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 (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **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!`

Feature	Description
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 [ ]:

```
# !pip install tensorflow==1.15.0
# !pip install keras==2.3.1
# import tensorflow
# tensorflow.__version__
```

In [ ]:

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

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

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

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

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

from tqdm import tqdm
import os

from pandas import HDFStore,DataFrame
from pandas import read_hdf
from sklearn.model_selection import train_test_split
import os
# from chart_studio.plotly import plotly
# import plotly.offline as offline
# import plotly.graph_objs as go
#offline.init_notebook_mode()
from collections import Counter
```

In [ ]:

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

```
Mounted at /content/drive
```

In [ ]:
```
!cp -r '/content/drive/My Drive/_datasets/glove_vectors' '/content/'
!cp -r '/content/drive/My Drive/_datasets/resources.csv' '/content/'
!cp -r '/content/drive/My Drive/_datasets/train_data.csv' '/content/'
```

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

In [ ]:
```
## drop unnecesary columns
project_data.drop(columns=['Unnamed: 0'],inplace=True)
```

In [ ]:
```
resource_data.head(2)
```

Out[12]:

	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

In [ ]:
```
print('Number of train points in train data',project_data.shape)
print('*'*50)
print('Number of train points in resource data',project_data.shape)
print('*'*50)
print('Columns in resource data',resource_data.columns.values)
print('*'*50)
print('Columns in project data',project_data.columns.values)
```

```
Number of train points in train data (109248, 16)
**************************************************
Number of train points in resource data (109248, 16)
**************************************************
Columns in resource data ['id' 'description' 'quantity' 'price']
**************************************************
Columns in project data ['id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

2. Preprocessing Categorical Features: project_grade_category

In [ ]:
```
project_data['project_grade_category'].value_counts()
```

Out[14]:
```
Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
```

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

Out[15]:
```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

3. Preprocessing Categorical Features: project_subject_categories

In [ ]:
```
1 project_data['project_subject_categories'].value_counts(dropna=False)
```

Out[16]:
```
Literacy & Language                        23655
Math & Science                             17072
Literacy & Language, Math & Science        14636
Health & Sports                            10177
Music & The Arts                            5180
Special Needs                               4226
Literacy & Language, Special Needs          3961
Applied Learning                            3771
Math & Science, Literacy & Language         2289
Applied Learning, Literacy & Language       2191
History & Civics                            1851
Math & Science, Special Needs               1840
Literacy & Language, Music & The Arts       1757
Math & Science, Music & The Arts            1642
Applied Learning, Special Needs             1467
History & Civics, Literacy & Language       1421
Health & Sports, Special Needs              1391
Warmth, Care & Hunger                       1309
Math & Science, Applied Learning            1220
Applied Learning, Math & Science            1052
Literacy & Language, History & Civics        809
Health & Sports, Literacy & Language         803
Applied Learning, Music & The Arts           758
Math & Science, History & Civics             652
Literacy & Language, Applied Learning        636
Applied Learning, Health & Sports            608
Math & Science, Health & Sports              414
History & Civics, Math & Science             322
History & Civics, Music & The Arts           312
Special Needs, Music & The Arts              302
Health & Sports, Math & Science              271
History & Civics, Special Needs              252
Health & Sports, Applied Learning            192
Applied Learning, History & Civics           178
Health & Sports, Music & The Arts            155
Music & The Arts, Special Needs              138
Literacy & Language, Health & Sports          72
Health & Sports, History & Civics             43
History & Civics, Applied Learning            42
Special Needs, Health & Sports                42
Special Needs, Warmth, Care & Hunger          23
Health & Sports, Warmth, Care & Hunger        23
Music & The Arts, Health & Sports             19
Music & The Arts, History & Civics            18
History & Civics, Health & Sports             13
Math & Science, Warmth, Care & Hunger         11
Applied Learning, Warmth, Care & Hunger       10
Music & The Arts, Applied Learning            10
Literacy & Language, Warmth, Care & Hunger     9
Music & The Arts, Warmth, Care & Hunger        2
History & Civics, Warmth, Care & Hunger        1
Name: project_subject_categories, dtype: int64
```

In [ ]:

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

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

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

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

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

4. Preprocessing Categorical Features: teacher_prefix

In [ ]:

```
project_data['teacher_prefix'].value_counts(dropna=False)
```

Out[18]:

```
Mrs.       57269
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
NaN            3
Name: teacher_prefix, dtype: int64
```

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

In [ ]:

```
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
```

In [ ]:
```
1 project_data['teacher_prefix'].value_counts(dropna=False)
```

Out[20]:
```
Mrs.       57272
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
Name: teacher_prefix, dtype: int64
```

Remove '.'
convert all the chars to small

In [ ]:
```
1 project_data['teacher_prefix']=project_data['teacher_prefix'].str.replace('.','')
2 project_data['teacher_prefix']=project_data['teacher_prefix'].str.lower()
3 project_data['teacher_prefix'].value_counts()
```

Out[21]:
```
mrs        57272
ms         38955
mr         10648
teacher     2360
dr            13
Name: teacher_prefix, dtype: int64
```

5. Preprocessing Categorical Features: project_subject_subcategories

In [ ]:
```
1 project_data['project_subject_subcategories'].value_counts()
```

Out[22]:
```
Literacy                                 9486
Literacy, Mathematics                    8325
Literature & Writing, Mathematics        5923
Literacy, Literature & Writing           5571
Mathematics                              5379
                                         ...
Economics, Nutrition Education              1
Community Service, Music                    1
Gym & Fitness, Social Sciences              1
Parent Involvement, Team Sports             1
Extracurricular, Financial Literacy         1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

In [ ]:

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

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

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

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

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

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

6. Preprocessing Categorical Features: school_state

In [ ]:
```
1 project_data['school_state'].value_counts()
```

Out[24]:
```
CA    15388
TX     7396
NY     7318
FL     6185
NC     5091
IL     4350
GA     3963
SC     3936
MI     3161
PA     3109
IN     2620
MO     2576
OH     2467
LA     2394
MA     2389
WA     2334
OK     2276
NJ     2237
AZ     2147
VA     2045
WI     1827
AL     1762
UT     1731
TN     1688
CT     1663
MD     1514
NV     1367
MS     1323
KY     1304
OR     1242
MN     1208
CO     1111
AR     1049
ID      693
IA      666
KS      634
NM      557
DC      516
HI      507
ME      505
WV      503
NH      348
AK      345
DE      343
NE      309
SD      300
RI      285
MT      245
ND      143
WY       98
VT       80
Name: school_state, dtype: int64
```

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

Out[25]:
```
ca    15388
tx     7396
ny     7318
fl     6185
nc     5091
il     4350
ga     3963
sc     3936
mi     3161
pa     3109
in     2620
mo     2576
oh     2467
la     2394
ma     2389
wa     2334
ok     2276
nj     2237
az     2147
va     2045
wi     1827
al     1762
ut     1731
tn     1688
ct     1663
md     1514
nv     1367
ms     1323
ky     1304
or     1242
mn     1208
co     1111
ar     1049
id      693
ia      666
ks      634
nm      557
dc      516
hi      507
me      505
wv      503
nh      348
ak      345
de      343
ne      309
sd      300
ri      285
mt      245
nd      143
wy       98
vt       80
Name: school_state, dtype: int64
```

7. Preprocessing Categorical Features: project_title

In [ ]:

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

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

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

In [ ]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [ ]:

```
print("printing some random reviews")
print(9, project_data['project_title'].values[9])
print(34, project_data['project_title'].values[34])
print(147, project_data['project_title'].values[147])
```

```
printing some random reviews
9 Just For the Love of Reading--\r\nPure Pleasure
34 \"Have A Ball!!!\"
147 Who needs a Chromebook?\r\nWE DO!!
```

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

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

```

```
100%|██████████| 109248/109248 [00:02<00:00, 40305.44it/s]
```

In [ ]:
```
## combine all the essays into 1
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [ ]:
```
preprocessed_essays = preprocess_text(project_data['essay'])
```

```
100%|██████████| 109248/109248 [01:00<00:00, 1803.75it/s]
```

In [ ]:
```
## drop unnecessary columns
project_data.drop(columns=['teacher_id',"project_essay_1","project_essay_2"
                           ,"project_essay_3","project_essay_4"],inplace=True)
project_data.head(2)
```

Out[33]:

	id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_projects	project_is_approved	clean_categories
0	p253737	mrs	in	2016-12-05 13:43:57	grades_prek_2	Educational Support for English Learners at Home	My students need opportunities to practice beg...	0	0	literacy_language
1	p258326	mr	fl	2016-10-25 09:22:10	grades_6_8	Wanted: Projector for Hungry Learners	My students need a projector to help with view...	7	1	history_civics health_sports

In [ ]:
```
project_data['preprocessed_essay'] = preprocessed_essays
```

8. Preprocessing Numerical Values: price

In [ ]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[35]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [ ]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [ ]:
```
project_data['price'].head()
```

Out[37]:
```
0    154.60
1    299.00
2    516.85
3    232.90
4     67.98
Name: price, dtype: float64
```

In [ ]:
```
if (not os.path.isfile('processed_data_before_split.h1')):
    hdf = pd.HDFStore('processed_data_before_split.h1')
    hdf.put('project_data',project_data, format='table', data_columns=True)
    hdf.close()
```

9. Split the data into Train,Test and CV

In [ ]:
```

if (not os.path.isfile('processed_data_split.h2')):
    x_train,x_test,y_train,y_test = train_test_split(project_data,project_data['project_is_approved'],test_size=0.2,stratify=project_data['project_is_approved'])
    x_train,x_cv,y_train,y_cv = train_test_split(x_train,y_train,test_size=0.2,stratify=y_train)
    hdf = pd.HDFStore('processed_data_split.h2')
    hdf.put('x_train',x_train, format='table', data_columns=True)
    hdf.put('x_test',x_test, format='table', data_columns=True)
    hdf.put('x_cv',x_cv, format='table', data_columns=True)
    hdf.put('y_train',y_train, format='table', data_columns=True)
    hdf.put('y_test',y_test, format='table', data_columns=True)
    hdf.put('y_cv',y_cv, format='table', data_columns=True)
    hdf.close()
else:
    x_train = pd.read_hdf('processed_data_split.h2', 'x_train',mode='r')
    x_test = pd.read_hdf('processed_data_split.h2', 'x_test',mode='r')
    x_cv = pd.read_hdf('processed_data_split.h2', 'x_cv',mode='r')
    y_train =pd.read_hdf('processed_data_split.h2', 'y_train',mode='r')
    y_test =pd.read_hdf('processed_data_split.h2', 'y_test',mode='r')
    y_cv =pd.read_hdf('processed_data_split.h2', 'y_cv',mode='r')
    print(' Successfully loaded processed data split')
```

In [ ]:
```
1 x_train.head(2)
```

Out[40]:

	id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_projects	project_is_approved	clean_catego
54916	p243348	mrs	mo	2016-09-09 12:14:53	grades_3_5	Help Us Wiggle While We Work! We Need *Flexibl...	My students need flexible seating options such...	0	1	literacy_langu math_scie
33715	p173833	mrs	ca	2016-09-21 17:35:27	grades_prek_2	Using iPads as a Superpower!	My students need iPads as a teaching tool, as ...	1	1	health_sp specialne

10. Vectorization on categorical(label encoding) and numerical features

In [ ]:
```
1  ## clean categories
2  from sklearn.preprocessing import LabelEncoder
3  t = LabelEncoder()
4  t.fit(x_train['clean_categories'].values)
5
6  train_categories = t.transform(x_train['clean_categories'].values)
7  test_categories = t.transform(x_test['clean_categories'].values)
8  cv_categories = t.transform(x_cv['clean_categories'].values)
9
10 print("Shape of matrix of Train data after one hot encoding ",train_categories.shape)
11 print("Shape of matrix of Test data after one hot encoding ",test_categories.shape)
12 print("Shape of matrix of CV data after one hot encoding ",cv_categories.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918,)
Shape of matrix of Test data after one hot encoding  (21850,)
Shape of matrix of CV data after one hot encoding  (17480,)
```

In [ ]:

```
## clean subcategories

t = LabelEncoder()
t.fit(x_train['clean_subcategories'].values)

# https://www.thetopsites.net/article/51321922.shtml
# few categories which are not in train but are in test are labled as unknown
x_test['clean_subcategories'] = x_test['clean_subcategories'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
x_cv['clean_subcategories'] = x_cv['clean_subcategories'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
t.classes_ = np.append(t.classes_, '<unknown>')

train_subcategories = t.transform(x_train['clean_subcategories'].values)
test_subcategories = t.transform(x_test['clean_subcategories'].values)
cv_subcategories = t.transform(x_cv['clean_subcategories'].values)

print("Shape of matrix of Train data after one hot encoding ",train_subcategories.shape)
print("Shape of matrix of Test data after one hot encoding ",test_subcategories.shape)
print("Shape of matrix of CV data after one hot encoding ",cv_categories.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918,)
Shape of matrix of Test data after one hot encoding  (21850,)
Shape of matrix of CV data after one hot encoding  (17480,)
```

In [ ]:

```
# we use count vectorizer to convert the values into one hot vectors
##  school state

t = LabelEncoder()
t.fit(x_train['school_state'].values)

# https://www.thetopsites.net/article/51321922.shtml
# few categories which are not in train but are in test are labled as unknown
x_test['school_state'] = x_test['school_state'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
x_cv['school_state'] = x_cv['school_state'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
t.classes_ = np.append(t.classes_, '<unknown>')

sklstate_train = t.transform(x_train['school_state'].values)
sklstate_test = t.transform(x_test['school_state'].values)
sklstate_cv = t.transform(x_cv['school_state'].values)

print("Shape of matrix of Train data after one hot encoding ",sklstate_train.shape)
print("Shape of matrix of Test data after one hot encoding ",sklstate_test.shape)
print("Shape of matrix of CV data after one hot encoding ",sklstate_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918,)
Shape of matrix of Test data after one hot encoding  (21850,)
Shape of matrix of CV data after one hot encoding  (17480,)
```

In [ ]:

```
# we use count vectorizer to convert the values into one hot vectors
## teacher_prefix


t = LabelEncoder()
t.fit(x_train['teacher_prefix'].values)

# https://www.thetopsites.net/article/51321922.shtml
# few categories which are not in train but are in test are labled as unknown

x_test['teacher_prefix'] = x_test['teacher_prefix'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
x_cv['teacher_prefix'] = x_cv['teacher_prefix'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
t.classes_ = np.append(t.classes_, '<unknown>')

teacher_prefix_train = t.transform(x_train['teacher_prefix'].values)
teacher_prefix_test = t.transform(x_test['teacher_prefix'].values)
teacher_prefix_cv = t.transform(x_cv['teacher_prefix'].values)

print("Shape of matrix of Train data after one hot encoding ",teacher_prefix_train.shape)
print("Shape of matrix of Test data after one hot encoding ",teacher_prefix_test.shape)
print("Shape of matrix of CV data after one hot encoding ",teacher_prefix_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918,)
Shape of matrix of Test data after one hot encoding  (21850,)
Shape of matrix of CV data after one hot encoding  (17480,)
```

In [ ]:

```
# we use count vectorizer to convert the values into one hot vectors
## project_grade

t = LabelEncoder()
t.fit(x_train['project_grade_category'].values)

# https://www.thetopsites.net/article/51321922.shtml
# few categories which are not in train but are in test are labled as unknown

x_test['project_grade_category'] = x_test['project_grade_category'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
x_cv['project_grade_category'] = x_cv['project_grade_category'].map(lambda s: '<unknown>' if s not in t.classes_ else s)
t.classes_ = np.append(t.classes_, '<unknown>')

proj_grade_train = t.transform(x_train['project_grade_category'].values)
proj_grade_test = t.transform(x_test['project_grade_category'].values)
proj_grade_cv = t.transform(x_cv['project_grade_category'].values)

print("Shape of matrix of Train data after one hot encoding ",proj_grade_train.shape)
print("Shape of matrix of Test data after one hot encoding ",proj_grade_test.shape)
print("Shape of matrix of CV data after one hot encoding ",proj_grade_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918,)
Shape of matrix of Test data after one hot encoding  (21850,)
Shape of matrix of CV data after one hot encoding  (17480,)
```

Vectorize Numerical features

In [ ]:
```
### price
from sklearn.preprocessing import StandardScaler
price_vectorize = StandardScaler()

price_vectorize.fit(x_train['price'].values.reshape(-1,1))
proj_price_train = price_vectorize.transform(x_train['price'].values.reshape(-1,1))
proj_price_test = price_vectorize.transform(x_test['price'].values.reshape(-1,1))
proj_price_cv = price_vectorize.transform(x_cv['price'].values.reshape(-1,1))

#print(price_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",proj_price_train.shape)
print("Shape of matrix of Test data after one hot encoding ",proj_price_test.shape)
print("Shape of matrix of CV data after one hot encoding ",proj_price_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918, 1)
Shape of matrix of Test data after one hot encoding  (21850, 1)
Shape of matrix of CV data after one hot encoding  (17480, 1)
```

In [ ]:
```
print(proj_price_train[0],'\n',proj_price_train[1])
```

```
[0.4741993]
 [-0.07774343]
```

In [ ]:
```
## quantity
quantity_vectorize = StandardScaler()

quantity_vectorize.fit(x_train['quantity'].values.reshape(-1,1))
proj_quantity_train = quantity_vectorize.transform(x_train['quantity'].values.reshape(-1,1))
proj_quantity_test = quantity_vectorize.transform(x_test['quantity'].values.reshape(-1,1))
proj_quantity_cv = quantity_vectorize.transform(x_cv['quantity'].values.reshape(-1,1))

#print(quantity_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",proj_quantity_train.shape)
print("Shape of matrix of Test data after one hot encoding ",proj_quantity_test.shape)
print("Shape of matrix of CV data after one hot encoding ",proj_quantity_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918, 1)
Shape of matrix of Test data after one hot encoding  (21850, 1)
Shape of matrix of CV data after one hot encoding  (17480, 1)
```

In [ ]:
```
print(proj_quantity_train[0],'\n',proj_quantity_train[1])
```

```
[0.04025279]
 [-0.57287171]
```

In [ ]:
```
## teacher_number_of_previously_posted_projects
prev_proj_vectorize = StandardScaler()

prev_proj_vectorize.fit(x_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
proj_prev_proj_train = prev_proj_vectorize.transform(x_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
proj_prev_proj_test = prev_proj_vectorize.transform(x_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
proj_prev_proj_cv = prev_proj_vectorize.transform(x_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

#print(prev_proj_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",proj_prev_proj_train.shape)
print("Shape of matrix of Test data after one hot encoding ",proj_prev_proj_test.shape)
print("Shape of matrix of CV data after one hot encoding ",proj_prev_proj_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (69918, 1)
Shape of matrix of Test data after one hot encoding  (21850, 1)
Shape of matrix of CV data after one hot encoding  (17480, 1)
```

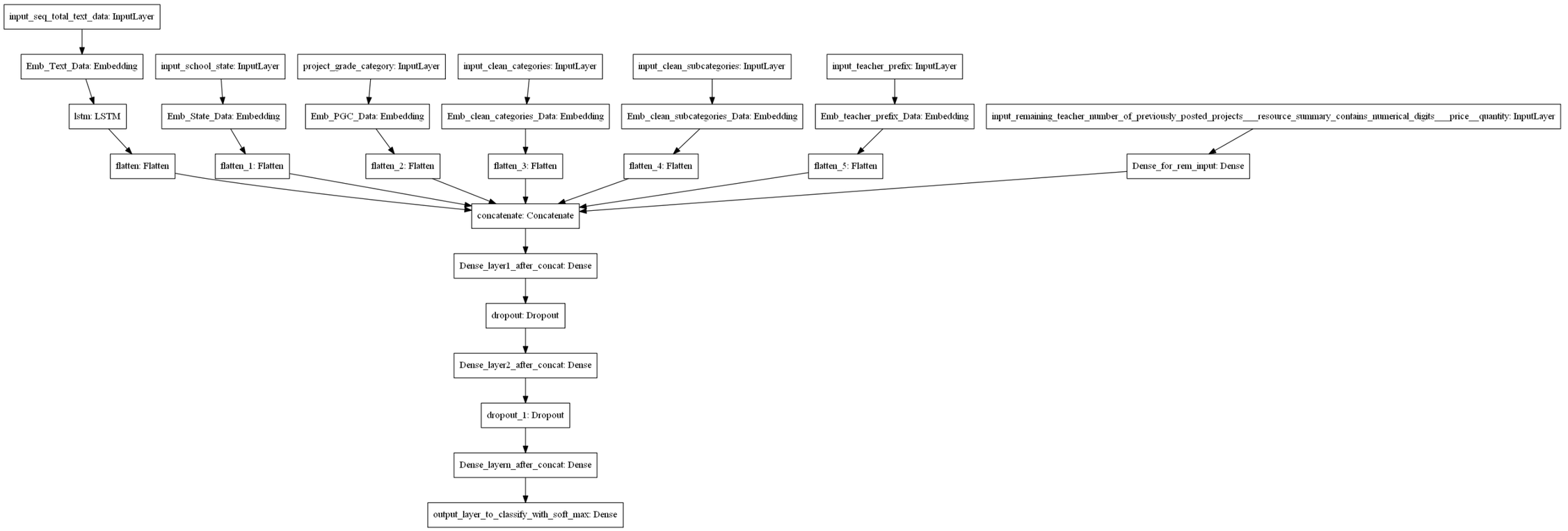
In [ ]:
```

### concatenating numerical features for building model

numerical_train = np.concatenate((proj_price_train,proj_quantity_train,proj_prev_proj_train),axis=1)
numerical_test = np.concatenate((proj_price_test,proj_quantity_test,proj_prev_proj_test),axis=1)
numerical_cv = np.concatenate((proj_price_cv,proj_quantity_cv,proj_prev_proj_cv),axis=1)

print("Shape of matrix of Train data after one hot encoding and concatination ",numerical_train.shape)
print("Shape of matrix of Test data after one hot encoding and concatination ",numerical_test.shape)
print("Shape of matrix of CV data after one hot encoding and concatination ",numerical_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding and concatination  (69918, 3)
Shape of matrix of Test data after one hot encoding and concatination  (21850, 3)
Shape of matrix of CV data after one hot encoding and concatination  (17480, 3)
```

Model-1

Build and Train deep neural network as shown below

Type *Markdown* and LaTeX: α^2



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

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

In [ ]:
```
## convert essay to sequences
'''The essay is in textual format ,we need to convert to sequences of index and pad them'''
from keras.preprocessing.text import Tokenizer
tok = Tokenizer()
tok.fit_on_texts(x_train['preprocessed_essay'].values)
seq_x_train = tok.texts_to_sequences(x_train['preprocessed_essay'].values)
seq_x_test = tok.texts_to_sequences(x_test['preprocessed_essay'].values)
seq_x_cv = tok.texts_to_sequences(x_cv['preprocessed_essay'].values)
vocab_size = len(tok.word_index) + 1
```

```
Using TensorFlow backend.
```

In [ ]:
```
print(vocab_size)
```

```
47255
```

In [ ]:

```
## lets pad the sequenced essay
'''After indexing the essay lets padd them using post padding '''
from keras.preprocessing import sequence
max_review_length = 300
padseq_x_train = sequence.pad_sequences(seq_x_train,maxlen=max_review_length,padding='post')
padseq_x_test = sequence.pad_sequences(seq_x_test,maxlen=max_review_length,padding='post')
padseq_x_cv = sequence.pad_sequences(seq_x_cv,maxlen=max_review_length,padding='post')
```

In [ ]:

```
print('Train data after padding and sequencing')
print('*'*50)
print(padseq_x_train[1],len(padseq_x_train[1]))
```

```
Train data after padding and sequencing
**************************************************
[   1   94  211  199  496 1270  192   43   16    1   11   94   52    1
  312  387  865  735   31  192  176   70   60  149  342   63  135    1
  211 1597 1628 1022 6629 4427  118    3    1   11  702 4297  133   29
    6  264   51  216   83  342  138   29    3  548 1640 1841  443  138
  118    1  871   94   52  209  264   51    1   10   73  732  496  903
  496 2069  101  732  887  496  144 1520 3248  196 3175  985  157  159
  216   83  157  216  149  174  445   87  406 1216    7 1677    1  568
  272  216   83   62    1  435   63 2188  426   95    8    9    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    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    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    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] 300
```

In [ ]:

```
'''using glove vectors lets create a embedding matrix such that for every word
  in vocabulary we store its corresponding glove vector in matrix form'''
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
```

In [ ]:

```
import numpy as np
#embedd_matrix = np.zeros((vocab_len,max_review_length))
glove_words = model.keys()
emd_i =dict()

## lets create a dictionary that stores the 300 dim glove vector as value and the word's index as key
for i,w in tok.index_word.items():
    #if w in glove_words:
    emd_i[i] = model.get(w)
    #else:  emd_i[i] = np.zeros((1,max_review_length))

## emd_matrix stores all the 300 dimensional glove vectors of words based on their rank from  the tokenizer.
## the most frequent  word is given the highest rank

# emd_matrix = np.zeros((vocab_len,max_review_length))
# print(emd_matrix.shape)
# for i in range(1,vocab_len+1):
#     emd_matrix[i-1] = emd_i[i]

# create a weight matrix for words in training docs
print('Loaded %s word vectors.' % len(emd_i))
# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size, 300))
for word, i in tok.word_index.items():
    embedding_vector = model.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
Loaded 47254 word vectors.
```

In [ ]:

```
print('shape of embedding matrix = ',embedding_matrix.shape)
```

```
shape of embedding matrix =  (47255, 300)
```

In [ ]:

```
if  os.path.isfile('model_inputs_labelencode.pkl') :
    os.remove("model_inputs_labelencode.pkl")
    print("File model_inputs Removed!")
with open('model_inputs_labelencode.pkl', 'wb') as f:
    pickle.dump([emd_i,embedding_matrix,seq_x_train,seq_x_test,seq_x_cv,
                 padseq_x_train,sklstate_train,proj_grade_train,train_categories,train_subcategories,
    teacher_prefix_train,numerical_train,
    padseq_x_test,sklstate_test,proj_grade_test,test_categories,test_subcategories,
    teacher_prefix_test,numerical_test,
    padseq_x_cv,sklstate_cv,proj_grade_cv,cv_categories,cv_subcategories,
    teacher_prefix_cv,numerical_cv],f)

```

Model 1

In [ ]:
```
import tensorflow as tf
from keras.callbacks import TensorBoard
import keras
import keras.backend as k
from sklearn.metrics import roc_auc_score
from keras.layers import Dropout,Input,Activation,Dense,Embedding,concatenate,LSTM,Flatten,BatchNormalization
from keras.models import Model
def aucroc(y_true,y_pred):
    try:
        return tf.py_func(roc_auc_score, (y_true, y_pred), tf.double)
    except ValueError:
        pass
```

```
Using TensorFlow backend.
```

Hyperparameter tuning

* hyperparamters:
* embedding output dimension
* number of lstm layers
* numerical layer for numerical input

In [ ]:
```
### convert y_train,y_test and y_cv to categorical
## one hot encode the target labels
from keras.utils import np_utils
from keras.regularizers import l2
classes=2

if  not os.path.isfile('model_input_cat_lables.pkl') :
    y_train_cat = np_utils.to_categorical(y_train,classes)
    y_test_cat = np_utils.to_categorical(y_test,classes)
    y_cv_cat = np_utils.to_categorical(y_cv,classes)
    with open('model_input_cat_lables.pkl', 'wb') as f:
        pickle.dump([y_train_cat,y_test_cat,y_cv_cat] , f)
else:
    y_train_cat,y_test_cat,y_cv_cat = pickle.load(open('model_input_cat_lables.pkl', 'rb'))
    print('--------Successfully loaded one hot y lables----------')
```

In [ ]:

```

## clear the graph of the tensorflow
k.clear_session()
### defining all the Input layer
set_random_seed(2)
input_seq_total_text_data = Input(shape=padseq_x_train[0].shape,name='text_Input')
input_school_state = Input(shape=(1,),name='school_state_Input')
input_project_grade_category = Input(shape=(1,),name='project_grade_category_Input')
input_clean_categories = Input(shape=(1,),name='input_clean_categories_Input')
input_clean_subcategories = Input(shape=(1,),name='input_clean_subcategories_Input')
input_teacher_prefix = Input(shape=(1,),name='input_teacher_prefix')
input_numerical = Input(shape=(numerical_train.shape[1],),name='input_numerical')

auc_scores_model1 = []
if (not os.path.isfile('tuning_output.pkl')):
    for embedding_index in [128,64,32]:
        for lstm_index in [128,64,32]:
            for num_dense_index in [128,64,32]:
                ## Define embedding layers for all inputs
                embedding_layer_text = Embedding(input_dim=vocab_size,output_dim=300,weights = [embedding_matrix]
                                                 ,trainable=False)(input_seq_total_text_data)
                embedding_layer_school_state = Embedding(input_dim=1,output_dim=embedding_index,
                                                         input_length=1)(input_school_state)
                embedding_layer_project_grade_category = Embedding(input_dim=1,output_dim=embedding_index,
                                                         input_length=1)(input_project_grade_category)
                embedding_layer_clean_categories = Embedding(input_dim=1,output_dim=embedding_index,
                                                         input_length=1)(input_clean_categories)
                embedding_layer_clean_subcategories = Embedding(input_dim=1,output_dim=embedding_index,
                                                         input_length=1)(input_clean_subcategories)
                embedding_layer_teacher_prefix = Embedding(input_dim=1,output_dim=embedding_index,
                                                         input_length=1)(input_teacher_prefix)

                ### Define LSTM for the text
                '''Return sequences = True ensure output from all theLSTM is returned not just the final output from last LSTM'''
                lstm_layer_text = LSTM(lstm_index,return_sequences=True)(embedding_layer_text)

                ### Define flatten layer and Dense layer for numerical input
                flatten_text = Flatten()(lstm_layer_text)
                flatten_school_state = Flatten()(embedding_layer_school_state)
                flatten_project_grade_category = Flatten()(embedding_layer_project_grade_category)
                flatten_clean_categories = Flatten()(embedding_layer_clean_categories)
                flatten_clean_subcategories = Flatten()(embedding_layer_clean_subcategories)
                flatten_teacher_prefix = Flatten()(embedding_layer_teacher_prefix)
                rem_input_dense = Dense(num_dense_index,activation='relu',kernel_initializer='he_normal')(input_numerical)

                ##Concatenate all the layers
                concat_layer = concatenate([flatten_text,flatten_school_state,flatten_project_grade_category,flatten_clean_categories,
                                            flatten_clean_subcategories,flatten_teacher_prefix,rem_input_dense])

                ##define three dense layers with dropout
                dense1_layer = Dense(256,activation='relu',kernel_initializer='he_normal')(concat_layer)
                regularization_layer1 = BatchNormalization()(dense1_layer)
                regularization_layer1 = Dropout(0.35)(regularization_layer1)
                dense2_layer = Dense(128,activation='relu',kernel_initializer='he_normal')(regularization_layer1)
                regularization_layer2 = BatchNormalization()(dense2_layer)
                regularization_layer2 = Dropout(0.35)(regularization_layer2)
                dense3_layer = Dense(64,activation='relu',kernel_initializer='he_normal')(regularization_layer2)
                regularization_layer2 = BatchNormalization()(dense3_layer)
                #regularization_layer2 = Dropout(0.25)(regularization_layer2)
```

```
                output_layer = Dense(2,activation='sigmoid',kernel_initializer='glorot_normal',activity_regularizer=l2(0.0001))(regularization_layer2)

                model1 = Model(inputs=[input_seq_total_text_data,
                                       input_school_state,input_project_grade_category,
                                       input_clean_categories,input_clean_subcategories,
                                       input_teacher_prefix,input_numerical],outputs=output_layer)

                ## Compile the model1 with default learning rate
                model1.compile(optimizer=keras.optimizers.Adam(),loss='categorical_crossentropy',metrics=['accuracy',aucroc])
                callback = tf.keras.callbacks.EarlyStopping(monitor='val_aucroc',verbose=1, patience=3,restore_best_weights=True,mode='max')
                history = model1.fit([padseq_x_train,sklstate_train,proj_grade_train,train_categories,train_subcategories,
                                      teacher_prefix_train,numerical_train],y_train_cat,epochs=10,batch_size=1000,verbose=1,
                                     validation_data=[[padseq_x_cv,sklstate_cv,proj_grade_cv,cv_categories,
                                                       cv_subcategories,teacher_prefix_cv,numerical_cv],y_cv_cat],
                                      callbacks=[callback])
                max_= np.argmax(history.history['val_aucroc'])
                print('Validation loss for embedding units={0}, lstm layer={1} ,numerical dense units={2} '.format(embedding_index,lstm_index,num_dense_index),
                      ' is :' ,history.history['val_loss'][max_])

                auc_scores_model1.append((embedding_index,lstm_index,num_dense_index,history.history['accuracy'][max_]
                                    ,history.history['loss'][max_],history.history['aucroc'][max_],
                                    history.history['val_accuracy'][max_],history.history['val_loss'][max_],history.history['val_aucroc'][max_]))

    df = pd.DataFrame(data=auc_scores_model1,columns=['Embedding units','LSTM units','Dense numerical units',
                                          'Train Accuracy','Train Loss','Train auc','Test Accuracy','Test Loss','Test auc'])
    best_param = df[df['Test auc'] == df['Test auc'].max()]
    with open('tuning_output.pkl', 'wb') as f:
        pickle.dump([df,auc_scores_model1,best_param] , f)
else:
    df,auc_scores_model1,best_param = pickle.load(open('tuning_output.pkl','rb'))
    print('----Tuning output loaded -------')
```

```
Train on 69918 samples, validate on 17480 samples
Epoch 1/10
69918/69918 [==============================] - 30s 430us/step - loss: 0.5324 - accuracy: 0.7301 - aucroc: 0.5473 - val_loss: 0.5243 - val_accuracy: 0.7248 - val_aucroc: 0.50
70
Epoch 2/10
69918/69918 [==============================] - 28s 407us/step - loss: 0.4343 - accuracy: 0.7942 - aucroc: 0.5990 - val_loss: 0.4122 - val_accuracy: 0.8062 - val_aucroc: 0.53
79
Epoch 3/10
69918/69918 [==============================] - 29s 410us/step - loss: 0.4042 - accuracy: 0.8100 - aucroc: 0.6251 - val_loss: 0.4498 - val_accuracy: 0.8096 - val_aucroc: 0.59
30
Epoch 4/10
69918/69918 [==============================] - 28s 407us/step - loss: 0.3785 - accuracy: 0.8239 - aucroc: 0.6499 - val_loss: 0.4575 - val_accuracy: 0.8146 - val_aucroc: 0.64
83
Epoch 5/10
69918/69918 [==============================] - 28s 406us/step - loss: 0.3474 - accuracy: 0.8390 - aucroc: 0.7008 - val_loss: 0.4596 - val_accuracy: 0.8190 - val_aucroc: 0.65
24
Epoch 6/10
69918/69918 [==============================] - 29s 408us/step - loss: 0.3052 - accuracy: 0.8578 - aucroc: 0.7496 - val_loss: 0.4588 - val_accuracy: 0.8188 - val_aucroc: 0.68
98
```

In [ ]:
```
1 df.head(10)
```

Out[101]:

	Embedding units	LSTM units	Dense numerical units	Train Accuracy	Train Loss	Train auc	Test Accuracy	Test Loss	Test auc
0	128	128	128	0.857833	0.305186	0.749588	0.818821	0.458777	0.689762
1	128	128	64	0.830687	0.344531	0.707659	0.806121	0.471380	0.671242
2	128	128	32	0.814297	0.408000	0.599179	0.814130	0.421263	0.657687
3	128	64	128	0.889242	0.236805	0.849639	0.782666	0.531789	0.683967
4	128	64	64	0.861180	0.348306	0.663887	0.841133	0.392900	0.687134
5	128	64	32	0.868060	0.297007	0.751041	0.829519	0.451504	0.686675
6	128	32	128	0.822392	0.389784	0.644577	0.827689	0.386795	0.706951
7	128	32	64	0.847121	0.389075	0.562900	0.854348	0.392449	0.613339
8	128	32	32	0.818473	0.391271	0.634026	0.817334	0.407131	0.683609
9	64	128	128	0.822864	0.362238	0.692673	0.810412	0.417715	0.686193

Best Hyperparameter values

In [ ]:
```
1 df[df['Test auc'] == df['Test auc'].max()]
```

Out[25]:

	Embedding units	LSTM units	Dense numerical units	Train Accuracy	Train Loss	Train auc	Test Accuracy	Test Loss	Test auc
24	32	32	128	0.859121	0.327327	0.725428	0.834153	0.409079	0.714769

In [ ]:
```
1 # Load the TensorBoard notebook extension
2 %load_ext tensorboard
3 import datetime, os
4 from tensorflow import set_random_seed
5
```

Train the Model with best hyperparameters

In [ ]:

```
from keras.callbacks import ModelCheckpoint,ReduceLROnPlateau
# Clear any logs from previous runs
!rm -rf ./logs/
#set_random_seed(65)
## clear the graph of the tensorflow
tf.keras.backend.clear_session()
### defining all the Input layer
input_seq_total_text_data = Input(shape=padseq_x_train[0].shape,name='text_Input')
input_school_state = Input(shape=(1,),name='school_state_Input')
input_project_grade_category = Input(shape=(1,),name='project_grade_category_Input')
input_clean_categories = Input(shape=(1,),name='input_clean_categories_Input')
input_clean_subcategories = Input(shape=(1,),name='input_clean_subcategories_Input')
input_teacher_prefix = Input(shape=(1,),name='input_teacher_prefix')
input_numerical = Input(shape=(numerical_train.shape[1],),name='input_numerical')

## Define embedding layers for all inputs
embedding_layer_text = Embedding(input_dim=vocab_size,output_dim=300,weights = [embedding_matrix],trainable=False)(input_seq_total_text_data)
embedding_layer_school_state = Embedding(input_dim=1,output_dim=int(best_param['Embedding units']))(input_school_state)
embedding_layer_project_grade_category = Embedding(input_dim=1,output_dim=int(best_param['Embedding units']))(input_project_grade_category)
embedding_layer_clean_categories = Embedding(input_dim=1,output_dim=int(best_param['Embedding units']))(input_clean_categories)
embedding_layer_clean_subcategories = Embedding(input_dim=1,output_dim=int(best_param['Embedding units']))(input_clean_subcategories)
embedding_layer_teacher_prefix = Embedding(input_dim=1,output_dim=int(best_param['Embedding units']))(input_teacher_prefix)

### Define LSTM for the text
'''Return sequences = True ensure output from all theLSTM is returned not just the final output from last LSTM'''
lstm_layer_text = LSTM(int(best_param['LSTM units']),return_sequences=True)(embedding_layer_text)

### Define flatten layer and Dense layer for numerical input
flatten_text = Flatten()(lstm_layer_text)
flatten_school_state = Flatten()(embedding_layer_school_state)
flatten_project_grade_category = Flatten()(embedding_layer_project_grade_category)
flatten_clean_categories = Flatten()(embedding_layer_clean_categories)
flatten_clean_subcategories = Flatten()(embedding_layer_clean_subcategories)
flatten_teacher_prefix = Flatten()(embedding_layer_teacher_prefix)
rem_input_dense = Dense(int(best_param['Dense numerical units']),activation='relu',kernel_initializer='he_normal')(input_numerical)

##Concatenate all the layers
concat_layer = concatenate([flatten_text,flatten_school_state,flatten_project_grade_category,flatten_clean_categories,
                            flatten_clean_subcategories,flatten_teacher_prefix,rem_input_dense])

##define three dense layers with dropout
dense1_layer = Dense(512,kernel_initializer='he_normal')(concat_layer)
activation = LeakyReLU(0.3)(dense1_layer)
regularization_layer1 = BatchNormalization()(activation)
regularization_layer1 = Dropout(0.15)(regularization_layer1)
dense2_layer = Dense(256,kernel_initializer='he_normal')(regularization_layer1)
activation = LeakyReLU(0.3)(dense2_layer)
regularization_layer2 = BatchNormalization()(activation)
regularization_layer2 = Dropout(0.25)(regularization_layer2)
dense3_layer = Dense(128,kernel_initializer='he_normal')(regularization_layer2)
activation = LeakyReLU(0.3)(dense3_layer)
regularization_layer2 = BatchNormalization()(activation)
regularization_layer2 = Dropout(0.15)(regularization_layer2)
dense4_layer = Dense(64,kernel_initializer='he_normal')(regularization_layer2)
activation = LeakyReLU(0.3)(dense4_layer)
regularization_layer2 = BatchNormalization()(activation)
regularization_layer2 = Dropout(0.25)(regularization_layer2)
dense5_layer = Dense(32,kernel_initializer='he_normal')(regularization_layer2)
activation = LeakyReLU(0.3)(dense5_layer)
regularization_layer2 = BatchNormalization()(activation)
```

```
regularization_layer2 = Dropout(0.25)(regularization_layer2)
output_layer = Dense(2,activation='sigmoid',kernel_initializer='glorot_normal',activity_regularizer=l2(0.002))(regularization_layer2)

if not os.path.isfile('best_model_output.pkl'):
    model1 = Model(inputs=[input_seq_total_text_data,
                           input_school_state,input_project_grade_category,
                           input_clean_categories,input_clean_subcategories,
                           input_teacher_prefix,input_numerical],outputs=output_layer)

    model1.compile(optimizer=keras.optimizers.Adam(),loss='categorical_crossentropy',metrics=['accuracy',aucroc])

    log_dir="logs/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=0, write_graph=True,write_grads=True)

    ## early stopping
    #https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping
    #https://stackoverflow.com/questions/48285129/saving-best-model-in-keras
    # https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau
    callback = tf.keras.callbacks.EarlyStopping(monitor='val_aucroc',verbose=1, patience=4,restore_best_weights=True,mode='max')
    mcp_save = ModelCheckpoint('mdl_wts.hdf5', save_best_only=True, monitor='val_aucroc', mode='max')
    reduce_lr_2 = ReduceLROnPlateau(monitor='val_aucroc', factor=0.2,patience=2, min_lr=0.001,verbose = 1,mode='max')
    history = model1.fit([padseq_x_train,sklstate_train,proj_grade_train,train_categories,train_subcategories,
                          teacher_prefix_train,numerical_train],y_train_cat,epochs=25,batch_size=1000,verbose=1,
                         validation_data=[[padseq_x_cv,sklstate_cv,proj_grade_cv,cv_categories,
                                           cv_subcategories,teacher_prefix_cv,numerical_cv],y_cv_cat],
                           callbacks=[tensorboard_callback,callback,mcp_save,reduce_lr_2])
    hist = history.history
    with open('best_model_output.pkl','wb') as f:
        pickle.dump(hist, f)

else:
    hist = pickle.load(open('best_model_output.pkl', 'rb'))
    print('----Model output loaded after tuning-------')
```

```
Train on 69918 samples, validate on 17480 samples
Epoch 1/25
69918/69918 [==============================] - 29s 419us/step - loss: 1.3231 - accuracy: 0.7031 - aucroc: 0.5269 - val_loss: 1.0148 - val_accuracy: 0.7858 - val_aucroc: 0.5286
Epoch 2/25
69918/69918 [==============================] - 28s 399us/step - loss: 0.8121 - accuracy: 0.8006 - aucroc: 0.5615 - val_loss: 0.6074 - val_accuracy: 0.8150 - val_aucroc: 0.5537
Epoch 3/25
69918/69918 [==============================] - 28s 398us/step - loss: 0.6442 - accuracy: 0.8168 - aucroc: 0.5961 - val_loss: 0.5800 - val_accuracy: 0.8226 - val_aucroc: 0.6332
Epoch 4/25
69918/69918 [==============================] - 28s 394us/step - loss: 0.5620 - accuracy: 0.8264 - aucroc: 0.6276 - val_loss: 0.5295 - val_accuracy: 0.8288 - val_aucroc: 0.6885
Epoch 5/25
69918/69918 [==============================] - 28s 394us/step - loss: 0.5042 - accuracy: 0.8357 - aucroc: 0.6500 - val_loss: 0.5388 - val_accuracy: 0.8020 - val_aucroc: 0.7339
Epoch 6/25
69918/69918 [==============================] - 28s 399us/step - loss: 0.4630 - accuracy: 0.8429 - aucroc: 0.6716 - val_loss: 0.4603 - val_accuracy: 0.8395 - val_aucroc: 0.7073
Epoch 7/25
69918/69918 [==============================] - 28s 398us/step - loss: 0.4340 - accuracy: 0.8486 - aucroc: 0.6872 - val_loss: 0.4614 - val_accuracy: 0.8377 - val_aucroc: 0.6988

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.001.
Epoch 8/25
69918/69918 [==============================] - 28s 399us/step - loss: 0.4078 - accuracy: 0.8546 - aucroc: 0.7137 - val_loss: 0.4485 - val_accuracy: 0.8372 - val_aucroc: 0.6979
Epoch 9/25
69918/69918 [==============================] - 28s 402us/step - loss: 0.3803 - accuracy: 0.8627 - aucroc: 0.7334 - val_loss: 1.4165 - val_accuracy: 0.5404 - val_aucroc: 0.5253
Restoring model weights from the end of the best epoch.

Epoch 00009: ReduceLROnPlateau reducing learning rate to 0.001.
Epoch 00009: early stopping
```

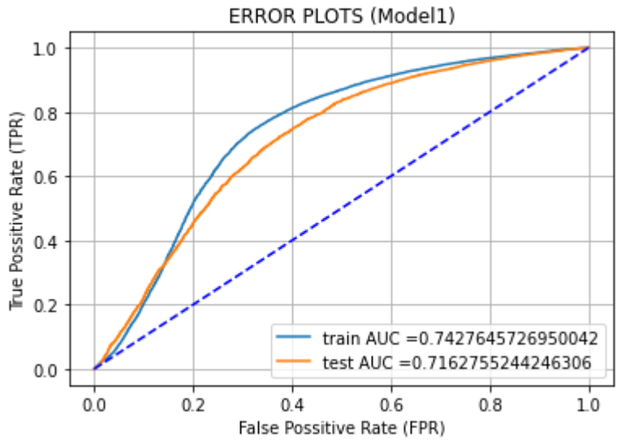
In [ ]:

```
%tensorboard --logdir logs/
# <img src='/drive/My Drive/LSTM_preprocessed/model1/epoch_aucroc.PNG' width="300" height="300">
```

Confusion Matrix and ROC_AUC curve

In [ ]:

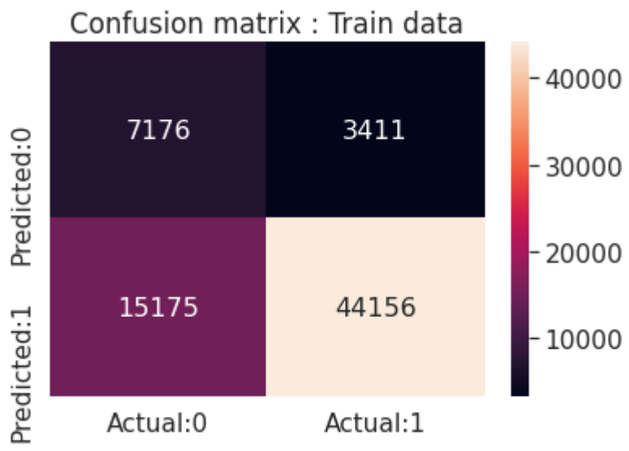
```
from sklearn.metrics import confusion_matrix
## Finding best threshold for predictions
def best_threshold(thresholds,fpr,tpr):
    t=thresholds[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshold):
    predictions = []
    for i in proba:
        if i>=threshold:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions


model1.load_weights('mdl_wts.hdf5')
## Predict the test and train
# if not os.path.file('model_predictions.pkl'):
y_test_predict = model1.predict([padseq_x_test,sklstate_test,proj_grade_test,test_categories,test_subcategories,
                         teacher_prefix_test,numerical_test],use_multiprocessing=True)[:,1]
y_train_predict = model1.predict([padseq_x_train,sklstate_train,proj_grade_train,train_categories,train_subcategories,
                         teacher_prefix_train,numerical_train],use_multiprocessing=True)[:,1]
    #
# else:
#          y_train_predict,y_test_predict = pickle.load(open('model_predictions.pkl','rb'))
#          print('------Predicts loaded------')
## Store fpr and tpr rates


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_predict)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_predict)


#plot
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Possitive Rate (FPR)")
plt.ylabel("True Possitive Rate (TPR)")
plt.title("ERROR PLOTS (Model1)")
plt.plot([0, 1], [0, 1],'b--')
plt.grid()
plt.show()

print("="*100)

best_t=best_threshold(tr_thresholds,train_fpr, train_tpr)

```




```
==================================================================================================
the maximum value of tpr*(1-fpr) 0.5044493576696496 for threshold 0.08
```

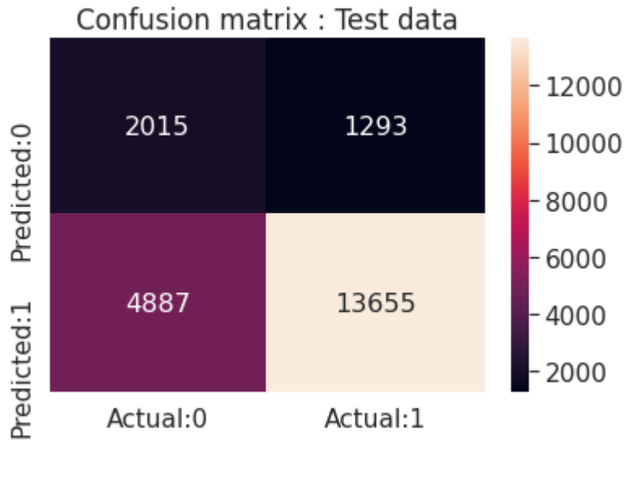
In [ ]:

```
### PLOT the matrix for Train
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_predict, best_t))
                     , range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',
            xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Train data')
plt.show()
```

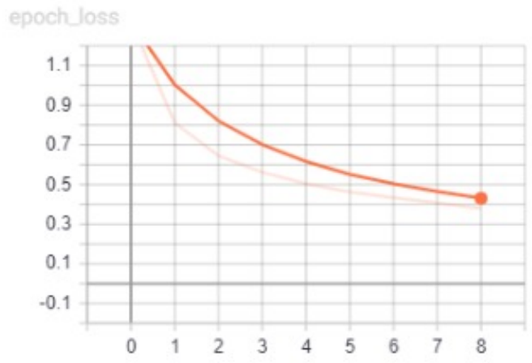


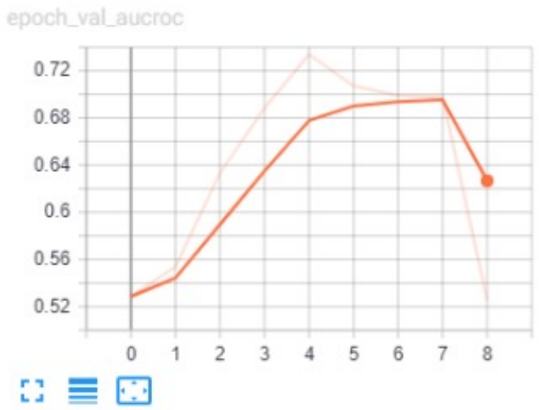
In [ ]:

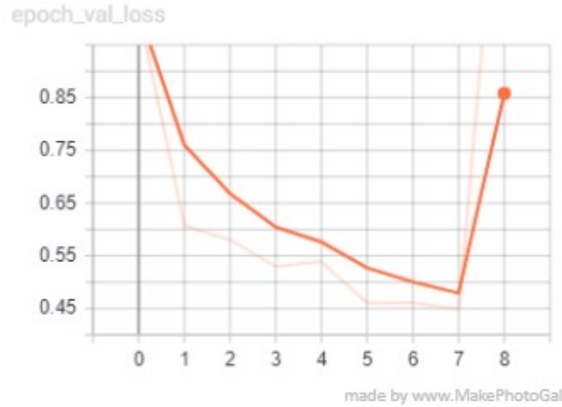
```
### PLOT the matrix for Test
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_predict, best_t))
                     , range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',
            xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Test data')
plt.show()
```



- **Validation Loss,Validation aucroc,Train Loss,Train aucroc**







Model1 Summary :

* Preprocesed and vectorized the input variables along with label encoding categorical data.
* We Tuned our model with various embedding input dimensions,dense numerical units and LSTM units with early stopping techniques and found the best combination with highest Test auc values .

* After finding the best combination we tuned our best model with this combination increasing the number of layers,
  dropout and batch normalization layers,including Leaky relu activation and activity regularizers to imporve the model performance

* Used keras callback methods to perform early stopping,reduce learning rate,model checkpoints to monitor our model
  to attain maximum validation auc.

* We see that as the number of epochs goes above 7 our Validation auc reduces and loss increases.Hence we save the model at the best epoch.

* When we plot FPR against TPR our model gives 0.74 as train auc and 0.71 as test auc.

In [ ]:

```
### load evrything to drive
# !cp -r  '/content/tuning_output.pkl'    '/content/drive/My Drive/LSTM_preprocessed/model1/'
# !cp -r  '/content/processed_data_split.h2'    '/content/drive/My Drive/LSTM_preprocessed/model1/'
# #!cp -r  '/content/processed_data_before_split.h1'    '/content/drive/My Drive/LSTM_preprocessed/model1/'
# #!cp -r  '/content/model_predictions.pkl'    '/content/drive/My Drive/LSTM_preprocessed/model1/'
# !cp -r  '/content/model_inputs_labelencode.pkl'    '/content/drive/My Drive/LSTM_preprocessed/model1/'
# !cp -r  '/content/model_input_cat_lables.pkl'    '/content/drive/My Drive/LSTM_preprocessed/model1/'
# !cp -r  '/content/best_model_output.pkl'    '/content/drive/My Drive/LSTM_preprocessed/model1/'u
# !cp -r  '/content/mdl_wts.hdf5'    '/content/drive/My Drive/LSTM_preprocessed/model1/'
```

In [ ]:

```
### load evrything from drive
# !cp -r      '/content/drive/My Drive/LSTM_preprocessed/model1/tuning_output.pkl'    '/content/'
# !cp -r      '/content/drive/My Drive/LSTM_preprocessed/model1/processed_data_split.h2'    '/content/'
# # !cp -r      '/content/drive/My Drive/LSTM_preprocessed/model1/processed_data_before_split.h1'    '/content/'
# # !cp -r      '/content/drive/My Drive/LSTM_preprocessed/model1/model_predictions.pkl'    '/content/'
# !cp -r      '/content/drive/My Drive/LSTM_preprocessed/model1/model_inputs_labelencode.pkl'    '/content/'
# !cp -r      '/content/drive/My Drive/LSTM_preprocessed/model1/model_input_cat_lables.pkl'    '/content/'
# !cp -r      '/content/drive/My Drive/LSTM_preprocessed/model1/best_model_output.pkl'    '/content/'

```