DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:**

	• `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

from tqdm import tqdm
import os

import datetime
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import roc_curve, auc

import re
import pickle
from sklearn.model_selection import train_test_split
```

After Preprocessing

Reading Data

In [2]:
```
project_data = pd.read_csv('preprocessed_data.csv')
```

In [3]:
```
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, 9)
--------------------------------------------------
The attributes of data : ['school_state' 'teacher_prefix' 'project_grade_cat
egory'
 'teacher_number_of_previously_posted_projects' 'project_is_approved'
 'clean_categories' 'clean_subcategories' 'essay' 'price']
```

In [4]:
```
project_data.head()
```

Out[4]:

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

In [5]:
```
resource = pd.read_csv('resources.csv')

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

```
Number of data points in resource data (1541272, 4)
--------------------------------------------------
The attributes of data : ['id' 'description' 'quantity' 'price']
```

In [6]:
```
resource.head()
```

Out[6]:

	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
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [7]:
```
# Function to check the presence of number in string
def presence_of_num(sent):
    count = 0
    for word in (str(sent).split()):
        if (bool(re.match('^[0-9]+', str(word)))):
            count=1
    return count

resource['presence_of_number_in_description']=resource.description.apply(lambd
a x: presence_of_num(x))
```

In [8]:
```
# merging resourse data and preprocessed data based on unique ID

data = pd.read_csv('train_data.csv')
project_data['id']=data.id
del data

data_after_merged = pd.merge(project_data,
                             resource.groupby(["id"]).sum().drop(["price"],axis
=1),
                             how='inner',
                             on='id',)

print(f"shape: {data_after_merged.shape}\n")
data_after_merged.head()
```

```
shape: (109248, 12)
```

Out[8]:

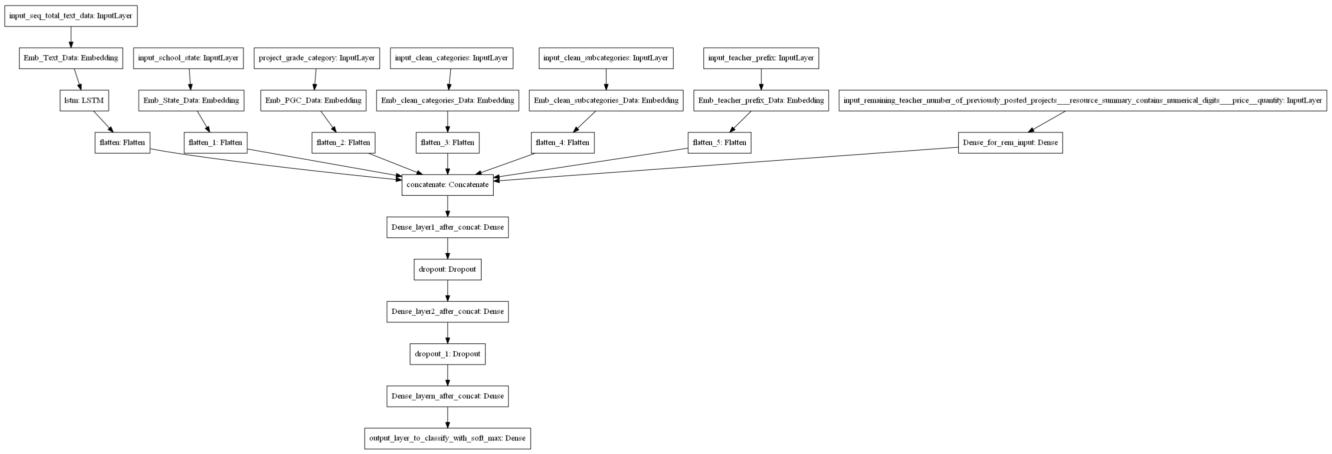
	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	p
0	ca	mrs	grades_prek_2	53	
1	ut	ms	grades_3_5	4	
2	ca	mrs	grades_prek_2	10	
3	ga	mrs	grades_prek_2	2	
4	wa	mrs	grades_3_5	2	

Assignment :

```
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 hav
e 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 video.
7. For all the model's use TensorBoard and plot the Metric value and Los
s with epoch. While submitting, take a screenshot of plots and include t
hose images in .ipynb notebook and PDF.
8. Use Categorical Cross Entropy as Loss to minimize.
```

Model-1

Build and Train deep neural network as shown below


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

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

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

In [9]:
```python
#import tensorflow
#from tensorflow import keras

from numpy import asarray
from numpy import zeros
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras import Model
from tensorflow.keras.layers import Input,concatenate,Dropout
from tensorflow.keras import Sequential,layers
from tensorflow.keras.callbacks import ModelCheckpoint,TensorBoard
import tensorflow as tf
import numpy as np
from sklearn.metrics import roc_auc_score
```

Data Preparation for training

In [10]:
```python
# one hot encoding of class label
classlabel =project_data.project_is_approved
classlabel_ohe = to_categorical(classlabel)

# Train Test Split
X_train, X_cv, y_train, y_cv = train_test_split(data_after_merged.drop(["proje
ct_is_approved",'id'],axis=1), classlabel_ohe, test_size=0.2,stratify= classla
bel, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_siz
e=0.2, random_state=42)

print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(69918, 10)
(21850, 10)
(17480, 10)
```

In [11]:
```python
X_train.head()
```

Out[11]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_project
37914	fl	mrs	grades_prek_2	
69690	wv	mr	grades_3_5	
34920	ca	mrs	grades_prek_2	

64621	az	mrs	grades_3_5	1
1410	md	ms	grades_6_8	

1.Training Using Model 1

1.1 preparing essay feature

In [12]:
```
# Utility function
# Finding maximum length of sentence in the corpus

def max_len(doc):
    temp=0
    for sent in doc:
        if (len(sent.split()))>temp:
            temp = len(sent.split())
    return temp

max_len_of_sent=max_len(X_train.essay)
print(f'maximumn length sentece in corpus:{max_len_of_sent}')
```

```
maximumn length sentece in corpus:331
```

In [13]:
```
# refer: https://machinelearningmastery.com/use-word-embedding-layers-deep-lea
rning-keras/

# define documents
train_essay = X_train.essay
cv_essay = X_cv.essay
test_essay = X_test.essay

# prepare tokenizer
t = Tokenizer()
t.fit_on_texts(train_essay)
vocab_size_essay = len(t.word_index) + 1
# integer encode the documents
train_encoded_essay = t.texts_to_sequences(train_essay)
cv_encoded_essay = t.texts_to_sequences(cv_essay)
test_encoded_essay = t.texts_to_sequences(test_essay)
print(train_encoded_essay)
```

```
# pad documents to a max length of words :339 in our case
max_length = max_len_of_sent
train_padded_essay = pad_sequences(train_encoded_essay, maxlen=max_length, pad
ding='post')
cv_padded_essay = pad_sequences(cv_encoded_essay, maxlen=max_length, padding=
'post')
test_padded_essay = pad_sequences(test_encoded_essay, maxlen=max_length, paddi
ng='post')
print(train_padded_essay)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
[[  272    20     2 ...     0     0     0]
 [  106     1    72 ...     0     0     0]
 [    4     1   382 ...     0     0     0]
 ...
 [    2 11025    76 ...     0     0     0]
 [    4    62   398 ...     0     0     0]
 [    2  1388    79 ...     0     0     0]]
```

In [ ]:

```
# load the whole embedding into memory
embeddings_index = dict()
f = open('glove.42B.300d.txt',encoding='utf-8')
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))
# create a weight matrix for words in training docs
embedding_matrix = zeros((train_encoded_essay, 300))
for word, i in t.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [14]:

```
"""# Saving embedding matrix
file = open("embedding_matrix",'wb')
np.save(file,embedding_matrix)
file.close()
"""
with open("embedding_matrix",'rb') as f:
    embedding_matrix=np.load(f)
```

1.2. Preparing school_state data for training

In [15]:

```
from sklearn.preprocessing import LabelEncoder
```

```
lb=LabelEncoder()

train_school_state_label_encoded = lb.fit_transform(X_train.school_state).resh
ape(-1,1)
cv_school_state_label_encoded = lb.transform(X_cv.school_state).reshape(-1,1)
test_school_state_label_encoded = lb.transform(X_test.school_state).reshape(-1
,1)

print(f"before: \n{X_test.school_state.iloc[:5]}\n")
print(f"after: \n{test_school_state_label_encoded[:5]}")
```

```
before:
73624    tx
2158     mi
74759    nv
89804    ma
18127    ca
Name: school_state, dtype: object

after:
[[43]
 [22]
 [33]
 [19]
 [ 4]]
```

1.3. Preparing teacher_prefix data for training

In [16]:

```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()

train_teacher_prefix_label_encoded = lb.fit_transform(X_train.teacher_prefix).
reshape(-1,1)
cv_teacher_prefix_label_encoded = lb.transform(X_cv.teacher_prefix).reshape(-1
,1)
test_teacher_prefix_label_encoded = lb.transform(X_test.teacher_prefix).reshap
e(-1,1)

print(f"before: \n{X_test.teacher_prefix.iloc[:5]}\n")
print(f"after: \n{test_teacher_prefix_label_encoded[:5]}")
```

```
before:
73624     ms
2158     mrs
74759    mrs
89804    mrs
18127    mrs
Name: teacher_prefix, dtype: object

after:
[[3]
 [2]
 [2]
 [2]
 [2]]
```

1.4. Preparing project_grade_category data for training

```
In [17]: np.unique(project_data.project_grade_category)

from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()

train_grade_label_encoded = lb.fit_transform(X_train.project_grade_category).r
eshape(-1,1)
cv_grade_label_encoded = lb.transform(X_cv.project_grade_category).reshape(-1,
1)
test_grade_label_encoded = lb.transform(X_test.project_grade_category).reshape
(-1,1)

print(f"before: \n{X_test.project_grade_category.iloc[:5]}\n")
print(f"after: \n{test_grade_label_encoded[:5]}")
```

```
before:
73624    grades_prek_2
2158     grades_prek_2
74759    grades_prek_2
89804       grades_3_5
18127    grades_prek_2
Name: project_grade_category, dtype: object

after:
[[3]
 [3]
 [3]
 [0]
 [3]]
```

1.5. Preparing clean_categories data for training

```
In [18]: # refer: https://machinelearningmastery.com/use-word-embedding-layers-deep-lea
rning-keras/

# prepare tokenizer
t = Tokenizer(filters='\t')
t.fit_on_texts(X_train.clean_categories)
# integer encode the documents
train_encoded_clean_categories = t.texts_to_sequences( X_train.clean_categorie
s)
cv_encoded_clean_categories  = t.texts_to_sequences( X_cv.clean_categories)
test_encoded_clean_categories = t.texts_to_sequences( X_test.clean_categories)

# vocab size
category_vocab_size = len(t.word_counts)+1

# pad documents to a max length of subcategories :3 in our case
max_length = max_len(X_train.clean_categories)

train_padded_clean_categories = pad_sequences(train_encoded_clean_categories,
maxlen=max_length, padding='post')
cv_padded_clean_categories = pad_sequences(cv_encoded_clean_categories, maxlen
=max_length, padding='post')
test_padded_clean_categories = pad_sequences(test_encoded_clean_categories, ma
xlen=max_length, padding='post')
print(X_test.clean_categories.iloc[:5])
print(test_padded_clean_categories[:5])
```

```
73624                   literacy_language
2158      literacy_language math_science
```

```
74759                  literacy_language
89804                  literacy_language
18127                    appliedlearning
Name: clean_categories, dtype: object
[[1 0 0]
 [1 2 0]
 [1 0 0]
 [1 0 0]
 [5 0 0]]
```

In [19]:

```python
print(f"Encoded word index foe subcategories:\n")
t.index_word
```

```
Encoded word index foe subcategories:
```

Out[19]:

```
{1: 'literacy_language',
 2: 'math_science',
 3: 'health_sports',
 4: 'specialneeds',
 5: 'appliedlearning',
 6: 'music_arts',
 7: 'history_civics',
 8: 'warmth',
 9: 'care_hunger'}
```

1.6. Preparing clean_subcategories data for training

In [20]:

```python
# refer: https://machinelearningmastery.com/use-word-embedding-layers-deep-lea
rning-keras/
# define documents
train_clean_subcategories = X_train.clean_subcategories
cv_clean_subcategories = X_cv.clean_subcategories
test_clean_subcategories = X_test.clean_subcategories

# prepare tokenizer
t = Tokenizer(filters='\t')
t.fit_on_texts(train_clean_subcategories)
# integer encode the documents
train_encoded_clean_subcategories = t.texts_to_sequences(train_clean_subcatego
ries)
cv_encoded_clean_subcategories = t.texts_to_sequences(cv_clean_subcategories)
test_encoded_clean_subcategories = t.texts_to_sequences(test_clean_subcategori
es)

sub_category_vocab_size = len(t.word_counts)+1

# pad documents to a max length of subcategories :3 in our case
max_length = 3
train_padded_clean_subcategories = pad_sequences(train_encoded_clean_subcatego
ries, maxlen=max_length, padding='post')
cv_padded_clean_subcategories = pad_sequences(cv_encoded_clean_subcategories,
maxlen=max_length, padding='post')
test_padded_clean_subcategories = pad_sequences(test_encoded_clean_subcategori
es, maxlen=max_length, padding='post')
print(X_train.clean_categories.iloc[:5])
print(train_padded_clean_subcategories[:5])
```

```
37914      appliedlearning specialneeds
69690                  literacy_language
```

```
34920    literacy_language math_science
64621                    health_sports
1410                        music_arts
Name: clean_categories, dtype: object
[[11  4  0]
 [ 1  0  0]
 [ 1  2  0]
 [ 9 17  0]
 [14 19  0]]
```

In [21]:
```
print(f"Encoded word index foe subcategories:\n\n{t.index_word}")
```

```
Encoded word index foe subcategories:

{1: 'literacy', 2: 'mathematics', 3: 'literature_writing', 4: 'specialneed
s', 5: 'appliedsciences', 6: 'health_wellness', 7: 'visualarts', 8: 'environ
mentalscience', 9: 'gym_fitness', 10: 'esl', 11: 'earlydevelopment', 12: 'he
alth_lifescience', 13: 'history_geography', 14: 'music', 15: 'college_career
prep', 16: 'other', 17: 'teamsports', 18: 'charactereducation', 19: 'perform
ingarts', 20: 'socialsciences', 21: 'warmth', 22: 'care_hunger', 23: 'nutrit
ioneducation', 24: 'foreignlanguages', 25: 'civics_government', 26: 'extracu
rricular', 27: 'parentinvolvement', 28: 'financialliteracy', 29: 'communitys
ervice', 30: 'economics'}
```

Remaining Feature will be as it is used in Model with using keras concetenation layer

1.7 Modeling on All features

Model -1

In [124]:
```
#initialisation of vocab size and shapes of all input features

# Vocab size of each feature for Embeding Layer
vocab_size_essay = vocab_size_essay
category_vocab_size = category_vocab_size
sub_category_vocab_size = sub_category_vocab_size
train_school_state_label_encoded_vocab_size = len(np.unique(train_school_state
_label_encoded))
train_teacher_prefix_label_encoded_vocab_size = len(np.unique(train_teacher_pr
efix_label_encoded))
train_grade_label_encoded_vocab_size = len(np.unique(train_grade_label_encoded
))

# shapes of input of each feaures for Input Layer
train_padded_essay_shape = train_padded_essay.shape
train_school_state_label_encoded_shape = train_school_state_label_encoded.shap
e
train_teacher_prefix_label_encoded_shape = train_teacher_prefix_label_encoded.
shape
train_grade_label_encoded_shape = train_grade_label_encoded.shape
train_padded_clean_subcategories_shape = train_padded_clean_subcategories.shap
e
train_padded_clean_categories_shape = train_padded_clean_categories.shape
train_price_shape = X_train['price'].shape
train_teacher_number_of_previously_posted_projects_shape = X_train["teacher_nu
mber_of_previously_posted_projects"].shape
```

```
In [129]: # Model 1 using Keras Functional Api

          tf.keras.backend.clear_session()
          # Eaasy
          input1= Input(shape=(train_padded_essay_shape[1],) ,name='essay_input' )
          # Essay embed layer using pretrained Glove embedings of 300 dim
          Embeded_layer1 = Embedding(vocab_size_essay, output_dim=300, weights=[embeddin
          g_matrix], input_length=339, trainable=False)(input1)
          # Lstm layer for essay
          lstm_layer1 = (layers.LSTM(units=32,return_sequences=True))(Embeded_layer1)
          flatten1 = Flatten()(lstm_layer1)

          # school state
          input2= Input(shape=(train_school_state_label_encoded_shape[1],) ,name='school
          _state_input')
          Embeded_layer2= Embedding(train_school_state_label_encoded_vocab_size, output_
          dim=100, input_length=1)(input2)
          flatten2 = Flatten()(Embeded_layer2)

          # teacher_prefix
          input3= Input(shape=(train_teacher_prefix_label_encoded_shape[1],) ,name='teac
          her_prefix')
          Embeded_layer3= Embedding(train_school_state_label_encoded_vocab_size, output_
          dim=100, input_length=1)(input3)
          flatten3 = Flatten()(Embeded_layer3)

          # project_grade_category
          input4= Input(shape=(train_grade_label_encoded_shape[1],) ,name='project_grade
          _category')
          Embeded_layer4 = Embedding(train_grade_label_encoded_vocab_size, output_dim=10
          0, input_length=1)(input4)
          flatten4 = Flatten()(Embeded_layer4)

          # clean_subcategories
          input5= Input(shape=(train_padded_clean_categories_shape[1],) ,name='clean_cat
          egories')
          Embeded_layer5 = Embedding(category_vocab_size, output_dim=150, input_length=3
          )(input5)
          flatten5 = Flatten()(Embeded_layer5)

          # clean_subcategories
          input6= Input(shape=(train_padded_clean_subcategories_shape[1],) ,name='clean_
          subcategories')
          Embeded_layer6 = Embedding(sub_category_vocab_size, output_dim=150, input_leng
          th=3)(input6)
          flatten6 = Flatten()(Embeded_layer6)

          input7 = Input(shape=(1,) ,name='price')
          input8 = Input(shape=(1,) ,name='teacher_number_of_previously_posted_projects'
          )
          input9 = Input(shape=(1,) ,name='quantity')
          input10 = Input(shape=(1,) ,name='presence_of_num')

          concat_input_7_8_9_10 = concatenate([input7,input8,input9,input10], name='conc
          at_input7_input8')
          dense1 = Dense(32,activation='relu',kernel_initializer='he_uniform')(concat_in
          put_7_8_9_10)

          # Concatenation of all output of previous layers
          concatenate2 = concatenate([flatten1,flatten2,flatten3,flatten4,flatten5,dense
          1], name="concat_all_output")
```

```
# Dense layer 2 after concatenation of all output
dense_layer2 = Dense(units=128 ,activation='relu' ,kernel_initializer='he_unif
orm')(concatenate2)
dropout1=Dropout(rate=0.7)(dense_layer2)
batch_normalize1 = layers.BatchNormalization()(dropout1)


# Dense layer 3 after concatenation of all output
dense_layer3 = Dense(units=512 ,activation='relu' ,kernel_initializer='he_unif
orm')(batch_normalize1)
dropout2 = Dropout(rate=0.5)(dense_layer3)
batch_normalize2=layers.BatchNormalization()(dropout2)


# Dense layer 3 after concatenation of all output
dense_layer4 = Dense(units=128 ,activation='relu' ,kernel_initializer='he_unif
orm')(batch_normalize2)
dropout2 = Dropout(rate=0.5)(dense_layer4)

#output
output = Dense(units=2 ,activation='softmax' ,kernel_initializer='glorot_unifo
rm', name="output_layer")(dropout2)

Model1 = Model(inputs=[input1,input2,input3,input4,input5,input6,input7,input8
,input9,input10], outputs=output)
print(Model1.summary())
```

```
WARNING:tensorflow:Large dropout rate: 0.7 (>0.5). In TensorFlow 2.x, drop
out() uses dropout rate instead of keep_prob. Please ensure that this is i
ntended.
Model: "model"
____________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
============================================================================
========================
essay_input (InputLayer)        [(None, 331)]        0
____________________________________________________________________________
________________________
embedding (Embedding)           (None, 331, 300)     14232600    essay_inp
ut[0][0]
____________________________________________________________________________
________________________
school_state_input (InputLayer) [(None, 1)]          0
____________________________________________________________________________
________________________
teacher_prefix (InputLayer)     [(None, 1)]          0
____________________________________________________________________________
________________________
project_grade_category (InputLa [(None, 1)]          0
____________________________________________________________________________
________________________
clean_categories (InputLayer)   [(None, 3)]          0
____________________________________________________________________________
________________________
price (InputLayer)              [(None, 1)]          0
____________________________________________________________________________
________________________
teacher_number_of_previously_po [(None, 1)]          0
____________________________________________________________________________
________________________
```

```
quantity (InputLayer)           [(None, 1)]          0
__________________________________________________________________________________________________
presence_of_num (InputLayer)    [(None, 1)]          0
__________________________________________________________________________________________________
bidirectional (Bidirectional)   (None, 331, 64)      85248       embedding
[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 1, 100)       5100        school_st
ate_input[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 1, 100)       5100        teacher_p
refix[0][0]
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 1, 100)       400         project_g
rade_category[0][0]
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, 3, 150)       1500        clean_cat
egories[0][0]
__________________________________________________________________________________________________
concat_input7_input8 (Concatena (None, 4)            0           price[0]
[0]
                                                                 teacher_n
umber_of_previously_post
                                                                 quantity
[0][0]
                                                                 presence_
of_num[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 21184)        0           bidirecti
onal[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 100)          0           embedding
_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 100)          0           embedding
_2[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 100)          0           embedding
_3[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 450)          0           embedding
_4[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 32)           160         concat_in
put7_input8[0][0]
__________________________________________________________________________________________________
concat_all_output (Concatenate) (None, 21966)        0           flatten
[0][0]
```

```
                                                                   flatten_1
[0][0]
                                                                   flatten_2
[0][0]
                                                                   flatten_3
[0][0]
                                                                   flatten_4
[0][0]
                                                                   dense[0]
[0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          2811776     concat_al
l_output[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 128)          0           dense_1
[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 128)          512         dropout
[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 512)          66048       batch_nor
malization[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 512)          0           dense_2
[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 512)          2048        dropout_1
[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 128)          65664       batch_nor
malization_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           dense_3
[0][0]
__________________________________________________________________________________________________
clean_subcategories (InputLayer [(None, 3)]          0
__________________________________________________________________________________________________
output_layer (Dense)            (None, 2)            258         dropout_2
[0][0]
==================================================================================================
Total params: 17,276,414
Trainable params: 3,042,534
Non-trainable params: 14,233,880
__________________________________________________________________________________________________
None
```

In [130]:
```
train=[train_padded_essay,train_school_state_label_encoded,train_teacher_prefi
x_label_encoded,train_grade_label_encoded,train_padded_clean_categories,
       train_padded_clean_subcategories,X_train["price"],X_train["teacher_numb
er_of_previously_posted_projects"],X_train['quantity'],
       X_train['presence_of_number_in_description']]

cv = [cv_padded_essay,cv_school_state_label_encoded,cv_teacher_prefix_label_en
coded,cv_grade_label_encoded,cv_padded_clean_categories,
      cv_padded_clean_subcategories,X_cv["price"],X_cv["teacher_number_of_prev
iously_posted_projects"],X_cv['quantity'],
      X_cv['presence_of_number_in_description']]

test = [test_padded_essay,test_school_state_label_encoded,test_teacher_prefix_
label_encoded,test_grade_label_encoded,test_padded_clean_categories,
        test_padded_clean_subcategories,X_test["price"],X_test["teacher_number
_of_previously_posted_projects"],X_test['quantity'],
        X_test['presence_of_number_in_description']]
```

1.8 Training

In [131]:
```
# Traning using Model-1
import math
# defining custom ROC AUC for keras ,refer: https://stackoverflow.com/a/517349
92
from sklearn.metrics import roc_auc_score
def auroc(y_true, y_pred):
    return tf.compat.v1.py_func(roc_auc_score, (y_true, y_pred), tf.double)


# Checkpoint callbacks
check_point = ModelCheckpoint(
    filepath='weights.{epoch:02d}-{val_loss:.2f}.hdf5',
    monitor='val_auroc',
    verbose=1,
    save_best_only=True,
    save_weights_only=True,
    mode='max',
    period=5)


# TensorBoard
logs_base_dir = ".\logs"
os.makedirs(logs_base_dir, exist_ok=True)
logdir = os.path.join(logs_base_dir, datetime.datetime.now().strftime("%Y%m%d-
%H%M%S"))
tensorboard_callback = TensorBoard(logdir)

# Early Stopper callback
e_stopper = tf.keras.callbacks.EarlyStopping(monitor='val_auroc',
                             min_delta=0,
                             patience=5,
                             verbose=1, mode='max',restore_best_weights=True)

adam=tf.keras.optimizers.Adam(learning_rate=0.001)
Model1.compile(optimizer=adam, loss='binary_crossentropy', metrics=[auroc])
Model1.fit(x=train, y=y_train
           ,
           batch_size=512,
           epochs=150,
           validation_data=(cv, y_cv),
           callbacks=[check_point,tensorboard_callback,e_stopper],
           verbose=2,
           )
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` t
o specify the frequency in number of samples seen.
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Train on 69918 samples, validate on 21850 samples
Epoch 1/150
69918/69918 - 41s - loss: 0.5123 - auroc: 0.5433 - val_loss: 0.4207 - val_au
roc: 0.6446
Epoch 2/150
69918/69918 - 40s - loss: 0.4089 - auroc: 0.6537 - val_loss: 0.3991 - val_au
roc: 0.7131
Epoch 3/150
69918/69918 - 40s - loss: 0.3874 - auroc: 0.7155 - val_loss: 0.3858 - val_au
roc: 0.7398
Epoch 4/150
69918/69918 - 40s - loss: 0.3853 - auroc: 0.7225 - val_loss: 0.3714 - val_au
roc: 0.7487
Epoch 5/150
69918/69918 - 40s - loss: 0.3769 - auroc: 0.7373 - val_loss: 0.3682 - val_au
roc: 0.7547
Epoch 6/150
69918/69918 - 40s - loss: 0.3738 - auroc: 0.7444 - val_loss: 0.3660 - val_au
roc: 0.7583
Epoch 7/150
69918/69918 - 40s - loss: 0.3686 - auroc: 0.7539 - val_loss: 0.3685 - val_au
roc: 0.7484
Epoch 8/150
69918/69918 - 40s - loss: 0.3701 - auroc: 0.7527 - val_loss: 0.3720 - val_au
roc: 0.7488
Epoch 9/150
69918/69918 - 40s - loss: 0.3705 - auroc: 0.7521 - val_loss: 0.3696 - val_au
roc: 0.7685
Epoch 10/150
69918/69918 - 40s - loss: 0.3630 - auroc: 0.7665 - val_loss: 0.4154 - val_au
roc: 0.6601
Epoch 11/150
69918/69918 - 40s - loss: 0.3624 - auroc: 0.7645 - val_loss: 0.3654 - val_au
roc: 0.7672
Epoch 12/150
69918/69918 - 40s - loss: 0.3699 - auroc: 0.7517 - val_loss: 0.3657 - val_au
roc: 0.7524
Epoch 13/150
69918/69918 - 40s - loss: 0.3618 - auroc: 0.7664 - val_loss: 0.3660 - val_au
roc: 0.7685
Epoch 14/150
Restoring model weights from the end of the best epoch.
69918/69918 - 40s - loss: 0.3572 - auroc: 0.7748 - val_loss: 0.3772 - val_au
roc: 0.7682
Epoch 00014: early stopping
```

Out[131]:
```
<tensorflow.python.keras.callbacks.History at 0x2beb1940fc8>
```

1.9 Saving the weights and Evaluation

In [65]:
```
# refer: https://machinelearningmastery.com/save-load-keras-deep-learning-mode
```

```
ls/

# Saving the model weights
Model1.save_weights("model.h5")
print("Saved model to disk")

# Loading Model and weights
Model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[aur
oc])
Model1.load_weights("model.h5")
print("Loaded model from disk")
```

```
Saved model to disk
Loaded model from disk
```

Evalution using keras custom roc auc

In [70]:

```
# evaluate loaded model on test data
train_score = Model1.evaluate(train,y_train, verbose=2,batch_size=(128))
print(f"train {Model1.metrics_names[1]}:  {train_score[1]}\n")

cv_score = Model1.evaluate(cv,y_cv, verbose=2,batch_size=(128))
print(f"cv {Model1.metrics_names[1]}:  {cv_score[1]}\n")

test_score = Model1.evaluate(test,y_test, verbose=2,batch_size=(128))
print(f"test {Model1.metrics_names[1]}:  {test_score[1]}\n")
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
69918/69918 - 20s - loss: 0.3400 - auroc: 0.8068
train auroc:  0.8068248033523560

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
21850/21850 - 6s - loss: 0.3604 - auroc: 0.7674
cv auroc:  0.7673921585083008

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
17480/17480 - 5s - loss: 0.3680 - auroc: 0.7582
test auroc:  0.7582385540008545
```

Evalution using direct sklearn roc auc

In [73]:

```
# Function to calculate Sklearn ROC AUc
def sklearn_roc(actual_values,predicted_values):

    y_pred =(np.argmax(np.array(predicted_values), axis=1))
    y_true = (np.argmax(np.array(actual_values), axis=1))
    return roc_auc_score(y_pred,y_true)
```

```
# sklearn ROC AUC
train_score = Model1.predict(train)
print(f"Train {Model1.metrics_names[1]}:  {sklearn_roc(y_train,train_score)}\n
")

cv_score = Model1.predict(cv)
print(f"Cv {Model1.metrics_names[1]}:  {sklearn_roc(y_cv,cv_score)}\n")

test_score = Model1.predict(test)
print(f"test {Model1.metrics_names[1]}:  {sklearn_roc(y_test,test_score)}\n")
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Train auroc:  0.7802033800143382

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Cv auroc:  0.75785627224937

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
test auroc:  0.7507350471186969
```

2. Training Using 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 s
ome analysis on the Idf values and based on those values choose the low
 and high threshold value. Because very frequent words and very very rar
e words don't give much information. (you can plot a box plots and take
 only the idf scores within IQR range and corresponding words)

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

2.1 Training TFIDF on train essay data

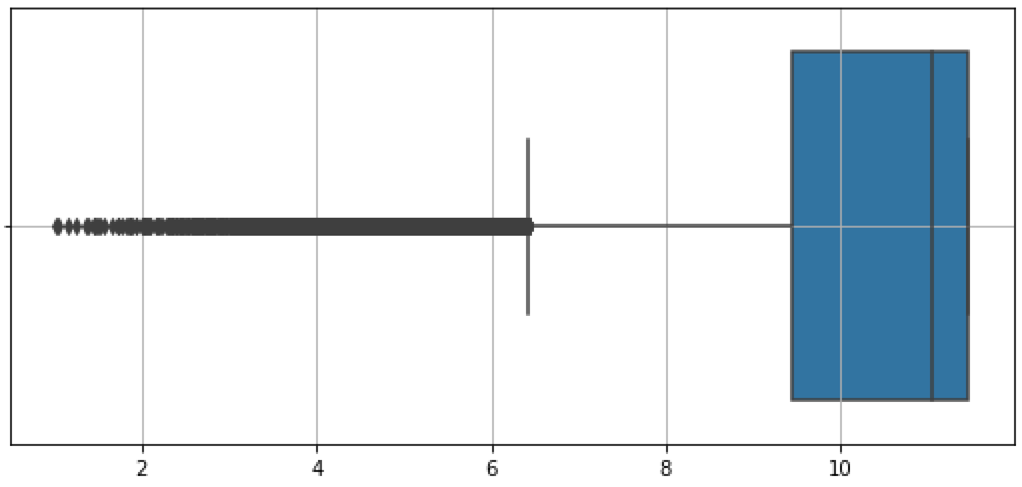
```
In [77]: from sklearn.feature_extraction.text import TfidfVectorizer

         tf_vect = TfidfVectorizer()
         tf_vect.fit_transform(X_train.essay)
```

```
Out[77]: <69918x47405 sparse matrix of type '<class 'numpy.float64'>'
                 with 7549633 stored elements in Compressed Sparse Row format>
```

2.2 BOX plot of IDF scores

```
In [78]: # IQR plot
         plt.figure(figsize=(9,4))
         sns.boxplot((tf_vect.idf_))
         plt.grid()
         plt.title("IQR plot of IDF score generated by TFIDF\n")
         plt.show()
```

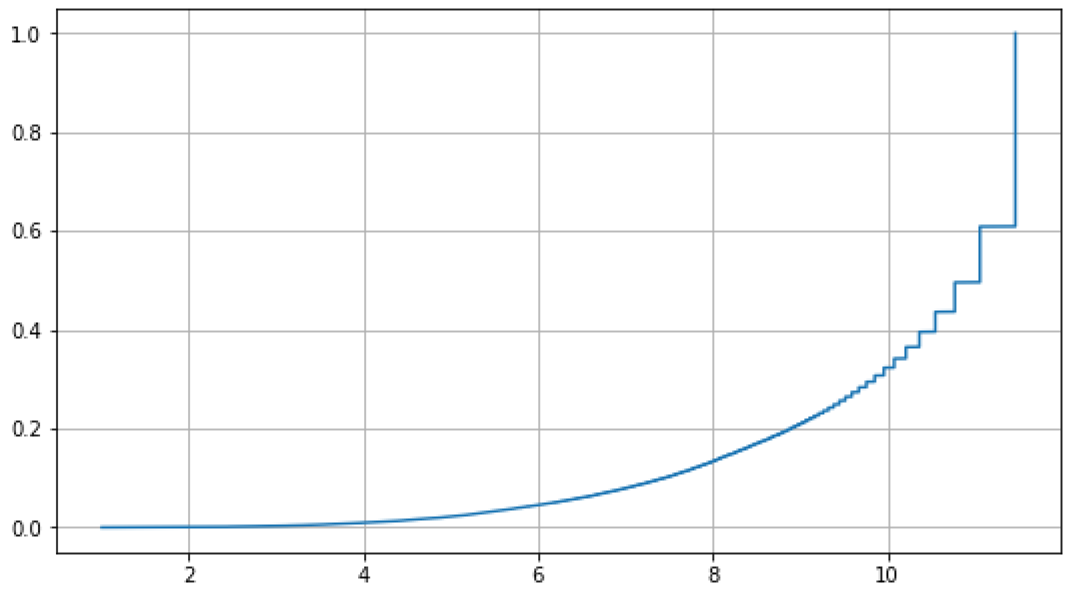
IQR plot of IDF score generated by TFIDF



```
In [79]: # 0th , 25th , 50th , 75th , 100th pecentile
         for i in range(0,101,25):
             print(f"{i}th percentile: {np.percentile(tf_vect.idf_,i)}")
```

```
0th percentile: 1.0077820287056232
25th percentile: 9.447042507065435
50th percentile: 11.056480419499536
75th percentile: 11.4619455276077
100th percentile: 11.4619455276077
```

```
In [80]: # CDF OF IQR
         plt.figure(figsize=(9,5))
         plt.plot(sorted(tf_vect.idf_),(np.arange(len(tf_vect.idf_))/len(tf_vect.idf_
         )),)
         plt.grid()
         plt.show()
```



2.3 Removing words with low and high IDF score and choosing vocabulary of IQR range of IDF scorw

In [81]:
```
# Getting index of vocabulary of IQR range IDF
top_vocab_idx = np.where(tf_vect.idf_>9.44) and  np.where(tf_vect.idf_<11.46)

# Selection of vocabulary
tfidf_vocab = np.array(tf_vect.get_feature_names())

# Final selected vocabulary
selected_vocab = tfidf_vocab[top_vocab_idx]

print("Length of vocab Before removing words: ",(vocab_size_essay))
print("\nLength of vocab after removing words: ",len(selected_vocab))
```

```
Length of vocab Before removing words:  47442

Length of vocab after removing words:  28855
```

2.4.preparing essay feature using optimal tf_idf scores

In [82]:
```
# refer: https://machinelearningmastery.com/use-word-embedding-layers-deep-lea
rning-keras/

# define documents
train_essay = X_train.essay
cv_essay = X_cv.essay
test_essay = X_test.essay

# prepare tokenizer
t = Tokenizer()
t.fit_on_texts(selected_vocab)    ## Using selected words in vocubuary based on
IDF score generated by TFIDF vectoriser
vocab_size_essay_selected_idf = len(t.word_index) + 1
# integer encode the documents
train_encoded_essay_selected_idf = t.texts_to_sequences(train_essay)
cv_encoded_essay_selected_idf = t.texts_to_sequences(cv_essay)
test_encoded_essay_selected_idf = t.texts_to_sequences(test_essay)
print(train_encoded_essay_selected_idf)
```

```
# pad documents to a max length of words :339 in our case
max_length = max_len_of_sent
train_padded_essay_selected_idf = pad_sequences(train_encoded_essay_selected_i
df, maxlen=max_length, padding='post')
cv_padded_essay_selected_idf = pad_sequences(cv_encoded_essay_selected_idf, ma
xlen=max_length, padding='post')
test_padded_essay_selected_idf = pad_sequences(test_encoded_essay_selected_idf
, maxlen=max_length, padding='post')
print(train_padded_essay_selected_idf)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
[[ 8339  6837 15135 ...     0     0     0]
 [ 2124 24805 10180 ...     0     0     0]
 [17023 24805 26643 ...     0     0     0]
 ...
 [ 1350 25553 15297 ...     0     0     0]
 [17023 14432 27894 ...     0     0     0]
 [19341 28530 18173 ...     0     0     0]]
```

In [83]:
```
"""# load the whole embedding into memory
embeddings_index = dict()
f = open('glove.42B.300d.txt',encoding='utf-8')
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))
# create a weight matrix for words in training docs
new_embedding_matrix = zeros((vocab_size_essay_selected_idf, 300))
for word, i in t.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        new_embedding_matrix[i] = embedding_vector"""
```

Out[83]:
```
"# load the whole embedding into memory\nembeddings_index = dict()\nf = open
('glove.42B.300d.txt',encoding='utf-8')\nfor line in f:\n    values = line.s
plit()\n    word = values[0]\n    coefs = asarray(values[1:], dtype='float3
2')\n    embeddings_index[word] = coefs\nf.close()\n\n\nprint('Loaded %s wor
d vectors.' % len(embeddings_index))\n# create a weight matrix for words in
training docs\nnew_embedding_matrix = zeros((vocab_size_essay_selected_idf,
300))\nfor word, i in t.word_index.items():\n    embedding_vector = embeddin
gs_index.get(word)\n    if embedding_vector is not None:\n        new_embedd
ing_matrix[i] = embedding_vector"
```

In [84]:
```
# Saving embedding matrix
```

```
"""file = open("new_embedding_matrix",'wb')
np.save(file,new_embedding_matrix)
file.close()"""

with open("new_embedding_matrix",'rb') as f:
    new_embedding_matrix=np.load(f)
```

2.5 Model-2 (same architecture as Model-1)

In [90]:
```
#initialisation of vocab size and shapes of all input features

# Vocab size of each feature for Embeding Layer
vocab_size_essay_idf = len(selected_vocab)+1  # adding 1 for zero padding inde
x
category_vocab_size = category_vocab_size
sub_category_vocab_size = sub_category_vocab_size
train_school_state_label_encoded_vocab_size = len(np.unique(train_school_state
_label_encoded))
train_teacher_prefix_label_encoded_vocab_size = len(np.unique(train_teacher_pr
efix_label_encoded))
train_grade_label_encoded_vocab_size = len(np.unique(train_grade_label_encoded
))

# shapes of input of each feaures for Input Layer
train_padded_essay_shape_idf = train_padded_essay_selected_idf.shape
train_school_state_label_encoded_shape = train_school_state_label_encoded.shap
e
train_teacher_prefix_label_encoded_shape = train_teacher_prefix_label_encoded.
shape
train_grade_label_encoded_shape = train_grade_label_encoded.shape
train_padded_clean_subcategories_shape = train_padded_clean_subcategories.shap
e
train_padded_clean_categories_shape = train_padded_clean_categories.shape
train_price_shape = X_train['price'].shape
train_teacher_number_of_previously_posted_projects_shape = X_train["teacher_nu
mber_of_previously_posted_projects"].shape
```

In [91]:
```
# Model 2 using Keras Functional Api

tf.keras.backend.clear_session()

# Eaasy
input1= Input(shape=(train_padded_essay_shape_idf[1],) ,name='essay_input' )

# Essay embed layer using pretrained Glove embedings of 300 dim
Embeded_layer1 = Embedding(vocab_size_essay_idf, output_dim=300, weights=[new_
embedding_matrix], input_length=339, trainable=False)(input1)
# Lstm layer for essay
lstm_layer1 = layers.LSTM(units=64,return_sequences=True)(Embeded_layer1)
flatten1 = Flatten()(lstm_layer1)

# school state
input2= Input(shape=(train_school_state_label_encoded_shape[1],) ,name='school
_state_input')
Embeded_layer2= Embedding(train_school_state_label_encoded_vocab_size, output_
dim=50, input_length=1)(input2)
flatten2 = Flatten()(Embeded_layer2)

# teacher_prefix
```

```
input3= Input(shape=(train_teacher_prefix_label_encoded_shape[1],) ,name='teac
her_prefix')
Embeded_layer3= Embedding(train_school_state_label_encoded_vocab_size, output_
dim=50, input_length=1)(input3)
flatten3 = Flatten()(Embeded_layer3)

# project_grade_category
input4= Input(shape=(train_grade_label_encoded_shape[1],) ,name='project_grade
_category')
Embeded_layer4 = Embedding(train_grade_label_encoded_vocab_size, output_dim=50
, input_length=1)(input4)
flatten4 = Flatten()(Embeded_layer4)

# clean_subcategories
input5= Input(shape=(train_padded_clean_categories_shape[1],) ,name='clean_cat
egories')
Embeded_layer5 = Embedding(category_vocab_size, output_dim=50, input_length=66
4)(input5)
flatten5 = Flatten()(Embeded_layer5)

# clean_subcategories
input6= Input(shape=(train_padded_clean_subcategories_shape[1],) ,name='clean_
subcategories')
Embeded_layer6 = Embedding(sub_category_vocab_size, output_dim=50, input_lengt
h=3)(input6)
flatten6 = Flatten()(Embeded_layer6)

input7 = Input(shape=(1,) ,name='price')
input8 = Input(shape=(1,) ,name='teacher_number_of_previously_posted_projects'
)
input9 = Input(shape=(1,) ,name='quantity')
input10 = Input(shape=(1,) ,name='presence_of_num')

concat_input_7_8_9_10 = concatenate([input7,input8,input9,input10], name='conc
at_input7_input8')
dense1 = Dense(32,activation='relu',kernel_initializer='he_uniform')(concat_in
put_7_8_9_10)

# Concatenation of all output of previous layers
concatenate2 = concatenate([flatten1,flatten2,flatten3,flatten4,flatten5,dense
1], name="concat_all_output")

# Dense layer 2 after concatenation of all output
dense_layer2 = Dense(units=128 ,activation='relu' ,kernel_initializer='he_unif
orm')(concatenate2)
dropout1=Dropout(rate=0.2)(dense_layer2)
batch_normalize1 = layers.BatchNormalization()(dropout1)


# Dense layer 3 after concatenation of all output
dense_layer3 = Dense(units=128 ,activation='relu' ,kernel_initializer='he_unif
orm')(batch_normalize1)
dropout2 = Dropout(rate=0.2)(dense_layer3)
batch_normalize2=layers.BatchNormalization()(dropout2)


# Dense layer 3 after concatenation of all output
dense_layer4 = Dense(units=64 ,activation='relu' ,kernel_initializer='he_unifo
rm')(batch_normalize2)
dropout2 = Dropout(rate=0.2)(dense_layer4)

#output
```

```
output = Dense(units=2 ,activation='softmax' ,kernel_initializer='glorot_unifo
rm', name="output_layer")(dropout2)

Model2 = Model(inputs=[input1,input2,input3,input4,input5,input6,input7,input8
,input9,input10], outputs=output)
print(Model2.summary())
```

```
Model: "model"
__________________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==================================================================================
========================
essay_input (InputLayer)        [(None, 331)]        0
__________________________________________________________________________________
________________________
embedding (Embedding)           (None, 331, 300)     8656800     essay_inp
ut[0][0]
__________________________________________________________________________________
________________________
school_state_input (InputLayer) [(None, 1)]          0
__________________________________________________________________________________
________________________
teacher_prefix (InputLayer)     [(None, 1)]          0
__________________________________________________________________________________
________________________
project_grade_category (InputLa [(None, 1)]          0
__________________________________________________________________________________
________________________
clean_categories (InputLayer)   [(None, 3)]          0
__________________________________________________________________________________
________________________
price (InputLayer)              [(None, 1)]          0
__________________________________________________________________________________
________________________
teacher_number_of_previously_po [(None, 1)]          0
__________________________________________________________________________________
________________________
quantity (InputLayer)           [(None, 1)]          0
__________________________________________________________________________________
________________________
presence_of_num (InputLayer)    [(None, 1)]          0
__________________________________________________________________________________
________________________
lstm (LSTM)                     (None, 331, 64)      93440       embedding
[0][0]
__________________________________________________________________________________
________________________
embedding_1 (Embedding)         (None, 1, 50)        2550        school_st
ate_input[0][0]
__________________________________________________________________________________
________________________
embedding_2 (Embedding)         (None, 1, 50)        2550        teacher_p
refix[0][0]
__________________________________________________________________________________
________________________
embedding_3 (Embedding)         (None, 1, 50)        200         project_g
rade_category[0][0]
__________________________________________________________________________________
________________________
embedding_4 (Embedding)         (None, 3, 50)        500         clean_cat
```

```
egories[0][0]
__________________________________________________________________________________________
________________________
concat_input7_input8 (Concatena (None, 4)            0           price[0]
[0]
                                                                 teacher_n
umber_of_previously_post
                                                                 quantity
[0][0]
                                                                 presence_
of_num[0][0]
__________________________________________________________________________________________
________________________
flatten (Flatten)               (None, 21184)        0           lstm[0]
[0]
__________________________________________________________________________________________
________________________
flatten_1 (Flatten)             (None, 50)           0           embedding
_1[0][0]
__________________________________________________________________________________________
________________________
flatten_2 (Flatten)             (None, 50)           0           embedding
_2[0][0]
__________________________________________________________________________________________
________________________
flatten_3 (Flatten)             (None, 50)           0           embedding
_3[0][0]
__________________________________________________________________________________________
________________________
flatten_4 (Flatten)             (None, 150)          0           embedding
_4[0][0]
__________________________________________________________________________________________
________________________
dense (Dense)                   (None, 32)           160         concat_in
put7_input8[0][0]
__________________________________________________________________________________________
________________________
concat_all_output (Concatenate) (None, 21516)        0           flatten
[0][0]
                                                                 flatten_1
[0][0]
                                                                 flatten_2
[0][0]
                                                                 flatten_3
[0][0]
                                                                 flatten_4
[0][0]
                                                                 dense[0]
[0]
__________________________________________________________________________________________
________________________
dense_1 (Dense)                 (None, 128)          2754176     concat_al
l_output[0][0]
__________________________________________________________________________________________
________________________
dropout (Dropout)               (None, 128)          0           dense_1
[0][0]
__________________________________________________________________________________________
________________________
batch_normalization (BatchNorma (None, 128)          512         dropout
[0][0]
__________________________________________________________________________________________
```

```
________________________________
dense_2 (Dense)                 (None, 128)          16512       batch_nor
malization[0][0]
__________________________________________________________________________________________________
________________________________
dropout_1 (Dropout)             (None, 128)          0           dense_2
[0][0]
__________________________________________________________________________________________________
________________________________
batch_normalization_1 (BatchNor (None, 128)          512         dropout_1
[0][0]
__________________________________________________________________________________________________
________________________________
dense_3 (Dense)                 (None, 64)           8256        batch_nor
malization_1[0][0]
__________________________________________________________________________________________________
________________________________
dropout_2 (Dropout)             (None, 64)           0           dense_3
[0][0]
__________________________________________________________________________________________________
________________________________
clean_subcategories (InputLayer [(None, 3)]          0
__________________________________________________________________________________________________
________________________________
output_layer (Dense)            (None, 2)            130         dropout_2
[0][0]
==================================================================================================
================================
Total params: 11,536,298
Trainable params: 2,878,986
Non-trainable params: 8,657,312
__________________________________________________________________________________________________
________________________________
None
```

In [92]:
```
train=[train_padded_essay_selected_idf,train_school_state_label_encoded,train_teacher_prefix_label_encoded,train_grade_label_encoded,train_padded_clean_categories,
       train_padded_clean_subcategories,X_train["price"],X_train["teacher_number_of_previously_posted_projects"],X_train['quantity'],
       X_train['presence_of_number_in_description']]

cv = [cv_padded_essay_selected_idf,cv_school_state_label_encoded,cv_teacher_prefix_label_encoded,cv_grade_label_encoded,cv_padded_clean_categories,
      cv_padded_clean_subcategories,X_cv["price"],X_cv["teacher_number_of_previously_posted_projects"],X_cv['quantity'],
      X_cv['presence_of_number_in_description']]

test = [test_padded_essay_selected_idf,test_school_state_label_encoded,test_teacher_prefix_label_encoded,test_grade_label_encoded,test_padded_clean_categories,
        test_padded_clean_subcategories,X_test["price"],X_test["teacher_number_of_previously_posted_projects"],X_test['quantity'],
        X_test['presence_of_number_in_description']]
```

2.6 Traning

In [93]:
```
# Traning using Model-2
```

```python
# defining custom ROC AUC for keras ,refer: https://stackoverflow.com/a/517349
92
from sklearn.metrics import roc_auc_score
def auroc(y_true, y_pred):
    return tf.compat.v1.py_func(roc_auc_score, (y_true, y_pred), tf.double)

# Checkpoint callbacks
check_point = ModelCheckpoint(
    filepath='weights.{epoch:02d}-{val_loss:.2f}.hdf5',
    monitor='val_auroc',
    verbose=1,
    save_best_only=True,
    save_weights_only=True,
    mode='max',
    period=10)

# Load the TensorBoard notebook extension
logs_base_dir = ".\logs"
os.makedirs(logs_base_dir, exist_ok=True)
logdir = os.path.join(logs_base_dir, datetime.datetime.now().strftime("%Y%m%d-
%H%M%S"))
tensorboard_callback = TensorBoard(logdir)

# Early Stopper callback
e_stopper = tf.keras.callbacks.EarlyStopping(monitor='val_auroc',
                              min_delta=0,
                              patience=10,
                              verbose=1, mode='max',restore_best_weights=True)

Model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[aur
oc])
Model2.fit(x=train, y=y_train,
           batch_size=128,
           epochs=150,
           validation_data=(cv, y_cv),
           callbacks=[check_point,tensorboard_callback,e_stopper],
           verbose=2,
           )
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` t
o specify the frequency in number of samples seen.
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Train on 69918 samples, validate on 21850 samples
Epoch 1/150
69918/69918 - 29s - loss: 0.4489 - auroc: 0.6156 - val_loss: 0.3945 - val_au
roc: 0.7125
Epoch 2/150
69918/69918 - 28s - loss: 0.3852 - auroc: 0.7236 - val_loss: 0.3788 - val_au
roc: 0.7520
Epoch 3/150
69918/69918 - 28s - loss: 0.3709 - auroc: 0.7493 - val_loss: 0.3652 - val_au
roc: 0.7576
Epoch 4/150
69918/69918 - 28s - loss: 0.3637 - auroc: 0.7619 - val_loss: 0.3660 - val_au
roc: 0.7526
Epoch 5/150
69918/69918 - 28s - loss: 0.3580 - auroc: 0.7725 - val_loss: 0.3583 - val_au
roc: 0.7666
```

```
Epoch 6/150
69918/69918 - 28s - loss: 0.3509 - auroc: 0.7846 - val_loss: 0.3643 - val_au
roc: 0.7704
Epoch 7/150
69918/69918 - 28s - loss: 0.3405 - auroc: 0.8012 - val_loss: 0.3626 - val_au
roc: 0.7607
Epoch 8/150
69918/69918 - 28s - loss: 0.3295 - auroc: 0.8173 - val_loss: 0.3729 - val_au
roc: 0.7560
Epoch 9/150
69918/69918 - 28s - loss: 0.3121 - auroc: 0.8388 - val_loss: 0.3744 - val_au
roc: 0.7464
Epoch 10/150

Epoch 00010: val_auroc improved from -inf to 0.73852, saving model to weight
s.10-0.41.hdf5
69918/69918 - 28s - loss: 0.2875 - auroc: 0.8674 - val_loss: 0.4132 - val_au
roc: 0.7385
Epoch 11/150
69918/69918 - 28s - loss: 0.2570 - auroc: 0.8978 - val_loss: 0.4553 - val_au
roc: 0.7157
Epoch 12/150
69918/69918 - 28s - loss: 0.2240 - auroc: 0.9230 - val_loss: 0.4668 - val_au
roc: 0.7178
Epoch 13/150
69918/69918 - 28s - loss: 0.1894 - auroc: 0.9461 - val_loss: 0.5379 - val_au
roc: 0.7097
Epoch 14/150
69918/69918 - 28s - loss: 0.1587 - auroc: 0.9636 - val_loss: 0.5945 - val_au
roc: 0.6796
Epoch 15/150
69918/69918 - 28s - loss: 0.1321 - auroc: 0.9758 - val_loss: 0.6302 - val_au
roc: 0.6949
Epoch 16/150
Restoring model weights from the end of the best epoch.
69918/69918 - 28s - loss: 0.1090 - auroc: 0.9836 - val_loss: 0.7054 - val_au
roc: 0.6959
Epoch 00016: early stopping
```

Out[93]:
```
<tensorflow.python.keras.callbacks.History at 0x2bea0e9f048>
```

2.7 Saving the weights and Evaluation

In [94]:
```
# refer: https://machinelearningmastery.com/save-load-keras-deep-learning-mode
ls/

# Saving the model weights
Model2.save_weights("model2.h5")
print("Saved model to disk")

# Loading Model and weights
Model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[aur
oc])
Model2.load_weights("model2.h5")
print("Loaded model from disk")
```

```
Saved model to disk
Loaded model from disk
```

Evalution using keras custom roc auc

In [95]:
```
# evaluate loaded model on test data
train_score = Model2.evaluate(train,y_train, verbose=2,batch_size=(128))
print(f"train {Model2.metrics_names[1]}:  {train_score[1]}\n")

cv_score = Model2.evaluate(cv,y_cv, verbose=2,batch_size=(128))
print(f"cv {Model2.metrics_names[1]}:  {cv_score[1]}\n")

test_score = Model2.evaluate(test,y_test, verbose=2,batch_size=(128))
print(f"test {Model2.metrics_names[1]}:  {test_score[1]}\n")
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
69918/69918 - 17s - loss: 0.3425 - auroc: 0.8100
train auroc:  0.8099930882453918

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
21850/21850 - 5s - loss: 0.3643 - auroc: 0.7715
cv auroc:  0.7715215682983398

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
17480/17480 - 4s - loss: 0.3690 - auroc: 0.7659
test auroc:  0.7659369707107544
```

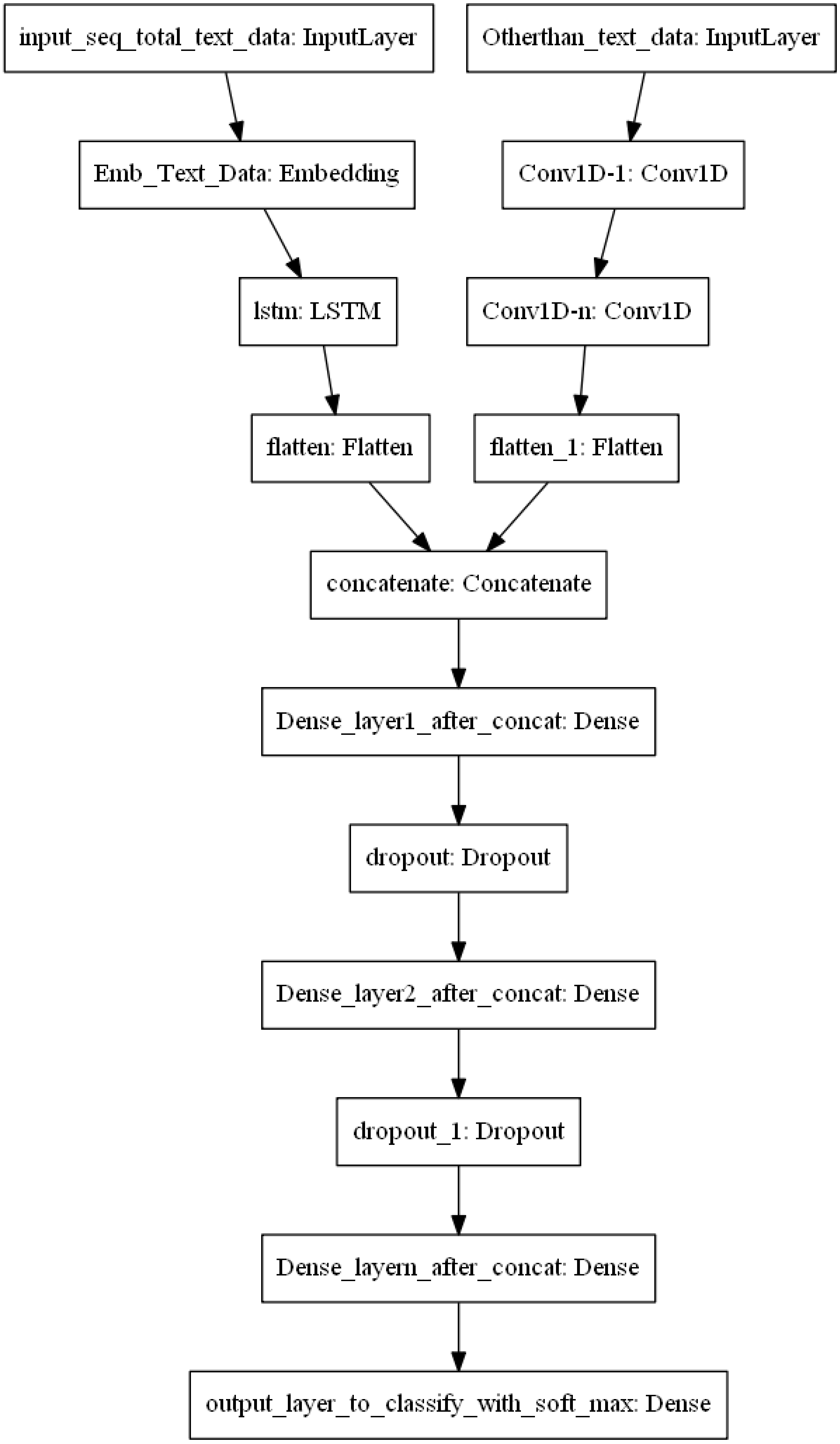
Evalution using direct sklearn roc auc

In [96]:
```
# sklearn ROC AUC
train_score = Model2.predict(train)
print(f"Train {Model2.metrics_names[1]}:  {sklearn_roc(y_train,train_score)}\n
")

cv_score = Model2.predict(cv)
print(f"Cv {Model2.metrics_names[1]}:  {sklearn_roc(y_cv,cv_score)}\n")

test_score = Model2.predict(test)
print(f"test {Model2.metrics_names[1]}:  {sklearn_roc(y_test,test_score)}\n")
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Train auroc:  0.7405537303896463

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Cv auroc:  0.7023508486906042


WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
```

```
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
test auroc:  0.7015559231611674
```

3. Model-3


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

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

- **input_seq_total_text_data**:

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

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

  . Use LSTM that is given above, get the LSTM output and Flatten th
at 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 con
catenate 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, str
ide.
```

</pre>

3.0 For Essay feature using the same embedding which is used in Model-2

3.1. Preparing school_state data for training

In [97]:

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(sparse=False)
train_school_state_label_encoded_oh = ohe.fit_transform(np.array(X_train.schoo
l_state).reshape(-1,1))
cv_school_state_label_encoded_oh = ohe.transform(np.array(X_cv.school_state).r
eshape(-1,1))
test_school_state_label_encoded_oh = ohe.transform(np.array(X_test.school_stat
e).reshape(-1,1))

print("number of Unique categories in School State feature :",len(np.unique(oh
e.categories_)))
print(test_school_state_label_encoded_oh)
```

```
number of Unique categories in School State feature : 51
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

3.2 Preparing teacher_prefix data for training

In [98]:
```
ohe = OneHotEncoder(sparse=False)
train_teacher_prefix_label_encoded_oh = ohe.fit_transform(np.array(X_train.tea
cher_prefix).reshape(-1,1))
cv_teacher_prefix_label_encoded_oh = ohe.transform(np.array(X_cv.teacher_prefi
x).reshape(-1,1))
test_teacher_prefix_label_encoded_oh = ohe.transform(np.array(X_test.teacher_p
refix).reshape(-1,1))

print("number of Unique categories in teacher_prefix feature :",len(np.unique(
ohe.categories_)))
print(f"\nbefore: \n{X_test.teacher_prefix.iloc[:5]}\n")
print(f"after: \n{test_teacher_prefix_label_encoded_oh[:5]}")
```

```
number of Unique categories in teacher_prefix feature : 5

before:
73624      ms
2158      mrs
74759     mrs
89804     mrs
18127     mrs
Name: teacher_prefix, dtype: object

after:
[[0. 0. 0. 1. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]]
```

3.3 Preparing project_grade_category data for training

In [99]:
```
ohe = OneHotEncoder(sparse=False)
train_grade_label_encoded_oh = ohe.fit_transform(np.array(X_train.project_grad
e_category).reshape(-1,1))
cv_grade_label_encoded_oh = ohe.transform(np.array(X_cv.project_grade_category
).reshape(-1,1))
test_grade_label_encoded_oh = ohe.transform(np.array(X_test.project_grade_cate
gory).reshape(-1,1))

print("number of Unique categories in project_grade_category feature :",len(np
.unique(ohe.categories_)))
print(f"\nbefore: \n{X_test.project_grade_category.iloc[:5]}\n")
print(f"after: \n{test_grade_label_encoded_oh[:5]}")
```

```
number of Unique categories in project_grade_category feature : 4

before:
73624    grades_prek_2
2158     grades_prek_2
74759    grades_prek_2
89804       grades_3_5
18127    grades_prek_2
Name: project_grade_category, dtype: object

after:
[[0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]]
```

3.4. Preparing clean_categories data for training

In [100]:

```
ohe = OneHotEncoder(sparse=False)
train_encoded_clean_categories_oh = ohe.fit_transform(np.array(X_train.clean_c
ategories).reshape(-1,1))
cv_encoded_clean_categories_oh = ohe.transform(np.array(X_cv.clean_categories)
.reshape(-1,1))
test_encoded_clean_categories_oh = ohe.transform(np.array(X_test.clean_categor
ies).reshape(-1,1))

print("number of Unique clean_categories in teacher_prefix feature :",len(np.u
nique(ohe.categories_)))
print(f"\nbefore: \n{X_test.clean_categories.iloc[:5]}\n")
print(f"after: \n{test_encoded_clean_categories_oh[:5]}")
```

```
number of Unique clean_categories in teacher_prefix feature : 51

before:
73624                  literacy_language
2158      literacy_language math_science
74759                  literacy_language
89804                  literacy_language
18127                    appliedlearning
Name: clean_categories, dtype: object

after:
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]]
```

3.5. Preparing clean_subcategories data for training

In [101]:

```
from sklearn.feature_extraction.text import CountVectorizer

count_vect =CountVectorizer()
train_encoded_clean_subcategories_oh = count_vect.fit_transform((X_train.clean
_subcategories)).toarray()
cv_encoded_clean_subcategories_oh = count_vect.transform((X_cv.clean_subcatego
ries)).toarray()
test_encoded_clean_subcategories_oh = count_vect.transform((X_test.clean_subca
tegories)).toarray()
```

```
print("number of Unique categories in clean_subcategories feature :",len(np.un
ique(count_vect.get_feature_names())))
print(f"\nbefore: \n{X_test.clean_subcategories.iloc[:5]}\n")
print(f"after: \n{test_encoded_clean_subcategories_oh[:5]}")
```

```
number of Unique categories in clean_subcategories feature : 30

before:
73624                literacy
2158     literacy mathematics
74759                literacy
89804      literature_writing
18127        earlydevelopment
Name: clean_subcategories, dtype: object

after:
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

3.6 Training using Model-3

In [103]:

```
#initialisation of vocab size and shapes of all input features

# Vocab size of each feature for Embeding Layer
vocab_size_essay_idf = len(selected_vocab)+1  # adding 1 for zero padding inde
x

# shapes of input of each feaures for Input Layer
train_padded_essay_shape = train_padded_essay_selected_idf.shape
train_school_state_label_encoded_shape = train_school_state_label_encoded_oh.s
hape
train_teacher_prefix_label_encoded_shape = train_teacher_prefix_label_encoded_
oh.shape
train_grade_label_encoded_shape = train_grade_label_encoded_oh.shape
train_padded_clean_subcategories_shape = train_encoded_clean_subcategories_oh.
shape
train_padded_clean_categories_shape = train_encoded_clean_categories_oh.shape
train_price_shape = X_train['price'].shape
train_teacher_number_of_previously_posted_projects_shape = X_train["teacher_nu
mber_of_previously_posted_projects"].shape
```

Model 3

In [109]:

```
# Model 3 using Keras Functional Api
tf.keras.backend.clear_session()

# Eaasy
input1= Input(shape=(train_padded_essay_shape[1],) ,name='essay_input' )
# Essay embed layer using pretrained Glove embedings of 300 dim
Embeded_layer1 = Embedding(vocab_size_essay_idf, output_dim=300, weights=[new_
embedding_matrix], input_length=339, trainable=False)(input1)
# Lstm layer for essay
lstm_layer1 = (layers.LSTM(units=64,return_sequences=True,dropout=0.2))(Embede
d_layer1)
flatten1 = Flatten()(lstm_layer1)
```

```
# school state
input2= Input(shape=(train_school_state_label_encoded_shape[1],) ,name='school
_state_input')

# teacher_prefix
input3= Input(shape=(train_teacher_prefix_label_encoded_shape[1],) ,name='teac
her_prefix')

# project_grade_category
input4= Input(shape=(train_grade_label_encoded_shape[1],) ,name='project_grade
_category')

# clean_subcategories
input5= Input(shape=(train_padded_clean_categories_shape[1],) ,name='clean_cat
egories')

# clean_subcategories
input6= Input(shape=(train_padded_clean_subcategories_shape[1],) ,name='clean_
subcategories')

# price
input7 = Input(shape=(1,) ,name='price')

# teacher_number_of_previously_posted_projects
input8 = Input(shape=(1,) ,name='teacher_number_of_previously_posted_projects'
)

# mean quantity
input9 = Input(shape=(1,) ,name='quantity')

# presence_of_num
input10 = Input(shape=(1,) ,name='presence_of_num')

# Concatenation of all inputs ecxept essay feature
concat_other_than_essay = layers.concatenate([input2,input3,input4,input5,inpu
t6,input7,input8,input9,input10],name='concat_other_than_essay')
print(concat_other_than_essay.shape)
concat_other_than_essay =  layers.Reshape([145,1])(concat_other_than_essay)


# Convolution 1-D on concatenated feature
convolution1=layers.Conv1D(filters=256,
              kernel_size=3,
              strides=1,
              padding='valid',
              activation='relu',
              kernel_initializer='glorot_uniform',name='Convolution_layer_1')(
concat_other_than_essay)
# Maxpool layer 1
maxpool1 = layers.MaxPool1D(pool_size=3)(convolution1)

# Convolution layer 2
convolution2=layers.Convolution1D(filters=128,
              kernel_size=3,
              strides=1,
              padding='valid',
              activation='relu',
              kernel_initializer='glorot_uniform',name='Convolution_layer_2')(
maxpool1)
# Maxpool layer 2
maxpool2 = layers.MaxPool1D(pool_size=3)(convolution2)
```

```
# Flatten the output of final convolution layer
flatten2 = Flatten()(maxpool2)

# Concat fllaten LSTM of essay and flatten Convolution of other_than_essay_fea
ture
Contat2 = layers.concatenate([flatten1,flatten2], name='concat_all_features')

# Train Dense on concatenation of all feat (concat2)
dense1 = Dense(128,activation='relu',kernel_initializer='he_uniform')(Contat2)
dropout1 = Dropout(rate=0.5)(dense1)

# Dense layer 2 after concatenation of all output
dense_layer2 = Dense(units=64 ,activation='relu' ,kernel_initializer='he_unifo
rm')(dropout1)
dropout2 = Dropout(rate=0.5)(dense_layer2)

# Dense layer 3 after concatenation of all output
dense_layer3 = Dense(units=64 ,activation='relu' ,kernel_initializer='he_unifo
rm')(dropout2)

#output
output = Dense(units=2 ,activation='softmax' ,kernel_initializer='glorot_unifo
rm', name="output_layer")(dense_layer3)

Model3 = Model(inputs=[input1,input2,input3,input4,input5,input6,input7,input8
,input9,input10], outputs=output)
print(Model3.summary())
```

```
(None, 145)
Model: "model"
__________________________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==========================================================================================
========================
school_state_input (InputLayer) [(None, 51)]         0
__________________________________________________________________________________________
________________________
teacher_prefix (InputLayer)     [(None, 5)]          0
__________________________________________________________________________________________
________________________
project_grade_category (InputLa [(None, 4)]          0
__________________________________________________________________________________________
________________________
clean_categories (InputLayer)   [(None, 51)]         0
__________________________________________________________________________________________
________________________
clean_subcategories (InputLayer [(None, 30)]         0
__________________________________________________________________________________________
________________________
price (InputLayer)              [(None, 1)]          0
__________________________________________________________________________________________
________________________
teacher_number_of_previously_po [(None, 1)]          0
__________________________________________________________________________________________
________________________
quantity (InputLayer)           [(None, 1)]          0
__________________________________________________________________________________________
________________________
presence_of_num (InputLayer)    [(None, 1)]          0
__________________________________________________________________________________________
```

```
________________________
concat_other_than_essay (Concat (None, 145)          0           school_st
ate_input[0][0]
                                                                 teacher_p
refix[0][0]
                                                                 project_g
rade_category[0][0]
                                                                 clean_cat
egories[0][0]
                                                                 clean_sub
categories[0][0]
                                                                 price[0]
[0]
                                                                 teacher_n
umber_of_previously_post
                                                                 quantity
[0][0]
                                                                 presence_
of_num[0][0]
__________________________________________________________________________
________________________
reshape (Reshape)               (None, 145, 1)       0           concat_ot
her_than_essay[0][0]
__________________________________________________________________________
________________________
Convolution_layer_1 (Conv1D)    (None, 143, 256)     1024        reshape
[0][0]
__________________________________________________________________________
________________________
essay_input (InputLayer)        [(None, 331)]        0
__________________________________________________________________________
________________________
max_pooling1d (MaxPooling1D)    (None, 47, 256)      0           Convoluti
on_layer_1[0][0]
__________________________________________________________________________
________________________
embedding (Embedding)           (None, 331, 300)     8656800     essay_inp
ut[0][0]
__________________________________________________________________________
________________________
Convolution_layer_2 (Conv1D)    (None, 45, 128)      98432       max_pooli
ng1d[0][0]
__________________________________________________________________________
________________________
lstm (LSTM)                     (None, 331, 64)      93440       embedding
[0][0]
__________________________________________________________________________
________________________
max_pooling1d_1 (MaxPooling1D)  (None, 15, 128)      0           Convoluti
on_layer_2[0][0]
__________________________________________________________________________
________________________
flatten (Flatten)               (None, 21184)        0           lstm[0]
[0]
__________________________________________________________________________
________________________
flatten_1 (Flatten)             (None, 1920)         0           max_pooli
ng1d_1[0][0]
__________________________________________________________________________
________________________
concat_all_features (Concatenat (None, 23104)        0           flatten
[0][0]
```

```
                                                                 flatten_1
[0][0]
__________________________________________________________________________
________________________
dense (Dense)                   (None, 128)          2957440     concat_al
l_features[0][0]
__________________________________________________________________________
________________________
dropout (Dropout)               (None, 128)          0           dense[0]
[0]
__________________________________________________________________________
________________________
dense_1 (Dense)                 (None, 64)           8256        dropout
[0][0]
__________________________________________________________________________
________________________
dropout_1 (Dropout)             (None, 64)           0           dense_1
[0][0]
__________________________________________________________________________
________________________
dense_2 (Dense)                 (None, 64)           4160        dropout_1
[0][0]
__________________________________________________________________________
________________________
output_layer (Dense)            (None, 2)            130         dense_2
[0][0]
==========================================================================
========================
Total params: 11,819,682
Trainable params: 3,162,882
Non-trainable params: 8,656,800
__________________________________________________________________________
________________________
None
```

In [110]:

```
train=[train_padded_essay_selected_idf,train_school_state_label_encoded_oh,tra
in_teacher_prefix_label_encoded_oh,train_grade_label_encoded_oh,
       train_encoded_clean_categories_oh,train_encoded_clean_subcategories_oh,
X_train["price"],X_train["teacher_number_of_previously_posted_projects"],
       X_train['quantity'],X_train['presence_of_number_in_description']]

cv = [cv_padded_essay_selected_idf,cv_school_state_label_encoded_oh,cv_teacher
_prefix_label_encoded_oh,cv_grade_label_encoded_oh,
      cv_encoded_clean_categories_oh,cv_encoded_clean_subcategories_oh,X_cv["p
rice"],X_cv["teacher_number_of_previously_posted_projects"],
      X_cv['quantity'],X_cv['presence_of_number_in_description']]

test = [test_padded_essay_selected_idf,test_school_state_label_encoded_oh,test
_teacher_prefix_label_encoded_oh,test_grade_label_encoded_oh,
        test_encoded_clean_categories_oh,test_encoded_clean_subcategories_oh,X
_test["price"],X_test["teacher_number_of_previously_posted_projects"],
        X_test['quantity'],X_test['presence_of_number_in_description']]
```

3.7 Training

In [111]:

```
# Traning using Model-3

# defining custom ROC AUC for keras ,refer: https://stackoverflow.com/a/517349
```

```python
92
from sklearn.metrics import roc_auc_score
def auroc(y_true, y_pred):
    return tf.compat.v1.py_func(roc_auc_score, (y_true, y_pred), tf.double)

# Checkpoint callbacks
check_point = ModelCheckpoint(
    filepath='weights.{epoch:02d}-{val_loss:.2f}.hdf5',
    monitor='val_auroc',
    verbose=1,
    save_best_only=True,
    save_weights_only=True,
    mode='max',
    period=10)

# TensorBoard
logs_base_dir = ".\logs"
os.makedirs(logs_base_dir, exist_ok=True)
logdir = os.path.join(logs_base_dir, datetime.datetime.now().strftime("%Y%m%d-
%H%M%S"))
tensorboard_callback = TensorBoard(logdir)

# Early Stopper callback
e_stopper = tf.keras.callbacks.EarlyStopping(monitor='val_auroc',
                              min_delta=0,
                              patience=10,
                              verbose=1, mode='max',restore_best_weights=True
,)

Model3.compile(optimizer='adam', loss='binary_crossentropy', metrics=[auroc])
Model3.fit(x=train, y=y_train,
           batch_size=256,
           epochs=150,
           validation_data=(cv, y_cv),
           callbacks=[check_point, tensorboard_callback, e_stopper],
           verbose=2,
           )
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` t
o specify the frequency in number of samples seen.
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Train on 69918 samples, validate on 21850 samples
Epoch 1/150
WARNING:tensorflow:Trace already enabled
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (1.487852). Check your callbacks.
69918/69918 - 25s - loss: 0.4373 - auroc: 0.6435 - val_loss: 0.3935 - val_au
roc: 0.7289
Epoch 2/150
69918/69918 - 21s - loss: 0.3888 - auroc: 0.7232 - val_loss: 0.4031 - val_au
roc: 0.7430
Epoch 3/150
69918/69918 - 21s - loss: 0.3729 - auroc: 0.7509 - val_loss: 0.3968 - val_au
roc: 0.7491
Epoch 4/150
69918/69918 - 21s - loss: 0.3616 - auroc: 0.7743 - val_loss: 0.3770 - val_au
roc: 0.7518
Epoch 5/150
69918/69918 - 21s - loss: 0.3441 - auroc: 0.8003 - val_loss: 0.3793 - val_au
```

```
roc: 0.7477
Epoch 6/150
69918/69918 - 21s - loss: 0.3228 - auroc: 0.8295 - val_loss: 0.3823 - val_au
roc: 0.7371
Epoch 7/150
69918/69918 - 21s - loss: 0.2920 - auroc: 0.8645 - val_loss: 0.3819 - val_au
roc: 0.7306
Epoch 8/150
69918/69918 - 21s - loss: 0.2642 - auroc: 0.8937 - val_loss: 0.3972 - val_au
roc: 0.7211
Epoch 9/150
69918/69918 - 21s - loss: 0.2326 - auroc: 0.9182 - val_loss: 0.4049 - val_au
roc: 0.7189
Epoch 10/150

Epoch 00010: val_auroc improved from -inf to 0.71483, saving model to weight
s.10-0.44.hdf5
69918/69918 - 22s - loss: 0.2093 - auroc: 0.9342 - val_loss: 0.4413 - val_au
roc: 0.7148
Epoch 11/150
69918/69918 - 21s - loss: 0.1838 - auroc: 0.9512 - val_loss: 0.4278 - val_au
roc: 0.7175
Epoch 12/150
69918/69918 - 21s - loss: 0.1695 - auroc: 0.9585 - val_loss: 0.4397 - val_au
roc: 0.7084
Epoch 13/150
69918/69918 - 21s - loss: 0.1535 - auroc: 0.9663 - val_loss: 0.5058 - val_au
roc: 0.7175
Epoch 14/150
Restoring model weights from the end of the best epoch.
69918/69918 - 21s - loss: 0.1429 - auroc: 0.9713 - val_loss: 0.4860 - val_au
roc: 0.7115
Epoch 00014: early stopping
```

Out[111]:
```
<tensorflow.python.keras.callbacks.History at 0x2beae466ec8>
```

2.7 Saving the weights and Evaluation

In [112]:
```
# refer: https://machinelearningmastery.com/save-load-keras-deep-learning-mode
ls/

# Saving the model weights
Model3.save_weights("model3.h5")
print("Saved model to disk")

# Loading Model and weights
Model3.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[aur
oc])
Model3.load_weights("model3.h5")
print("Loaded model from disk")
```

```
Saved model to disk
Loaded model from disk
```

Evalution using keras custom roc auc

In [113]:
```
# evaluate loaded model on test data
train_score = Model3.evaluate(train,y_train, verbose=2,batch_size=(128))
print(f"train {Model3.metrics_names[1]}:  {train_score[1]}\n")
```

```
cv_score = Model3.evaluate(cv,y_cv, verbose=2,batch_size=(128))
print(f"cv {Model3.metrics_names[1]}:  {cv_score[1]}\n")

test_score = Model3.evaluate(test,y_test, verbose=2,batch_size=(128))
print(f"test {Model3.metrics_names[1]}:  {test_score[1]}\n")
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
69918/69918 - 18s - loss: 0.3478 - auroc: 0.8349
train auroc:  0.8349214792251587

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
21850/21850 - 5s - loss: 0.3770 - auroc: 0.7528
cv auroc:  0.752826988697052

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
17480/17480 - 4s - loss: 0.3807 - auroc: 0.7430
test auroc:  0.7430437207221985
```

Evalution using direct sklearn roc auc

In [114]:
```
# sklearn ROC AUC
train_score = Model3.predict(train)
print(f"Train {Model3.metrics_names[1]}:  {sklearn_roc(y_train,train_score)}\n
")

cv_score = Model3.predict(cv)
print(f"Cv {Model3.metrics_names[1]}:  {sklearn_roc(y_cv,cv_score)}\n")

test_score = Model3.predict(test)
print(f"test {Model3.metrics_names[1]}:  {sklearn_roc(y_test,test_score)}\n")
```

```
WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Train auroc:  0.844925395892924

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>
Cv auroc:  0.7609313680046971

WARNING:tensorflow:Falling back from v2 loop because of error: Failed to fin
d data adapter that can handle input: (<class 'list'> containing values of t
ypes {"<class 'pandas.core.series.Series'>", "<class 'numpy.ndarray'>"}), <c
lass 'NoneType'>

test auroc:  0.7416349716749208
```

4.Result

In [16]:
```
%load_ext tensorboard
# Start TENSORBOARD
!del /q %TMP%\.tensorboard-info\* # Execute in case of 'localhost refused to c
onnect'
```

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

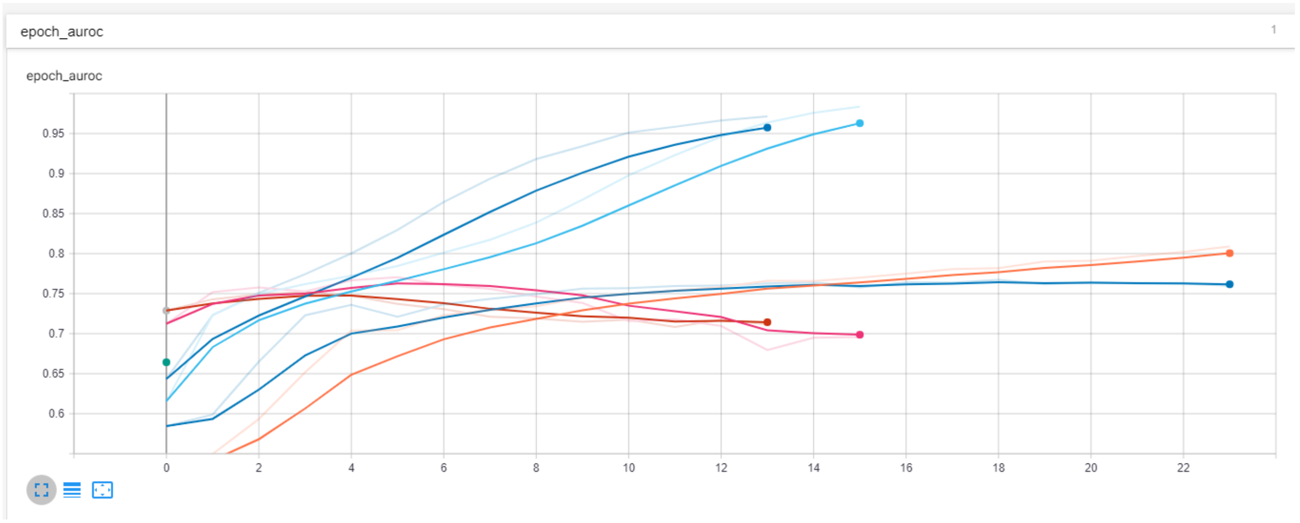
```
Could Not Find C:\Users\family\Desktop\adi_notepad\DONOR Choose\#
```

In [ ]:
```
%tensorboard --logdir {'logs'}  --host localhost
```

4.1 EPOCH AUROC graph from tensorboard

In [14]:
```
from IPython.display import Image
Image("epoch auroc.png")
```

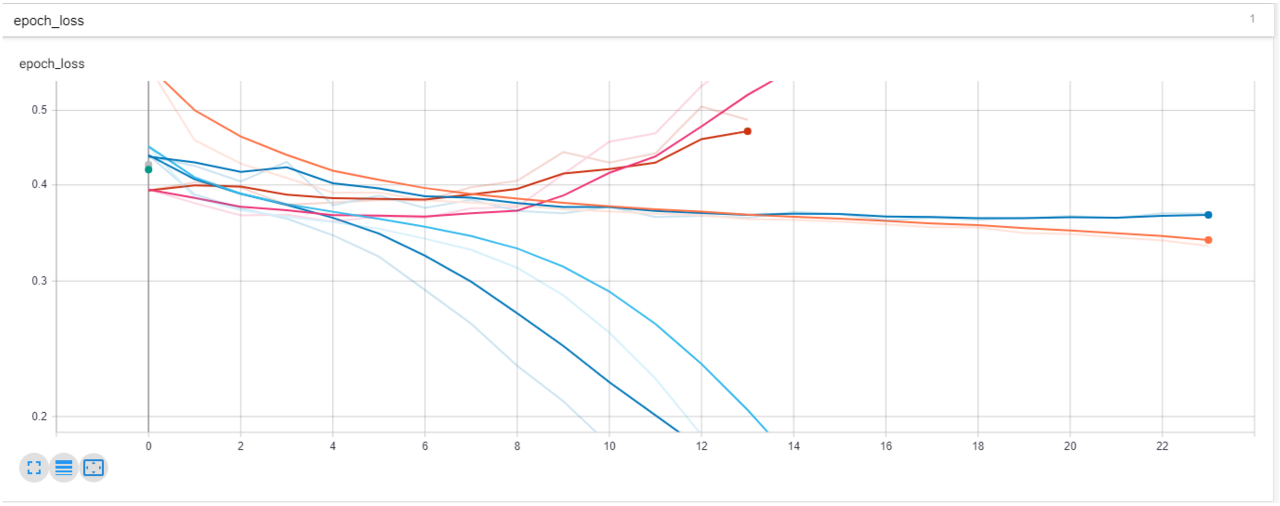
Out[14]:



4.2 EPOCH Loss graph

In [18]:
```
from IPython.display import Image
Image("epoch loss.png")
```

Out[18]:



4.3 Pretty Table

In [6]:
```
from prettytable import PrettyTable
x=PrettyTable(field_names=["Model","train_roc_auc","cv_roc_auc","test_roc_auc"
])
x.add_row(["Model1",0.780,0.757,0.750])
x.add_row(["Model2",0.740,0.702,0.701])
x.add_row(["Model3",0.844,0.760,0.741])
print(x)
```

```
+--------+---------------+------------+--------------+
| Model  | train_roc_auc | cv_roc_auc | test_roc_auc |
+--------+---------------+------------+--------------+
| Model1 |      0.78     |   0.757    |     0.75     |
| Model2 |      0.74     |   0.702    |    0.701     |
| Model3 |     0.844     |    0.76    |    0.741     |
+--------+---------------+------------+--------------+
```

END:)