name='output')(z3)
model_three = Model(inputs=[input_text,input_oth],outputs=[output])
print(model_three.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_text (InputLayer)         (None, 600)          0
__________________________________________________________________________________________________
oth (InputLayer)                (None, 102, 1)       0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 600, 300)     14396100    input_text[0][0]
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 100, 128)     512         oth[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 600, 128)     219648      embedding_1[0][0]
__________________________________________________________________________________________________
conv1d_2 (Conv1D)               (None, 98, 128)      49280       conv1d_1[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 76800)        0           lstm_1[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 12544)        0           conv1d_2[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 89344)        0           flatten_1[0][0]
                                                                 flatten_3[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          22872320    concatenate_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 256)          0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 128)          32896       dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           dense_2[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 128)          512         dropout_2[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 64)           8256        batch_normalization_1[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 64)           0           dense_3[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 64)           256         dropout_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 32)           2080        batch_normalization_2[0][0]
__________________________________________________________________________________________________
output (Dense)                  (None, 2)            66          dense_4[0][0]
==================================================================================================
Total params: 37,581,926
Trainable params: 23,185,442
Non-trainable params: 14,396,484
__________________________________________________________________________________________________
None
```

```
#https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/
from keras.utils.vis_utils import plot_model
plot_model(model_three, to_file='model_3.png', show_shapes=True, show_layer_names=True)
```


input_text: InputLayer
input: (None, 600)
output: (None, 600)
oth: InputLayer
input: (None, 102, 1)
output: (None, 102, 1)
embedding_1: Embedding
input: (None, 600)
output: (None, 600, 300)
conv1d_1: Conv1D
input: (None, 102, 1)
output: (None, 100, 128)
lstm_1: LSTM
input: (None, 600, 300)
output: (None, 600, 128)
conv1d_2: Conv1D
input: (None, 100, 128)
output: (None, 98, 128)
flatten_1: Flatten
input: (None, 600, 128)
output: (None, 76800)
flatten_3: Flatten
input: (None, 98, 128)
output: (None, 12544)
concatenate_1: Concatenate
input: [(None, 76800), (None, 12544)]
output: (None, 89344)
dense_1: Dense
input: (None, 89344)
output: (None, 256)
dropout_1: Dropout
input: (None, 256)
output: (None, 256)
dense_2: Dense
input: (None, 256)
output: (None, 128)
dropout_2: Dropout
input: (None, 128)



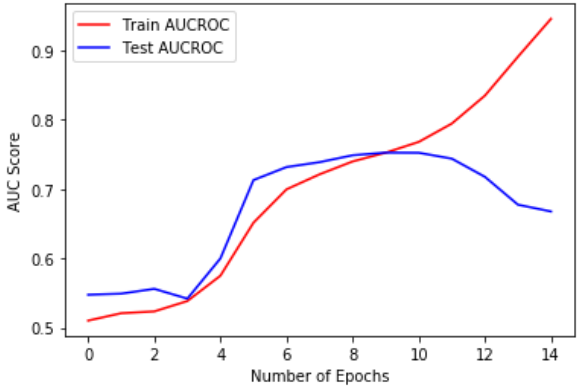

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
checkpoint_three = ModelCheckpoint("model_three.h5",
                monitor="val_auroc",
                mode="max",
                save_best_only = True,
                verbose=1)
earlystop_three = EarlyStopping(monitor = 'val_auroc',
              mode="max",
              min_delta = 0,
```

```
                patience = 5,
                verbose = 1,)
tensorboard_three = TensorBoard(log_dir='graph_4', histogram_freq=0, batch_size=256, update_freq='epoch')

callbacks_three = [checkpoint_three,earlystop_three,tensorboard_three]


train_three = [padded_essay_train,oth_fts_train]
test_three = [padded_essay_test,oth_fts_test]


# 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)


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_three.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[auroc])
h3 = model_three.fit(train_three, y_train, batch_size=512, epochs=15, validation_data=(test_three, y_test), verbose=1,callbacks=callbacks_three)
```

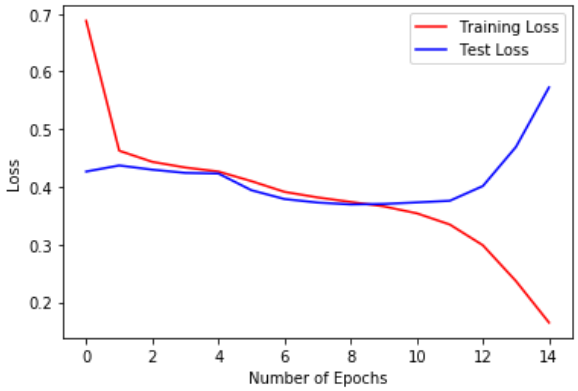
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer

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.

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 49041 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
49041/49041 [==============================] - 180s 4ms/step - loss: 0.6879 - auroc: 0.5102 - val_loss: 0.4266 - val_auroc: 0.5475

Epoch 00001: val_auroc improved from -inf to 0.54747, saving model to model_three.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
49041/49041 [==============================] - 176s 4ms/step - loss: 0.4627 - auroc: 0.5208 - val_loss: 0.4371 - val_auroc: 0.5490

Epoch 00002: val_auroc improved from 0.54747 to 0.54904, saving model to model_three.h5
Epoch 3/15
49041/49041 [==============================] - 176s 4ms/step - loss: 0.4434 - auroc: 0.5234 - val_loss: 0.4298 - val_auroc: 0.5560

Epoch 00003: val_auroc improved from 0.54904 to 0.55598, saving model to model_three.h5
Epoch 4/15
49041/49041 [==============================] - 177s 4ms/step - loss: 0.4335 - auroc: 0.5383 - val_loss: 0.4243 - val_auroc: 0.5418

Epoch 00004: val_auroc did not improve from 0.55598
Epoch 5/15
49041/49041 [==============================] - 174s 4ms/step - loss: 0.4266 - auroc: 0.5749 - val_loss: 0.4231 - val_auroc: 0.5998

Epoch 00005: val_auroc improved from 0.55598 to 0.59981, saving model to model_three.h5
Epoch 6/15
49041/49041 [==============================] - 174s 4ms/step - loss: 0.4099 - auroc: 0.6514 - val_loss: 0.3942 - val_auroc: 0.7129

Epoch 00006: val_auroc improved from 0.59981 to 0.71293, saving model to model_three.h5
Epoch 7/15
49041/49041 [==============================] - 174s 4ms/step - loss: 0.3913 - auroc: 0.6996 - val_loss: 0.3789 - val_auroc: 0.7317

Epoch 00007: val_auroc improved from 0.71293 to 0.73172, saving model to model_three.h5
Epoch 8/15
49041/49041 [==============================] - 174s 4ms/step - loss: 0.3817 - auroc: 0.7213 - val_loss: 0.3728 - val_auroc: 0.7388

Epoch 00008: val_auroc improved from 0.73172 to 0.73879, saving model to model_three.h5
Epoch 9/15
49041/49041 [==============================] - 175s 4ms/step - loss: 0.3739 - auroc: 0.7398 - val_loss: 0.3696 - val_auroc: 0.7488

Epoch 00009: val_auroc improved from 0.73879 to 0.74876, saving model to model_three.h5
Epoch 10/15
```

```
49041/49041 [==============================] - 175s 4ms/step - loss: 0.3662 - auroc: 0.7525 - val_loss: 0.3704 - val_auroc: 0.7526

Epoch 00010: val_auroc improved from 0.74876 to 0.75258, saving model to model_three.h5
Epoch 11/15
49041/49041 [==============================] - 174s 4ms/step - loss: 0.3542 - auroc: 0.7680 - val_loss: 0.3731 - val_auroc: 0.7524

Epoch 00011: val_auroc did not improve from 0.75258
Epoch 12/15
49041/49041 [==============================] - 176s 4ms/step - loss: 0.3347 - auroc: 0.7945 - val_loss: 0.3759 - val_auroc: 0.7438

Epoch 00012: val_auroc did not improve from 0.75258
Epoch 13/15
49041/49041 [==============================] - 176s 4ms/step - loss: 0.2988 - auroc: 0.8348 - val_loss: 0.4015 - val_auroc: 0.7176

Epoch 00013: val_auroc did not improve from 0.75258
Epoch 14/15
49041/49041 [==============================] - 177s 4ms/step - loss: 0.2367 - auroc: 0.8910 - val_loss: 0.4693 - val_auroc: 0.6774

Epoch 00014: val_auroc did not improve from 0.75258
Epoch 15/15
49041/49041 [==============================] - 176s 4ms/step - loss: 0.1647 - auroc: 0.9457 - val_loss: 0.5725 - val_auroc: 0.6676

Epoch 00015: val_auroc did not improve from 0.75258
Epoch 00015: early stopping
```

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

plt.plot(h3.history['auroc'], 'r')
plt.plot(h3.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(h3.history['loss'], 'r')
plt.plot(h3.history['val_loss'], 'b')
plt.legend({'Training Loss': 'r', 'Test Loss':'b'})
plt.show()
```



The best score we got for Model 3 is 0.752 with a corresponding loss of 0.37.

```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names=["Serial No.", "Model name", "Train_auc", "Test_auc"]
x.add_row(["1", "Model-1", "0.743", "0.751"])
x.add_row(["2", "Model-2", "0.759", "0.759"])
x.add_row(["3", "Model-3", "0.752", "0.752"])
print(x)
```

```
+------------+------------+-----------+----------+
| Serial No. | Model name | Train_auc | Test_auc |
+------------+------------+-----------+----------+
|     1      |  Model-1   |   0.743   |  0.751   |
|     2      |  Model-2   |   0.759   |  0.759   |
|     3      |  Model-3   |   0.752   |  0.752   |
+------------+------------+-----------+----------+
```

Conclusion :-

Scores for all the 3 Models are very close to each other with Model 2 being slightly better than the rest.