DonorsChoose

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

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer


import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/

from nltk.corpus import stopwords
import pickle

from tqdm import tqdm
import os

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

import sqlite3
```

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

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

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

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

from tqdm import tqdm
import os
```

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

Reading Data

```
import gdown #to download files from drive
url='https://drive.google.com/uc?id=1bDLwb_Vq7q2W9S89JB96PgmZG3LsLns9'
output='train.csv'
gdown.download(url,output,quiet=False)
```

```
Downloading...
From: https://drive.google.com/uc?id=1bDLwb_Vq7q2W9S89JB96PgmZG3LsLns9
To: /content/train.csv
201MB [00:02, 69.9MB/s]
'train.csv'
```

```
import gdown
output='resources.csv'
url='https://drive.google.com/uc?id=14OVXWu_SJU-lJD-jKMOCld14EZ21lYYe'
gdown.download(url,output,quiet=False)
```

```
Downloading...
From: https://drive.google.com/uc?id=14OVXWu_SJU-lJD-jKMOCld14EZ21lYYe
To: /content/resources.csv
127MB [00:01, 82.0MB/s]
'resources.csv'
```

```
#reading data
project_data=pd.read_csv('train.csv')
resource_data=pd.read_csv('resources.csv')
```

```
print('no of points in train',project_data.shape[0])
print(project_data.columns.shape)
print(project_data.columns.values)
```

```
no of points in train 109248
(17,)
['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

```
print("Number of data points in resouces data", resource_data.shape[0])
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in resouces data 1541272
['id' 'description' 'quantity' 'price']
```

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

```
project_data['Date']=project_data['project_submitted_datetime']

project_data['Date']=pd.to_datetime(project_data['Date'])


project_data.sort_values('Date',inplace=True)


project_data.drop('project_submitted_datetime',axis=1,inplace=True)



categories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/408403

# 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 categories:
    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 & Hu
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proje
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
```

```
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())


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



for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Warmth               :      1388
Care_Hunger          :      1388
History_Civics       :      5914
Music_Arts           :     10293
AppliedLearning      :     12135
SpecialNeeds         :     13642
Health_Sports        :     14223
Math_Science         :     41421
Literacy_Language    :     52239
```

Univariate Analysis: project_subject_subcategories

```
sub_categories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/408403

# 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_categories:
    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 & Hu
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())


project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proje
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	

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

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


for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

Univariate Analysis: Text features (Title)

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/408
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))
```

```
approved_word_count = project_data[project_data['project_is_approved']==1]['project_title'].str.spli
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0]['project_title'].str.spli
rejected_word_count = rejected_word_count.values
```

Univariate Analysis: Text features (Project Essay's)

```
# 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)
```

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/408
```

```
word_count = project_data['essay'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))
```

```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.split().appl
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0]['essay'].str.split().appl
rejected_word_count = rejected_word_count.values
```

Univariate Analysis: Cost per project

```
# we get the cost of the project using resource.csv file

resource_data.head(2)
```

	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

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

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

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


approved_price = project_data[project_data['project_is_approved']==1]['price'].values

rejected_price = project_data[project_data['project_is_approved']==0]['price'].values


print("\nColumns in project_data:\n")
print(project_data.columns)

#print("Head of project_data:\n")
#project_data.head()
```

```
Columns in project_data:

Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_grade_category', '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',
       'Date', 'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

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

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

we need to remove the spaces, replace the '-' with '_' and convert all the letters to small

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

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

```
project_data['clean_categories'].value_counts()
```

```
Literacy_Language                       23655
Math_Science                            17072
Literacy_Language Math_Science          14636
Health_Sports                           10177
Music_Arts                               5180
SpecialNeeds                             4226
Literacy_Language SpecialNeeds           3961
AppliedLearning                          3771
Math_Science Literacy_Language           2289
AppliedLearning Literacy_Language        2191
History_Civics                           1851
Math_Science SpecialNeeds                1840
Literacy_Language Music_Arts             1757
Math_Science Music_Arts                  1642
AppliedLearning SpecialNeeds             1467
History_Civics Literacy_Language         1421
Health_Sports SpecialNeeds               1391
Warmth Care_Hunger                       1309
Math_Science AppliedLearning             1220
AppliedLearning Math_Science             1052
Literacy_Language History_Civics          809
Health_Sports Literacy_Language           803
AppliedLearning Music_Arts                758
Math_Science History_Civics               652
Literacy_Language AppliedLearning         636
AppliedLearning Health_Sports             608
Math_Science Health_Sports                414
History_Civics Math_Science               322
History_Civics Music_Arts                 312
SpecialNeeds Music_Arts                   302
Health_Sports Math_Science                271
History_Civics SpecialNeeds               252
Health_Sports AppliedLearning             192
```

```
Health_Sports AppliedLearning              192
AppliedLearning History_Civics             178
Health_Sports Music_Arts                   155
Music_Arts SpecialNeeds                    138
Literacy_Language Health_Sports             72
Health_Sports History_Civics                43
SpecialNeeds Health_Sports                  42
History_Civics AppliedLearning              42
Health_Sports Warmth Care_Hunger            23
SpecialNeeds Warmth Care_Hunger             23
Music_Arts Health_Sports                    19
Music_Arts History_Civics                   18
History_Civics Health_Sports                13
Math_Science Warmth Care_Hunger             11
Music_Arts AppliedLearning                  10
AppliedLearning Warmth Care_Hunger          10
Literacy_Language Warmth Care_Hunger         9
Music_Arts Warmth Care_Hunger                2
History_Civics Warmth Care_Hunger            1
Name: clean_categories, dtype: int64
```

> remove spaces, 'the'
> replace '&' with '_', and ',' with '_'

```
project_data['teacher_prefix'].value_counts()
```

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

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

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

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

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

```
project_data['teacher_prefix'].value_counts()
```

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

> Remove '.'
> convert all the chars to small

```
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix'].value_counts()
```

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

```
project_data['clean_subcategories'].value_counts()
```

```
Literacy                                9486
Literacy Mathematics                    8325
Literature_Writing Mathematics          5923
Literacy Literature_Writing             5571
Mathematics                             5379
                                        ...
Gym_Fitness Warmth Care_Hunger             1
CommunityService FinancialLiteracy         1
Other Warmth Care_Hunger                   1
ESL TeamSports                             1
ParentInvolvement TeamSports               1
Name: clean_subcategories, Length: 401, dtype: int64
```

> same process we did in project_subject_categories

```
project_data['school_state'].value_counts()
```

```
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
```

```
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
```

> convert all of them into small letters

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

```
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
an     1049
```

```
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
```

```
# 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


# 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', 'himse
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'thes
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'w
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', '
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
            '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'
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't
            'won', "won't", 'wouldn', "wouldn't"]


project_data.head(5)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_gr
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	mrs	ca	
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	ms	ut	
2	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	mrs	ca	
3	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	mrs	ga	

4	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	mrs	wa

```
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 Dash and Dot Robotic Duo Needed
34 iPad Fun to be Had if We Had Covers
147 Learning to fly
```

```
# 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


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

```
100%|██████████| 109248/109248 [00:02<00:00, 40308.34it/s]
```

```
print("printing some random reviews")
print(9, preprocessed_titles[9])
print(34, preprocessed_titles[34])
print(147, preprocessed_titles[147])
```

```
printing some random reviews
9 dash dot robotic duo needed
34 ipad fun covers
147 learning fly
```

```
# 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)


print("printing some random essay")
print(9, project_data['essay'].values[9])
print('-'*50)
print(34, project_data['essay'].values[34])
print('-'*50)
```

```
print(147, project_data['essay'].values[147])
```

```
preprocessed_essays = preprocess_text(project_data['essay'].values)
```

```
100%|██████████| 109248/109248 [01:00<00:00, 1811.91it/s]
```

```
print("printing some random essay")
print(9, preprocessed_essays[9])
print('-'*50)
print(34, preprocessed_essays[34])
print('-'*50)
print(147, preprocessed_essays[147])
```

```
printing some random essay
9 remember first time saw star wars wall e robots wonderfully complex amazing see action studen
--------------------------------------------------
34 class active world children make day exciting enjoy hands creative learning received ipads c
--------------------------------------------------
147 want see fireworks come see students roundtable discussion see students engaged fervent dis
```

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

```
import tensorflow as tf
device_name=tf.test.gpu_device_name()
```

---------- LSTM on DonorsChoose ----------

```
# Extracting numerical digits from project_resource_summary

summary = []
for i in tqdm(project_data['project_resource_summary']):
    sent = decontracted(i)
    sent = ' '.join(w for w in sent.split() if w.isdigit())
    l = len(sent)
    summary.append(l)

project_data["project_summary_numerical"] = summary
```

```
100%|██████████| 109248/109248 [00:01<00:00, 71994.88it/s]
```

```
project_data_1 = project_data.drop(['Unnamed: 0', 'id', 'teacher_id', 'project_title',
                                    'project_essay_1', 'project_essay_2', 'project_essay_3', 'projec
                                    'project_resource_summary'], axis = 1)
```

Assigning independent variables (x) and dependent variable (y)

```
x = project_data_1.drop(['project_is_approved'], axis = 1)
y = project_data_1['project_is_approved']
```

Splitting into train, cv and test set

```
from sklearn.model_selection import train_test_split

# Splitting into x and y into train and test set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42, strati

# Splitting train set into tr and cv set
x_tr, x_cv, y_tr, y_cv = train_test_split(x_train, y_train, test_size = 0.25, random_state = 42, str


print("Shape of x_tr:", x_tr.shape)
print("Shape of x_cv:", x_cv.shape)
print("Shape of x_test:", x_test.shape)
print("Shape of y_tr:", y_tr.shape)
print("Shape of y_cv:", y_cv.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of x_tr: (65548, 11)
Shape of x_cv: (21850, 11)
Shape of x_test: (21850, 11)
Shape of y_tr: (65548,)
Shape of y_cv: (21850,)
Shape of y_test: (21850,)
```

Loading GloVe predefined glove word vector

There are a few different embedding vector sizes, including 50, 100, 200 and 300 dimensions.

We will use 42B 300 dimensions

Source links:

https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

https://nlp.stanford.edu/projects/glove/

https://github.com/stanfordnlp/GloVe

We have loaded zipped file. Now we will unzip the file to use for our model

Source link: https://www.geeksforgeeks.org/working-zip-files-python/

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
```

```
--2020-04-23 09:30:03--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2020-04-23 09:30:04--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2020-04-23 09:30:04--  http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80... connect
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  2.23MB/s    in 6m 29s

2020-04-23 09:36:33 (2.12 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

```
!unzip glove*.zip
```

```
Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
```

```
print('Indexing word vectors.')
```

```
embeddings_index = {}
f = open('glove.6B.300d.txt', encoding='utf-8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('Found %s word vectors.' % len(embeddings_index))
```

```
Indexing word vectors.
Found 400000 word vectors.
```

Defining sequence length, vocabulary size and embedding size.

```
# Defining sequence length, vocabulary size and embedding size

seq_len = 500
vocab_size = 100000
emb_dim = 300
```

Tokenize:

Input data to layer should be integer. So, using tokenize inbuilt function, we will integer encode the text data.

```
from keras.preprocessing.text import Tokenizer

t = Tokenizer(num_words = vocab_size)

# Fit train text data
t.fit_on_texts(x_tr['essay'])

# Sequencing train, cv and test data i.e transforming
tr_seq = t.texts_to_sequences(x_tr['essay'])
cv_seq = t.texts_to_sequences(x_cv['essay'])
test_seq = t.texts_to_sequences(x_test['essay'])
print('Done!')
```

```
Done!
```

Double-click (or enter) to edit

```
# Let's create a weight matrix of train data from the glove vector.

from numpy import zeros

word_count = min(vocab_size, len(t.word_index) + 1)

emb_matrix = zeros((word_count, emb_dim))
for word, i in t.word_index.items():
  emb_vec = embeddings_index.get(word)
  if emb_vec is not None:
    emb_matrix[i] = emb_vec
```

```
print("Number for unique words in train data:", len(t.word_index) + 1)
print("Shape of train weight matrix:", emb_matrix.shape)
```

```
Number for unique words in train data: 46122
Shape of train weight matrix: (46122, 300)
```

Padding document

Padding document is to have the same input length of each document.

```
from keras.preprocessing.sequence import pad_sequences

pad_tr = pad_sequences(tr_seq, maxlen = seq_len, padding = 'post', truncating = 'post')
pad_cv = pad_sequences(cv_seq, maxlen = seq_len, padding = 'post', truncating = 'post')
pad_test = pad_sequences(test_seq, maxlen = seq_len, padding = 'post', truncating = 'post')

print("Shape of pad_tr:", pad_tr.shape)
print("Shape of pad_cv:", pad_cv.shape)
print("Shape of pad_test:", pad_test.shape)
```

```
Shape of pad_tr: (65548, 500)
Shape of pad_cv: (21850, 500)
Shape of pad_test: (21850, 500)
```

Embedding layer for text data

```
import warnings
warnings.filterwarnings('ignore')


from keras.layers import Embedding, Dense, Flatten, Input, LSTM, Dropout, BatchNormalization, concat


input_size = min(vocab_size, len(t.word_index) + 1)

# Creating an input layer
input_lay = Input(shape = (seq_len, ), name = "Input_Text_Data")

# Creating an embedding layer
emb_lay = Embedding(input_dim = input_size, output_dim = emb_dim,
                        input_length = seq_len, weights = [emb_matrix],
                        trainable = False, name = "lstm_text_layer")(input_lay)

# Creating LSTM layer
emb_lay_text = LSTM(128, return_sequences = True, dropout = 0.3)(emb_lay)

flatten_1 = Flatten()(emb_lay_text)
```

Categorical Feature: teacher_prefix

Embedding layer for teacher_prefix

```
# Unique values
tea_pre_uni = x_tr['teacher_prefix'].nunique()
```

```
emb_tea_pre_size = int(np.ceil((tea_pre_uni) / 2))

# Creating an input layer
inp_tea_pre = Input(shape = (1,), name = "teacher_prefix")

# Creating an embedding layer
emb_tea_pre = Embedding(input_dim = tea_pre_uni, output_dim = emb_tea_pre_size,
                        trainable = True, name = "teacher_prefix_emb")(inp_tea_pre)

flatten_tea_pre = Flatten()(emb_tea_pre)
```

Label encoding teacher_prefix

```
#https://stackoverflow.com/questions/21057621/sklearn-labelencoder-with-never-seen-before-values
# train and test are pandas.DataFrame's and c is whatever column

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
c='teacher_prefix'
le.fit(x_tr[c])
x_test[c] = x_test[c].map(lambda s: '<unknown>' if s not in le.classes_ else s)
le.classes_ = np.append(le.classes_, '<unknown>')
tr_tea_pre_encode = le.transform(x_tr[c])
cv_tea_pre_encode = le.transform(x_cv[c])
test_tea_pre_encode = le.transform(x_test[c])
```

Categorical feature: school_state

Embedding layer for school_state

```
# Unique values
sch_uni = x_tr['school_state'].nunique()
emb_sch_size = int(np.ceil((sch_uni) / 2))

# Creating an input layer
inp_sch = Input(shape = (1,), name = "school_state")

# Creating an embedding layer
emb_sch = Embedding(input_dim = sch_uni, output_dim = emb_sch_size,
                        trainable = True, name = "school_state_emb")(inp_sch)

flatten_sch = Flatten()(emb_sch)
```

Label encoding for school_state

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

```
le = LabelEncoder()



c='school_state'
le.fit(x_tr[c])
x_test[c] = x_test[c].map(lambda s: '<unknown>' if s not in le.classes_ else s)
le.classes_ = np.append(le.classes_, '<unknown>')
tr_sch_encode = le.transform(x_tr[c])
cv_sch_encode = le.transform(x_cv[c])
test_sch_encode = le.transform(x_test[c])
```

Categorical feature: project_grade_category

Creating embedding layer for project_grade_category

```
# Unique values
pro_gra_uni = x_tr['project_grade_category'].nunique()
emb_pro_gra_size = int(np.ceil((pro_gra_uni) / 2))

# Creating an input layer
inp_pro_gra = Input(shape = (1,), name = "project_grade_category")

# Creating an embedding layer
emb_pro_gra = Embedding(input_dim = pro_gra_uni, output_dim = emb_pro_gra_size,
                        trainable = True, name = "project_grade_category_emb")(inp_pro_gra)
```

```
flatten_pro_gra = Flatten()(emb_pro_gra)
```

Label encoding for project_grade_category

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()



c='project_grade_category'
le.fit(x_tr[c])
x_test[c] = x_test[c].map(lambda s: '<unknown>' if s not in le.classes_ else s)
le.classes_ = np.append(le.classes_, '<unknown>')
tr_pro_gra_encode = le.transform(x_tr[c])
cv_pro_gra_encode = le.transform(x_cv[c])
test_pro_gra_encode = le.transform(x_test[c])
```

Categorical feature: project_subject_categories

Embedding layer for project_subject_categories

```
# Unique values
pro_sub_uni = x_tr['clean_categories'].nunique()
emb_pro_sub_size = int(np.ceil((pro_sub_uni) / 2))
```

```
# Creating an input layer
inp_pro_sub = Input(shape = (1,), name = "clean_categories")

# Creating an embedding layer
emb_pro_sub = Embedding(input_dim = pro_sub_uni, output_dim = emb_pro_sub_size,
                        trainable = True, name = "clean_categories_emb")(inp_pro_sub)

flatten_pro_sub = Flatten()(emb_pro_sub)
```

Label encoding for project_subject_categories

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

le.fit(x_tr['clean_subcategories'])

x_test["clean_subcategories"] = x_test["clean_subcategories"].map(lambda a: '<unknown>' if a not in
x_cv["clean_subcategories"] = x_cv["clean_subcategories"].map(lambda a: '<unknown>' if a not in le.c

le.classes_ = np.append(le.classes_, '<unknown>')

tr_pro_sub_encode = le.transform(x_tr['clean_subcategories'])
cv_pro_sub_encode = le.transform(x_cv['clean_subcategories'])
test_pro_sub_encode = le.transform(x_test['clean_subcategories'])
```

Categorical feature: project_subject_subcategories

Embedding layer for project_subject_subcategories

```
# Unique values
pro_sub_1_uni = x_tr['clean_subcategories'].nunique()
emb_pro_sub_1_size = int(min(np.ceil((pro_sub_1_uni) / 2), 50))

# Creating an input layer
inp_pro_sub_1 = Input(shape = (1,), name = "clean_subcategories")

# Creating an embedding layer
emb_pro_sub_1 = Embedding(input_dim = pro_sub_1_uni, output_dim = emb_pro_sub_1_size,
                          trainable = True, name = "cleant_subcategories_emb")(inp_pro_sub)

flatten_pro_sub_1 = Flatten()(emb_pro_sub_1)
```

Label encoding for project_subject_subcategories

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

le.fit(x_tr["clean_subcategories"])
```

```
x_test["clean_subcategories"] = x_test["clean_subcategories"].map(lambda a: '<unknown>' if a not in
x_cv["clean_subcategories"] = x_cv["clean_subcategories"].map(lambda a: '<unknown>' if a not in le.c

le.classes_ = np.append(le.classes_, '<unknown>')

tr_sub_1_encoder = le.transform(x_tr["clean_subcategories"])
cv_sub_1_encoder = le.transform(x_cv["clean_subcategories"])
test_sub_1_encoder = le.transform(x_test["clean_subcategories"])
```

Numerical Features

We will reshape the numerical features to (-1, 1). Then concatenate numerical features and standardize the fi

```
# Train data
tr_1 = x_tr['price'].values.reshape(-1, 1)
tr_2 = x_tr['quantity'].values.reshape(-1, 1)
tr_3 = x_tr['project_summary_numerical'].values.reshape(-1, 1)
tr_4 = x_tr['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1)

# CV data
cv_1 = x_cv['price'].values.reshape(-1, 1)
cv_2 = x_cv['quantity'].values.reshape(-1, 1)
cv_3 = x_cv['project_summary_numerical'].values.reshape(-1, 1)
cv_4 = x_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1)

# Test data
test_1 = x_test['price'].values.reshape(-1, 1)
```

```
test_1 = x_test['price'].values.reshape(-1, 1)
test_2 = x_test['quantity'].values.reshape(-1, 1)
test_3 = x_test['project_summary_numerical'].values.reshape(-1, 1)
test_4 = x_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1)
```

Concatenating above reshaped features

```
# Train
tr_fin = np.concatenate((tr_1, tr_2, tr_3, tr_4), axis = 1)

# CV
cv_fin = np.concatenate((cv_1, cv_2, cv_3, cv_4), axis = 1)

# Test
test_fin = np.concatenate((test_1, test_2, test_3, test_4), axis = 1)
```

Standardizing the final data

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

tr_ss = ss.fit_transform(tr_fin)
cv_ss = ss.transform(cv_fin)
test_ss = ss.transform(test_fin)
```

Embedding layer for numerical features

```
inp_num = Input(shape=(4,), name = "numerical_features")

# We are not adding Flatten layer but applying Dense layer as we already have reshaped the data to (
emb_num = Dense(100, activation = "relu")(inp_num)
```

Concatenating all the flattened layers

```
from keras.layers import concatenate

con_lay = concatenate([flatten_1, flatten_tea_pre, flatten_sch, flatten_pro_gra, flatten_pro_sub, fl
```

---------- Model: 1 ----------

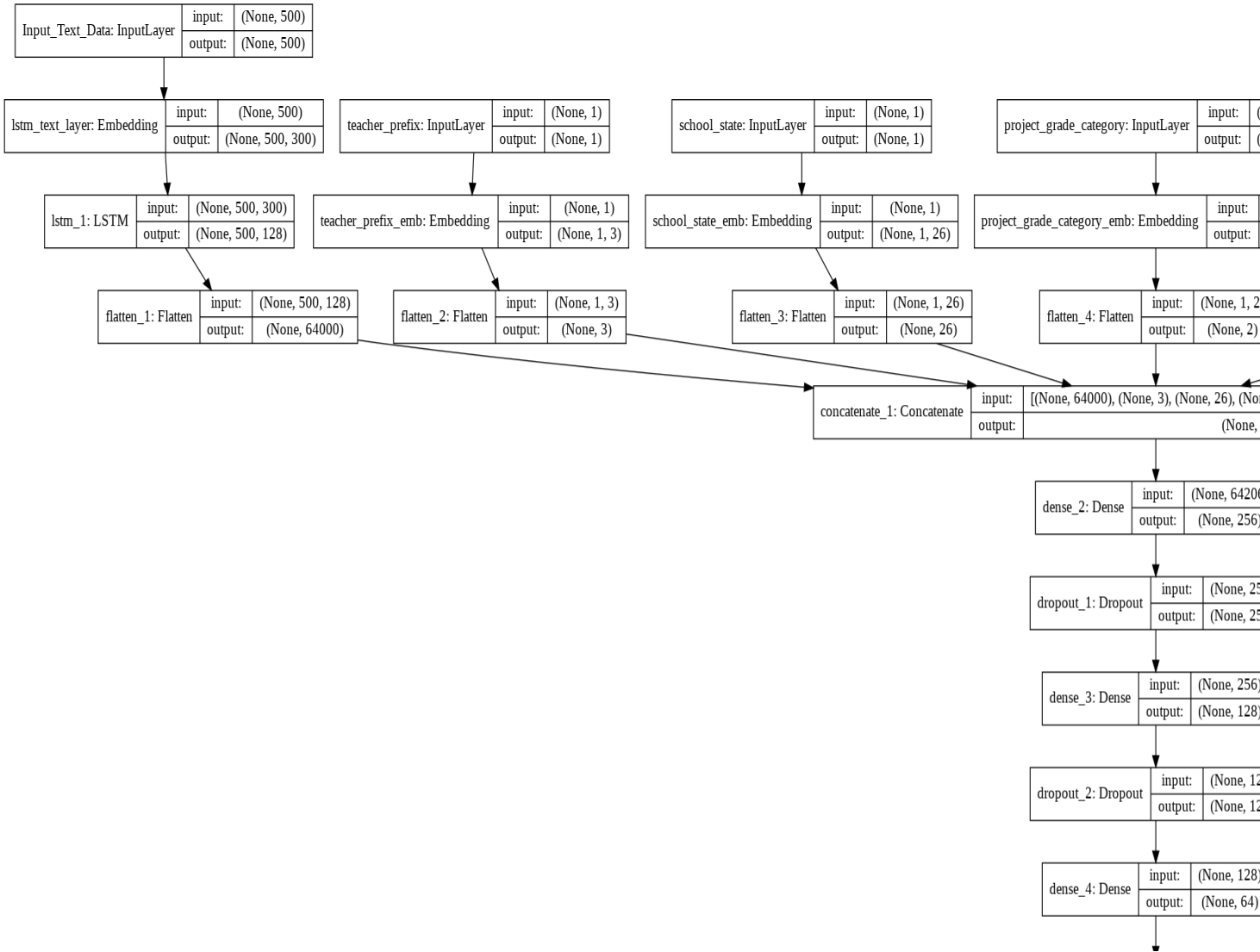
Keras model:

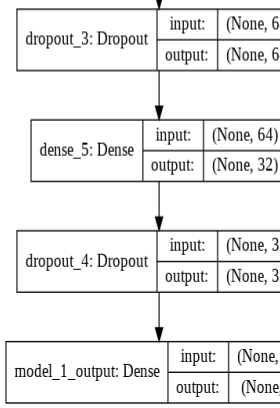
- Activation - 'relu' and 'softmax'.
- Dropout - 0.3
- kernel_regularizer - regularizers.l2(0.01)

```
from keras.models import Model
from keras import regularizers, initializers

# Layer 1
m = Dense(256, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(con_lay)
m = Dropout(0.3)(m)

# Layer 2
m = Dense(128, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m)
m = Dropout(0.3)(m)

# Layer 3
m = Dense(64, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m)
m = Dropout(0.3)(m)

# Layer 4
m = Dense(32, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m)
m = Dropout(0.3)(m)

# Output layer
output = Dense(2, activation = 'softmax', name= 'model_1_output')(m)

# Model
model_1 = Model(inputs = [input_lay, inp_tea_pre, inp_sch, inp_pro_gra,
                          inp_pro_sub, inp_pro_sub_1, inp_num], outputs = [output])
```

Network Architecture

```
# https://github.com/mmortazavi/EntityEmbedding-Working_Example/blob/master/EntityEmbedding.ipynb

import pydot_ng as pydot
from keras.utils import plot_model
from IPython.display import Image

plot_model(model_1, show_shapes = True, show_layer_names = True, to_file = 'model_1.png')

Image(retina = True, filename = 'model_1.png')
```


Input_Text_Data: InputLayer
input: (None, 500)
output: (None, 500)
lstm_text_layer: Embedding
input: (None, 500)
output: (None, 500, 300)
teacher_prefix: InputLayer
input: (None, 1)
output: (None, 1)
school_state: InputLayer
input: (None, 1)
output: (None, 1)
project_grade_category: InputLayer
input:
output:
lstm_1: LSTM
input: (None, 500, 300)
output: (None, 500, 128)
teacher_prefix_emb: Embedding
input: (None, 1)
output: (None, 1, 3)
school_state_emb: Embedding
input: (None, 1)
output: (None, 1, 26)
project_grade_category_emb: Embedding
input:
output:
flatten_1: Flatten
input: (None, 500, 128)
output: (None, 64000)
flatten_2: Flatten
input: (None, 1, 3)
output: (None, 3)
flatten_3: Flatten
input: (None, 1, 26)
output: (None, 26)
flatten_4: Flatten
input: (None, 1, 2
output: (None, 2)
concatenate_1: Concatenate
input: [(None, 64000), (None, 3), (None, 26), (No
output: (None,
dense_2: Dense
input: (None, 6420
output: (None, 256)
dropout_1: Dropout
input: (None, 25
output: (None, 25
dense_3: Dense
input: (None, 256)
output: (None, 128)
dropout_2: Dropout
input: (None, 12
output: (None, 12
dense_4: Dense
input: (None, 128)
output: (None, 64)



Getting all data into list.

```
# Train data
tr_data_1 = [pad_tr, tr_tea_pre_encode, tr_sch_encode, tr_pro_sub_encode, tr_sub_1_encoder, tr_pro_g

# CV data
cv_data_1 = [pad_cv, cv_tea_pre_encode, cv_sch_encode, cv_pro_sub_encode, cv_sub_1_encoder, cv_pro_g

# Test data
test_data_1 = [pad_test, test_tea_pre_encode, test_sch_encode, test_pro_sub_encode, test_sub_1_encod
```

Chaning type of dependent variable (y) to categorical type

```
from keras.utils import np_utils

y_tr_data_1 = np_utils.to_categorical(y_tr)
y_cv_data_1 = np_utils.to_categorical(y_cv)
y_test_data_1 = np_utils.to_categorical(y_test)


y_tr_data_1
```

```
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

AUC-ROC custom function

Source link: https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteris

```
from sklearn.metrics import roc_auc_score

import tensorflow as tf

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


%load_ext tensorboard
```

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

```
# Clear any logs from previous runs
!rm -rf ./logs/
```

Creating Callback with Checkpoint, EarlyStopping and Tensorboard

Source: https://keras.io/callbacks/

```
import keras
from keras.callbacks import TensorBoard, ModelCheckpoint, EarlyStopping

# Saves the model after every epoch
checkpoint_1 = ModelCheckpoint("model_1.h5", monitor = "val_auc", mode = "min",
                               save_best_only = True, verbose = 1)

# Stops training when a monitored quantity has stopped improving.
earlystop_1 = EarlyStopping(monitor = 'val_loss', mode = "min", patience = 5,
                            verbose = 1, restore_best_weights = True)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
# TensorBoard is a visualization tool provided with TensorFlow.
tensorboard_1 = TensorBoard(log_dir = log_dir,
                        histogram_freq = 1, batch_size = 500, write_graph = True,
                        write_grads = False, write_images = False, embeddings_freq = 0,
                        embeddings_layer_names = None, embeddings_metadata = None,
```

```
                          embeddings_data = None, update_freq = 'epoch')

# Creating Callback
callback_1 = [checkpoint_1, earlystop_1, tensorboard_1]
```

Compile the data

- **Optimizer: rmsprop**
- **Dropout - 0.3**
- **Loss: categorical_crossentropy**
- **Metric: AUC-ROC**

```
import warnings
warnings.filterwarnings('ignore')


from keras.optimizers import Adam, RMSprop

model_1.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = [auroc])
```

Fitting model and callback to visualize model

```
try:
  history_1 = model_1.fit(tr_data_1, y_tr_data_1, batch_size = 1000,
                          epochs = 10, validation_data = (cv_data_1, y_cv_data_1), verbose = 1,
                          callbacks = callback_1)
```

```
except ValueError:
    pass
```

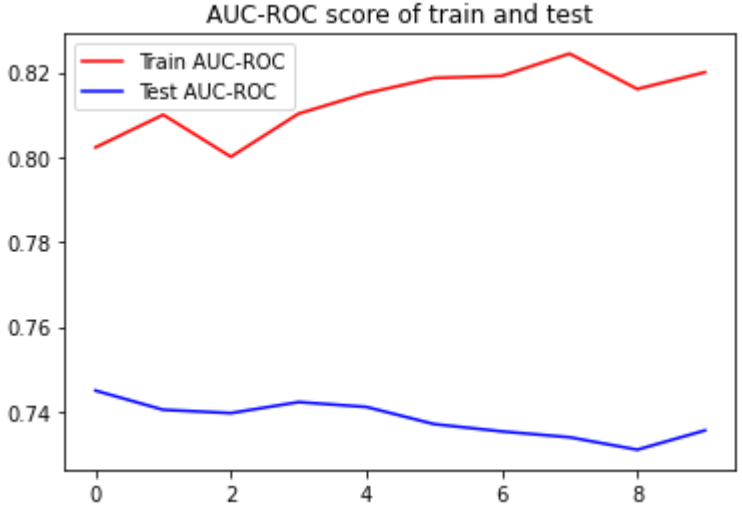
```
Train on 65548 samples, validate on 21850 samples
Epoch 1/10
65548/65548 [==============================] - 132s 2ms/step - loss: 0.4462 - auroc: 0.8023 - v
Epoch 2/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4139 - auroc: 0.8100 - v
Epoch 3/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4170 - auroc: 0.8001 - v
Epoch 4/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4114 - auroc: 0.8102 - v
Epoch 5/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4085 - auroc: 0.8150 - v
Epoch 6/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4054 - auroc: 0.8186 - v
Epoch 7/10
65548/65548 [==============================] - 125s 2ms/step - loss: 0.4039 - auroc: 0.8191 - v
Epoch 8/10
65548/65548 [==============================] - 128s 2ms/step - loss: 0.4010 - auroc: 0.8244 - v
Epoch 9/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4045 - auroc: 0.8160 - v
Epoch 10/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4063 - auroc: 0.8200 - v
Restoring model weights from the end of the best epoch
Epoch 00010: early stopping
```

Evaluating test data

```
# Evaluating test data
```

```
score_1 = model_1.evaluate(test_data_1, y_test_data_1, verbose = 1, batch_size = 512)
print('Test Loss:', score_1[0])
print('Test ROC-AUC score:', score_1[1], '\n')

# Plotting train and test auc roc score
plt.plot(history_1.history['auroc'], 'r')
plt.plot(history_1.history['val_auroc'], 'b')
plt.title("AUC-ROC score of train and test")
plt.legend({'Train AUC-ROC': 'r', 'Test AUC-ROC':'b'})
plt.show()
```

```
21850/21850 [==============================] - 14s 630us/step
Test Loss: 0.4297425600947723
Test ROC-AUC score: 0.7361505031585693
```



Observation:

- Test Loss - 0.449
- Test AUC-ROC - 0.734

```
%tensorboard --logdir logs/fit
```

TensorBoard

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

TOGGLE ALL RUNS

logs/fit

No scalar data w

Probable causes:

- You haven't wr
- TensorBoard c

If you're new to usin
and set up your ever
TensorBoard tutoria

If you think TensorB
the README devote
on GitHub.

---------- Model - 2 ----------

```
x = project_data_1.drop(['project_is_approved'], axis = 1)
y = project_data_1['project_is_approved']


from sklearn.model_selection import train_test_split

# Splitting into x and y into train and test set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 42, strat
```

```
# Splitting train set into tr and cv set
x_tr, x_cv, y_tr, y_cv = train_test_split(x_train, y_train,  test_size = 0.25, random_state = 42, st


print("Shape of x_tr:", x_tr.shape)
print("Shape of x_cv:", x_cv.shape)
print("Shape of x_test:", x_test.shape)
print("Shape of y_tr:", y_tr.shape)
print("Shape of y_cv:", y_cv.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of x_tr: (65548, 11)
Shape of x_cv: (21850, 11)
Shape of x_test: (21850, 11)
Shape of y_tr: (65548,)
Shape of y_cv: (21850,)
Shape of y_test: (21850,)
```

Applying TF-IDF vectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

tf = TfidfVectorizer()

# Fit and transform train data
x_tr_tf = tf.fit_transform(x_tr.essay)
```

```
# Transform cv data
x_cv_tf = tf.transform(x_cv.essay)

# Transform test data
x_te_tf = tf.transform(x_test.essay)
```

Getting IDF values and Feature Names

```
# Let take a look on first 10 idf values

print("First 10 idf values\n")
print(tf.idf_[:10])
```

```
First 10 idf values

[ 7.20020411  5.88397731 10.99194094 11.39740605 11.39740605 11.39740605
 10.48111532 11.39740605  9.78796814 10.48111532]
```

```
# Zipping feature names corresponding to idf_ values

feat_idf = sorted(zip(tf.idf_, tf.get_feature_names()))


print("First 5 feature names along with idf values:\n")

print(feat_idf[:5])

print("\nLast 5 feature names along with idf values:\n")
```

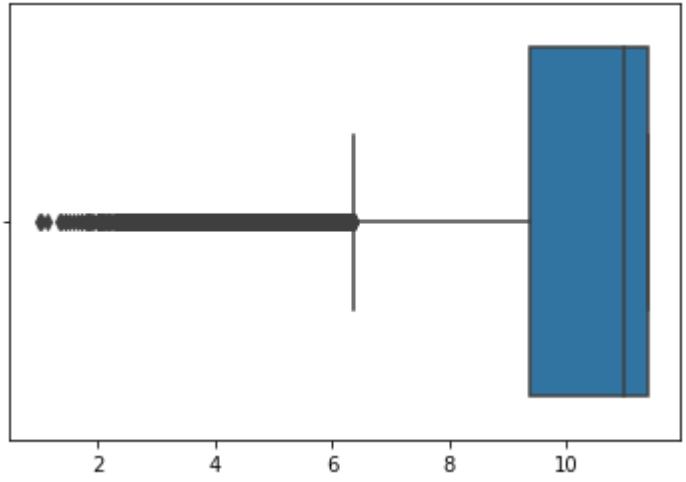
```
print("\nLast 5 feature names along with idf values:\n")

print(feat_idf[-5:])
```

```
First 5 feature names along with idf values:

[(1.0075187748451127, 'students'), (1.0454417967478642, 'nannan'), (1.1627280450519324, 'school

Last 5 feature names along with idf values:

[(11.397406052985405, 'zumwalt'), (11.397406052985405, 'zundel'), (11.397406052985405, 'zx110')
```

Box plot

```
print("Box plot for idf values\n")
sns.boxplot(tf.idf_)
plt.show()
```

```
Box plot for idf values
```



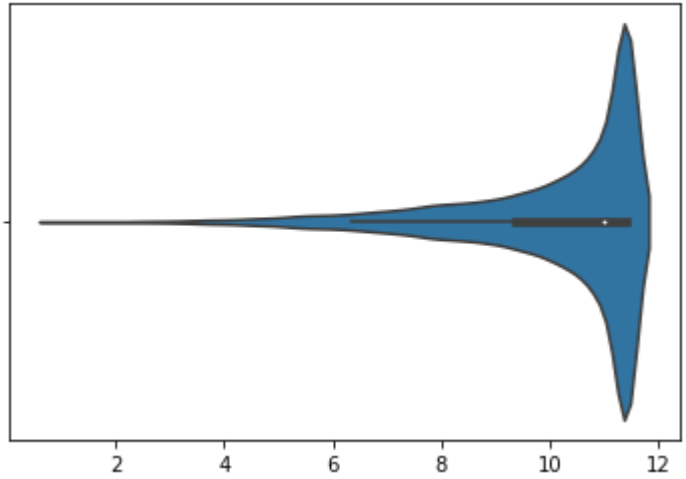
Observation:

- Quartile 1: IDF values ranges from 0 to 9.3.
- Quartile 2: IDF values ranges from 9.4 to 10.99.
- Quartile 3: IDF values ranges from 11 to 11.39.
- Quartile 4" IDF values ranges from 11.39 to 11.399

Violin plot

```
print("Violin plot for idf values\n")
sns.violinplot(tf.idf_)
plt.show()
```

```
Violin plot for idf values
```



Observation:

- Quartile 1: IDF values ranges from 0 to 9.3.
- Quartile 2: IDF values ranges from 9.4 to 10.99.
- Quartile 3: IDF values ranges from 11 to 11.39.
- Quartile 4" IDF values ranges from 11.39 to 11.399

```
sort_idf = sorted(tf.idf_)

print("Mean of idf values:", np.mean(sort_idf))
print("Median of idf values:", np.median(sort_idf))
print("Maximum of idf values:", max(sort_idf))
print("Minimum of idf values:", min(sort_idf))
```

```
Mean of idf values: 10.072181593175548
Median of idf values: 10.99194094487724
Maximum of idf values: 11.397406052985405
Minimum of idf values: 1.0075187748451127
```

```
# Get the IQR (Inter Quartile Range)

q1 = np.percentile(sort_idf, 25)
q3 = np.percentile(sort_idf, 75)

print("Quartile 1 (Q1):", np.percentile(sort_idf, 25))
print("Quartile 2 (Q2):", np.percentile(sort_idf, 50))
print("Quartile 3 (Q3):", np.percentile(sort_idf, 75))
print("Quartile 4 (Q4):", np.percentile(sort_idf, 100))

print("\nInter Quartile Range (Q3 - Q1):\n")
(np.percentile(sort_idf, 75) - np.percentile(sort_idf, 25))
```

```
Quartile 1 (Q1): 9.38250303244314
Quartile 2 (Q2): 10.99194094487724
Quartile 3 (Q3): 11.397406052985405
Quartile 4 (Q4): 11.397406052985405

Inter Quartile Range (Q3 - Q1):

2.014903020542265
```

Getting list of words whose IDF values falls under IQR i.e between Q1 and Q3

```
list_words = []

for i in range(len(feat_idf)):

    if feat_idf[i][0] > 1 and feat_idf[i][0] < 11:
        words = feat_idf[i][1]
        list_words.append(words)


print("Number of words before taking IQR:", len(feat_idf))
print("Number of words after taking IQR:", len(list_words))
```

```
Number of words before taking IQR: 46093
Number of words after taking IQR: 28121
```

Tokenize:

Input data to layer should be integer. So, using tokenize inbuilt function, we will integer encode the text data.

```
from keras.preprocessing.text import Tokenizer

t_2 = Tokenizer(num_words = vocab_size)

# Fit train text data
t_2.fit_on_texts(list_words)

# Sequencing train, cv and test data i.e transforming
tr_seq_2 = t.texts_to_sequences(x_tr['essay'])
cv_seq_2 = t.texts_to_sequences(x_cv['essay'])
test_seq_2 = t.texts_to_sequences(x_test['essay'])
print('Done!')
```

```
Done!
```

Weight Matrix

Let's create a weight matrix of train data from the glove vector.

Source Link: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

```
# Let's create a weight matrix of train data from the glove vector.
from numpy import zeros

word_count_2 = min(vocab_size, len(t_2.word_index) + 1)
```

```
word_count_2 = min(vocab_size, len(t_2.word_index) + 1)

emb_matrix_2 = zeros((word_count_2, emb_dim))
for word, i in t_2.word_index.items():
  emb_vec_2 = embeddings_index.get(word)
  if emb_vec_2 is not None:
    emb_matrix_2[i] = emb_vec_2

print("Number for unique words in train data:", len(t_2.word_index) + 1)
print("Shape of train weight matrix:", emb_matrix_2.shape)
```

```
Number for unique words in train data: 28122
Shape of train weight matrix: (28122, 300)
```

Padding document

Padding document is to have the same input length of each document.

```
from keras.preprocessing.sequence import pad_sequences

pad_tr_2 = pad_sequences(tr_seq_2, maxlen = seq_len, padding = 'post', truncating = 'post')
pad_cv_2 = pad_sequences(cv_seq_2, maxlen = seq_len, padding = 'post', truncating = 'post')
pad_test_2 = pad_sequences(test_seq_2, maxlen = seq_len, padding = 'post', truncating = 'post')

print("Shape of pad_tr:", pad_tr_2.shape)
print("Shape of pad_cv:", pad_cv_2.shape)
print("Shape of pad_test:", pad_test_2.shape)
```

```
Shape of pad_tr: (65548, 500)
Shape of pad_cv: (21850, 500)
Shape of pad_test: (21850, 500)
```

Embedding layer for text data

```
from keras.layers import Embedding, Dense, Flatten, Input, LSTM, Dropout, BatchNormalization, concat


input_size_2 = min(vocab_size, len(t_2.word_index) + 1)

# Creating an input layer
input_lay_2 = Input(shape = (seq_len, ), name = "Input_Text_Data")

# Creating an embedding layer
emb_lay_2 = Embedding(input_dim = input_size_2, output_dim = emb_dim,
                          input_length = seq_len, weights = [emb_matrix_2],
                          trainable = False, name = "lstm_text_layer")(input_lay_2)

# Creating LSTM layer
emb_lay_text_2 = LSTM(128, return_sequences = True, dropout = 0.3)(emb_lay_2)

flatten_1_2 = Flatten()(emb_lay_text_2)
```

Concatenating all the flattened layers

```
from keras.layers import concatenate

con_lay_2 = concatenate([flatten_1_2, flatten_tea_pre, flatten_sch, flatten_pro_gra, flatten_pro_sub
```

Keras model:

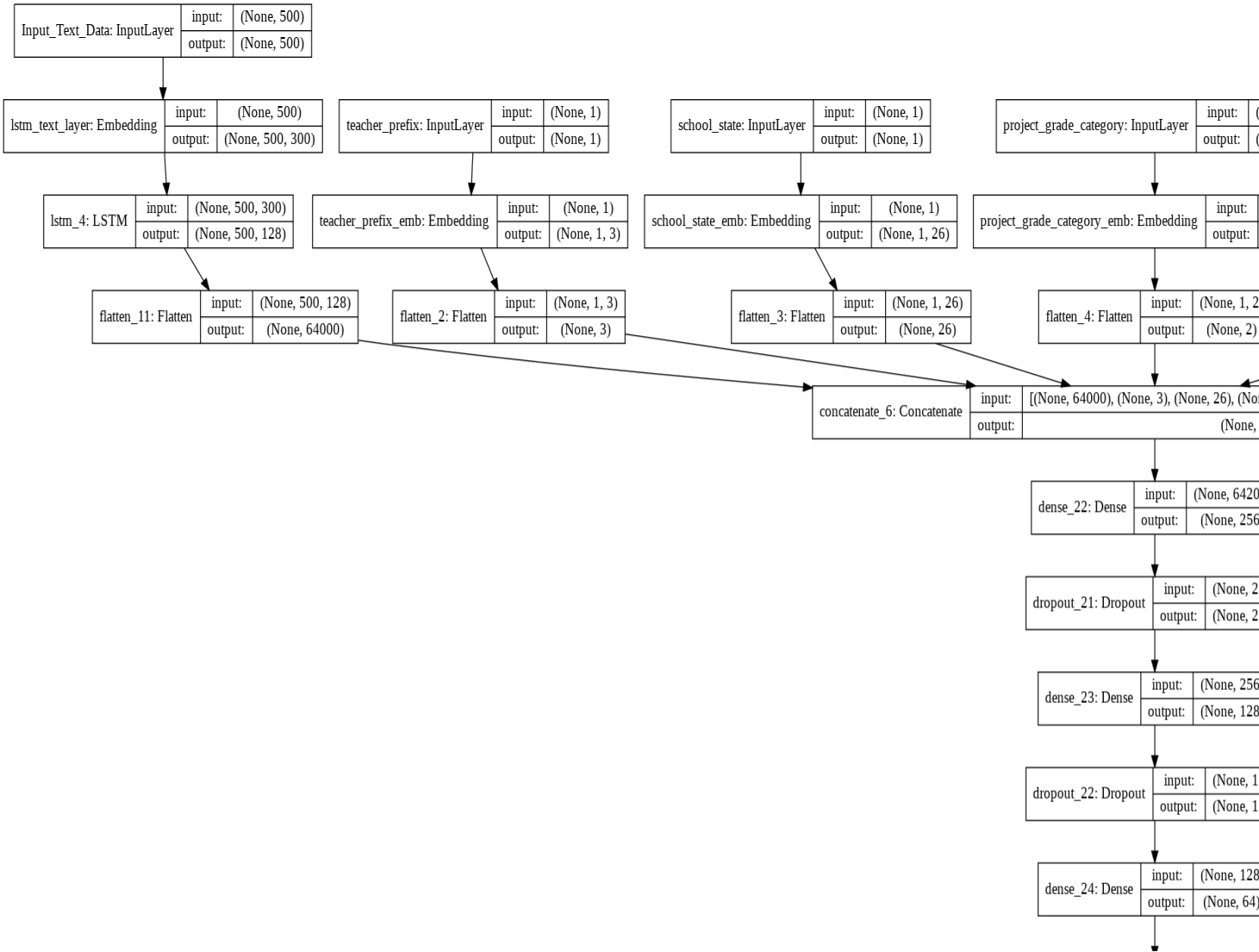
- Activation - 'relu' and 'softmax'.
- Dropout - 0.3
- kernel_regularizer - regularizers.l2(0.01)

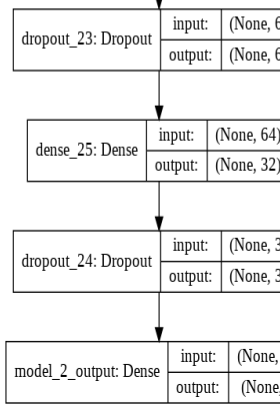
```
from keras import regularizers, initializers

# Layer 1
m_2 = Dense(256, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(con_lay_2)
m_2 = Dropout(0.3)(m_2)

# Layer 2
m_2 = Dense(128, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m_2)
m_2 = Dropout(0.3)(m_2)

# Layer 3
m_2 = Dense(64, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m_2)
m_2 = Dropout(0.3)(m_2)

# Layer 3
m_2 = Dense(32, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m_2)
```

```
m_2 = Dropout(0.3)(m_2)

# Output layer
output_2 = Dense(2, activation = 'softmax', name = 'model_2_output')(m_2)

# Model
model_2 = Model(inputs = [input_lay_2, inp_tea_pre, inp_sch, inp_pro_gra,
                          inp_pro_sub, inp_pro_sub_1, inp_num], outputs = [output_2])
```

Network Architecture

```
# https://github.com/mmortazavi/EntityEmbedding-Working_Example/blob/master/EntityEmbedding.ipynb

import pydot_ng as pydot
from keras.utils import plot_model
from IPython.display import Image

plot_model(model_2, show_shapes = True, show_layer_names = True, to_file = 'model_2.png')

Image(retina = True, filename = 'model_2.png')
```


Input_Text_Data: InputLayer
input: (None, 500)
output: (None, 500)
lstm_text_layer: Embedding
input: (None, 500)
output: (None, 500, 300)
teacher_prefix: InputLayer
input: (None, 1)
output: (None, 1)
school_state: InputLayer
input: (None, 1)
output: (None, 1)
project_grade_category: InputLayer
input:
output:
lstm_4: LSTM
input: (None, 500, 300)
output: (None, 500, 128)
teacher_prefix_emb: Embedding
input: (None, 1)
output: (None, 1, 3)
school_state_emb: Embedding
input: (None, 1)
output: (None, 1, 26)
project_grade_category_emb: Embedding
input:
output:
flatten_11: Flatten
input: (None, 500, 128)
output: (None, 64000)
flatten_2: Flatten
input: (None, 1, 3)
output: (None, 3)
flatten_3: Flatten
input: (None, 1, 26)
output: (None, 26)
flatten_4: Flatten
input: (None, 1, 2
output: (None, 2)
concatenate_6: Concatenate
input: [(None, 64000), (None, 3), (None, 26), (No
output: (None,
dense_22: Dense
input: (None, 6420
output: (None, 256
dropout_21: Dropout
input: (None, 2
output: (None, 2
dense_23: Dense
input: (None, 256
output: (None, 128
dropout_22: Dropout
input: (None, 1
output: (None, 1
dense_24: Dense
input: (None, 128
output: (None, 64)



Getting all data into list.

```
# Train data
tr_data_2 = [pad_tr_2, tr_tea_pre_encode, tr_sch_encode, tr_pro_sub_encode, tr_sub_1_encoder, tr_pro

# CV data
cv_data_2 = [pad_cv_2, cv_tea_pre_encode, cv_sch_encode, cv_pro_sub_encode, cv_sub_1_encoder, cv_pro

# Test data
test_data_2 = [pad_test_2, test_tea_pre_encode, test_sch_encode, test_pro_sub_encode, test_sub_1_enc


# Chaining type of dependent variable (y) to categorical type
from keras.utils import np_utils

y_tr_data_2 = np_utils.to_categorical(y_tr_2)
```

```
y_tr_data_2 = np_utils.to_categorical(y_tr,2)
y_cv_data_2 = np_utils.to_categorical(y_cv,2)
y_test_data_2 = np_utils.to_categorical(y_test,2)
```

Creating Callback with Checkpoint, EarlyStopping and Tensorboard

Source: https://keras.io/callbacks//

```
%load_ext tensorboard
```

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

```
# Clear any logs from previous runs
!rm -rf ./logs/
```

```
import keras
from keras.callbacks import TensorBoard, ModelCheckpoint, EarlyStopping

# Saves the model after every epoch
checkpoint_2 = ModelCheckpoint("model_2.h5", monitor = "val_loss", mode = "min",
                               save_best_only = True, verbose = 1)

# Stops training when a monitored quantity has stopped improving.
earlystop_2 = EarlyStopping(monitor = 'val_auc', mode = "min", patience = 5,
                            verbose = 1, restore_best_weights = True)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
# TensorBoard is a visualization tool provided with TensorFlow.
```

```
# TensorBoard is a visualization tool provided with TensorFlow.
tensorboard_2 = TensorBoard(log_dir = log_dir,
                            histogram_freq = 0, batch_size = 500, write_graph = True,
                            write_grads = False, write_images = False, embeddings_freq = 0,
                            embeddings_layer_names = None, embeddings_metadata = None,
                            embeddings_data = None, update_freq = 'epoch')

# Creating Callback
callback_2 = [checkpoint_2, earlystop_2, tensorboard_2]
```

Compile the data

- **Optimizer: rmsprop**
- **Dropout - 0.3**
- **Loss: categorical_crossentropy**
- **Metric: AUC-ROC**

```
import warnings
warnings.filterwarnings('ignore')


from sklearn.metrics import roc_auc_score

import tensorflow as tf

def auroc_2(y_true, y_pred):

    return tf.py_function(roc_auc_score, (y_true, y_pred), tf.double)
```

```
from keras.optimizers import Adam, RMSprop

model_2.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = [auroc_2])
```

Fitting model and callback to visualize model

```
try:
  history_2 = model_2.fit(tr_data_2, y_tr_data_2, batch_size = 1000,
                          epochs = 10, validation_data = (cv_data_2, y_cv_data_2), verbose = 1,
                          callbacks = callback_2)
except ValueError:
    pass
```

```
Train on 65548 samples, validate on 21850 samples
Epoch 1/10
65548/65548 [==============================] - 132s 2ms/step - loss: 0.4652 - auroc_2: 0.7097 -

Epoch 00001: val_loss improved from inf to 0.48266, saving model to model_2.h5
Epoch 2/10
65548/65548 [==============================] - 128s 2ms/step - loss: 0.4489 - auroc_2: 0.7248 -

Epoch 00002: val_loss improved from 0.48266 to 0.44986, saving model to model_2.h5
Epoch 3/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4482 - auroc_2: 0.7286 -

Epoch 00003: val_loss did not improve from 0.44986
Epoch 4/10
65548/65548 [==============================] - 128s 2ms/step - loss: 0.4475 - auroc_2: 0.7330 -

Epoch 00004: val_loss improved from 0.44986 to 0.44407, saving model to model_2.h5
Epoch 5/10
65548/65548 [==============================] - 128s 2ms/step - loss: 0.4420 - auroc_2: 0.7431 -

Epoch 00005: val_loss did not improve from 0.44407
Epoch 6/10
65548/65548 [==============================] - 128s 2ms/step - loss: 0.4419 - auroc_2: 0.7478 -

Epoch 00006: val_loss did not improve from 0.44407
Epoch 7/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4382 - auroc_2: 0.7508 -

Epoch 00007: val_loss did not improve from 0.44407
Epoch 8/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4383 - auroc_2: 0.7561 -

Epoch 00008: val_loss did not improve from 0.44407
```

```
Epoch 00008: val_loss did not improve from 0.44407
Epoch 9/10
65548/65548 [==============================] - 127s 2ms/step - loss: 0.4331 - auroc_2: 0.7656 -

Epoch 00009: val_loss did not improve from 0.44407
Epoch 10/10
65548/65548 [==============================] - 126s 2ms/step - loss: 0.4336 - auroc_2: 0.7665 -

Epoch 00010: val_loss did not improve from 0.44407
```

```
%tensorboard --logdir logs/fit
```

```
Reusing TensorBoard on port 6009 (pid 5269), started 0:26:14 ago. (Use '!kill 5269' to kill it.
```

TensorBoard

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

TOGGLE ALL RUNS

logs/fit

No scalar data w

Probable causes:

- You haven't wr
- TensorBoard c

If you're new to usin
and set up your ever
TensorBoard tutoria

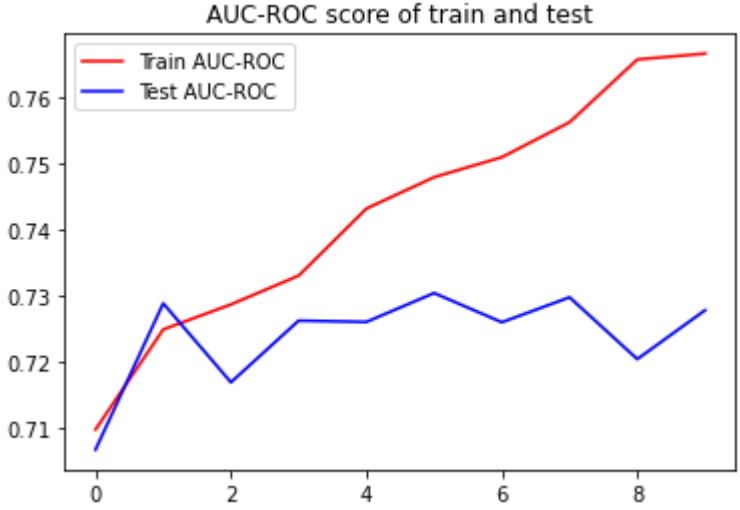
If you think TensorB
the README devote
on GitHub.

Evaluating test data

```
# Evaluating test data
score_2 = model_2.evaluate(test_data_2, y_test_data_2, verbose = 1, batch_size = 512)
print('Test Loss:', score_2[0])
print('Test ROC-AUC score:', score_2[1], '\n')

# Plotting train and test auc roc score
plt.plot(history_2.history['auroc_2'], 'r')
plt.plot(history_2.history['val_auroc_2'], 'b')
plt.title("AUC-ROC score of train and test")
plt.legend({'Train AUC-ROC':'r', 'Test AUC-ROC':'b'})
```

```
plt.legend({'Train AUC-ROC' : 'r', 'Test AUC-ROC' : 'b'})
plt.show()
```

```
21850/21850 [==============================] - 14s 625us/step
Test Loss: 0.47520854377091887
Test ROC-AUC score: 0.7179731130599976
```



Observation:

- Test loss - 0.475
- Test AUC-ROC - 0.717

---------- Model - 3 ----------

```
%load_ext tensorboard


# Clear any logs from previous runs
!rm -rf ./logs/


from sklearn.model_selection import train_test_split

# Splitting into x and y into train and test set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42, strati

# Splitting train set into tr and cv set
x_tr, x_cv, y_tr, y_cv = train_test_split(x_train, y_train, test_size = 0.25, random_state = 42, str


print("Shape of x_tr:", x_tr.shape)
print("Shape of x_cv:", x_cv.shape)
print("Shape of x_test:", x_test.shape)
print("Shape of y_tr:", y_tr.shape)
print("Shape of y_cv:", y_cv.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of x_tr: (65548, 11)
Shape of x_cv: (21850, 11)
Shape of x_test: (21850, 11)
Shape of y_tr: (65548,)
Shape of y_cv: (21850,)
Shape of y_test: (21850,)
```

```
# Train
df_cn_tr = pd.DataFrame()

df_cn_tr['tea_pre'] = tr_tea_pre_encode
df_cn_tr['sch'] = tr_sch_encode
df_cn_tr['pro_sub'] = tr_pro_sub_encode
df_cn_tr['sub_1'] = tr_sub_1_encoder
df_cn_tr['pro_gra'] = tr_pro_gra_encode
df_cn_tr['pri'] = tr_1
df_cn_tr['qua'] = tr_2
df_cn_tr['pro_sum'] = tr_3
df_cn_tr['tea_sum'] = tr_4

# CV
df_cn_cv = pd.DataFrame()

df_cn_cv['tea_pre'] = cv_tea_pre_encode
df_cn_cv['sch'] = cv_sch_encode
df_cn_cv['pro_sub'] = cv_pro_sub_encode
df_cn_cv['sub_1'] = cv_sub_1_encoder
df_cn_cv['pro_gra'] = cv_pro_gra_encode
```

```
df_cn_cv['pri'] = cv_1
df_cn_cv['qua'] = cv_2
df_cn_cv['pro_sum'] = cv_3
df_cn_cv['tea_sum'] = cv_4

# Test
df_cn_te = pd.DataFrame()

df_cn_te['tea_pre'] = test_tea_pre_encode
df_cn_te['sch'] = test_sch_encode
df_cn_te['pro_sub'] = test_pro_sub_encode
df_cn_te['sub_1'] = test_sub_1_encoder
df_cn_te['pro_gra'] = test_pro_gra_encode
df_cn_te['pri'] = test_1
df_cn_te['qua'] = test_2
df_cn_te['pro_sum'] = test_3
df_cn_te['tea_sum'] = test_4



tr_exp = np.expand_dims(df_cn_tr, 2)
cv_exp = np.expand_dims(df_cn_cv, 2)
te_exp = np.expand_dims(df_cn_te, 2)
```

Getting all data into a list

```
# Concatenating padded data and expanded data
```

```
# Concatenating padded data and expanded data.

tr_data_3 = [pad_tr, tr_exp]
cv_data_3 = [pad_cv, cv_exp]
te_data_3 = [pad_test, te_exp]


# Chaning type of dependent variable (y) to categorical type
from keras.utils import np_utils

y_tr_data_3 = np_utils.to_categorical(y_tr, 2)
y_cv_data_3 = np_utils.to_categorical(y_cv, 2)
y_test_data_3 = np_utils.to_categorical(y_test, 2)
```

Convolution 1D

- Layers - 4
- Kernel size - 3
- Activation - 'relu' and 'softmax'
- Padding - same

```
from keras.layers import Dense, Dropout, Flatten, Conv1D, MaxPooling1D, Activation,Input

# Input layer
inp_lay_1 = Input(shape = (9,1), name = "Conv1")

# Block 1
con1 = Conv1D(64, kernel_size = 3, activation = 'relu', name = 'block_1')(inp_lay_1)
```

```
con1 = Conv1D(64, kernel_size = 3, activation = 'relu', name = 'block_1')(inp_lay_1)

# Block 2
con2 = Conv1D(64, 3, activation='relu', padding = 'same', name = 'block_2')(con1)

# Block 3
con3 = Conv1D(32, 3, activation='softmax', padding = 'same',  name = 'block_3')(con2)

# Block 4
con4 = Conv1D(32, 3, activation='softmax', padding = 'same',  name = 'block_4')(con3)

# Flattening
flat1 = Flatten()(con4)
```

Concatenating LSTM output and Conv1D output

```
from keras.layers import concatenate

con_lay_3 = concatenate([flatten_1, flat1])
```

Keras model:

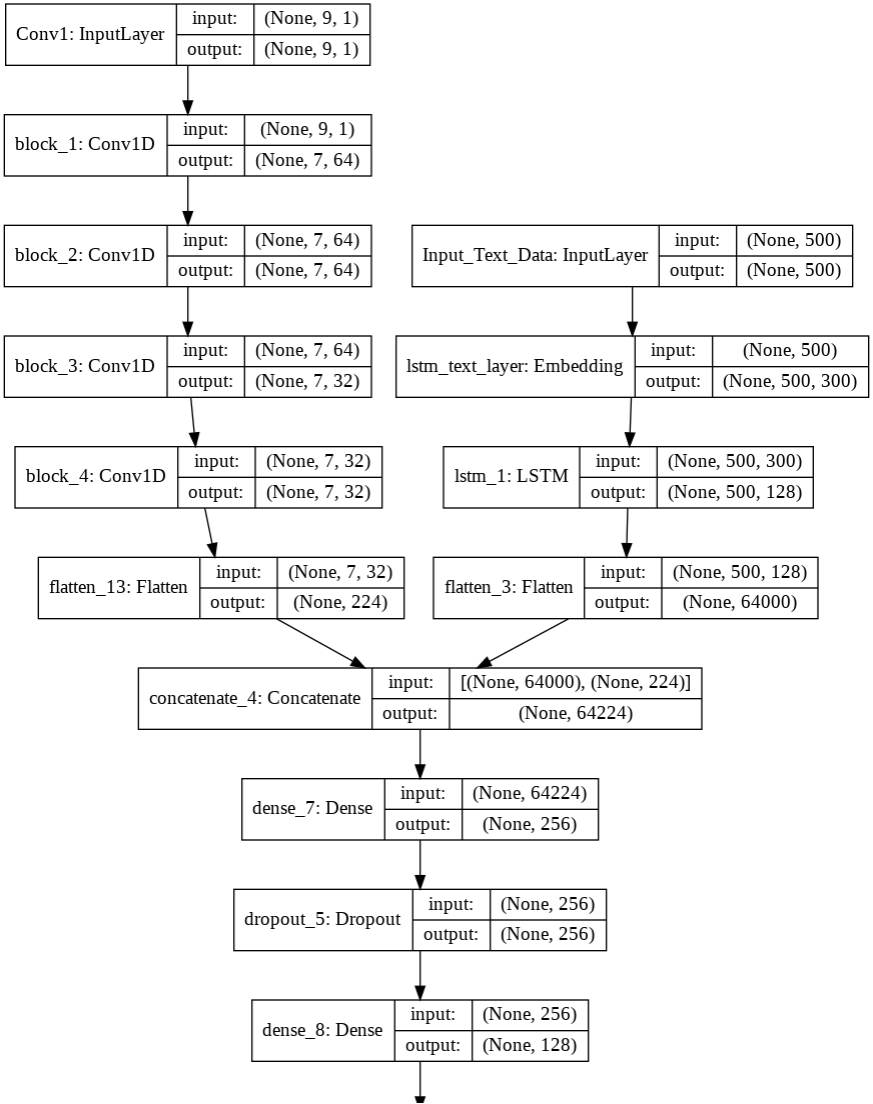
- Activation - 'relu' and 'softmax'.
- Dropout - 0.3
- kernel_regularizer - regularizers.l2(0.01)

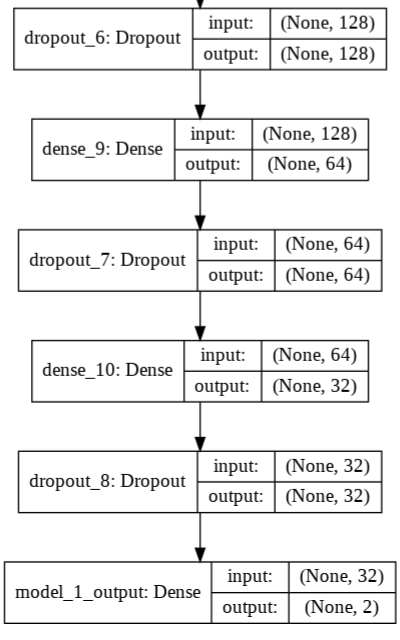
```
from keras.models import Model

# Layer 1
m_3 = Dense(256, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(con_lay_3)
m_3 = Dropout(0.3)(m_3)

# Layer 2
m_3 = Dense(128, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m_3)
m_3 = Dropout(0.3)(m_3)

# Layer 3
m_3 = Dense(64, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m_3)
m_3 = Dropout(0.3)(m_3)

# Layer 4
m_3 = Dense(32, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(m_3)
m_3 = Dropout(0.3)(m_3)

# Output layer
output_3 = Dense(2, activation = 'softmax', name= 'model_1_output')(m_3)

# Model
model_3 = Model(inputs = [input_lay, inp_lay_1], outputs = output_3)
```

Network Architecture

```
# https://github.com/mmortazavi/EntityEmbedding-Working_Example/blob/master/EntityEmbedding.ipynb
```

```
import pydot_ng as pydot
from keras.utils import plot_model
from IPython.display import Image

plot_model(model_3, show_shapes = True, show_layer_names = True, to_file = 'model_3.png')

Image(retina = True, filename = 'model_3.png')
```


Conv1: InputLayer
input: (None, 9, 1)
output: (None, 9, 1)
block_1: Conv1D
input: (None, 9, 1)
output: (None, 7, 64)
block_2: Conv1D
input: (None, 7, 64)
output: (None, 7, 64)
block_3: Conv1D
input: (None, 7, 64)
output: (None, 7, 32)
block_4: Conv1D
input: (None, 7, 32)
output: (None, 7, 32)
flatten_13: Flatten
input: (None, 7, 32)
output: (None, 224)
Input_Text_Data: InputLayer
input: (None, 500)
output: (None, 500)
lstm_text_layer: Embedding
input: (None, 500)
output: (None, 500, 300)
lstm_1: LSTM
input: (None, 500, 300)
output: (None, 500, 128)
flatten_3: Flatten
input: (None, 500, 128)
output: (None, 64000)
concatenate_4: Concatenate
input: [(None, 64000), (None, 224)]
output: (None, 64224)
dense_7: Dense
input: (None, 64224)
output: (None, 256)
dropout_5: Dropout
input: (None, 256)
output: (None, 256)
dense_8: Dense
input: (None, 256)
output: (None, 128)



Creating Callback with Checkpoint, EarlyStopping and Tensorboard

Source: https://keras.io/callbacks/

```
%load_ext tensorboard
```

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

```
import keras
from keras.callbacks import TensorBoard, ModelCheckpoint, EarlyStopping
import datetime
# Saves the model after every epoch
checkpoint_3 = ModelCheckpoint("model_3.h5", monitor = "val_loss", mode = "min",
                               save_best_only = True, verbose = 1)

# Stops training when a monitored quantity has stopped improving.
earlystop_3 = EarlyStopping(monitor = 'val_auc', mode = "min", patience = 5,
                            verbose = 1, restore_best_weights = True)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
# TensorBoard is a visualization tool provided with TensorFlow.
tensorboard_3 = TensorBoard(log_dir = log_dir,
                         histogram_freq = 0, batch_size = 500, write_graph = True,
                         write_grads = False, write_images = False, embeddings_freq = 0,
                         embeddings_layer_names = None, embeddings_metadata = None,
                         embeddings_data = None, update_freq = 'epoch')

# Creating Callback
callback_3 = [checkpoint_3, earlystop_3, tensorboard_3]
```

Compile the data

- **Optimizer: rmsprop**

- **Dropout - 0.3**
- **Loss: categorical_crossentropy**
- **Metric: AUC-ROC**

```
import warnings
warnings.filterwarnings('ignore')


from sklearn.metrics import roc_auc_score

import tensorflow as tf

def auroc_3(y_true, y_pred):
    return tf.py_function(roc_auc_score, (y_true, y_pred), tf.double)




from keras.optimizers import Adam, RMSprop

model_3.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = [auroc_3])
```

Fitting model and callback to visualize model

```
try:
  history_3 = model_3.fit(tr_data_3, y_tr_data_3, batch_size = 1000,epochs = 10, validation_data = (

except ValueError:
  pass
```

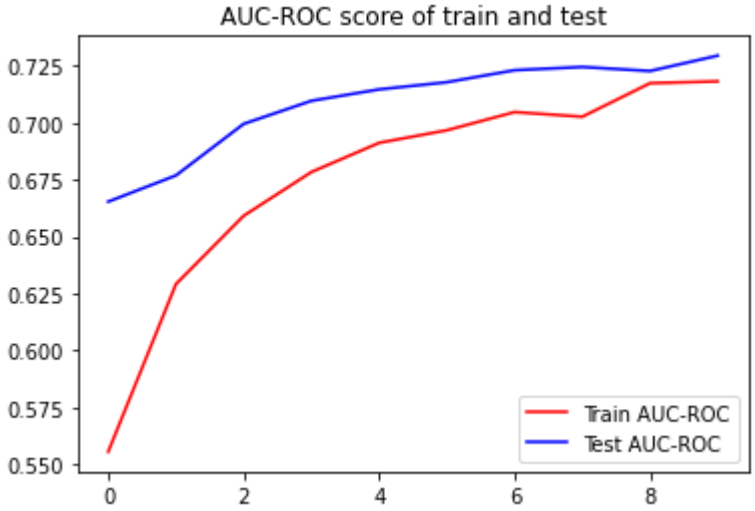
```
Train on 65548 samples, validate on 21850 samples
Epoch 1/10
65548/65548 [==============================] - 124s 2ms/step - loss: 2.4327 - auroc_3: 0.5556 -

Epoch 00001: val_loss improved from inf to 1.41554, saving model to model_3.h5
Epoch 2/10
65548/65548 [==============================] - 119s 2ms/step - loss: 0.9246 - auroc_3: 0.6292 -

Epoch 00002: val_loss improved from 1.41554 to 0.68510, saving model to model_3.h5
Epoch 3/10
65548/65548 [==============================] - 119s 2ms/step - loss: 0.5734 - auroc_3: 0.6592 -

Epoch 00003: val_loss improved from 0.68510 to 0.50942, saving model to model_3.h5
Epoch 4/10
65548/65548 [==============================] - 120s 2ms/step - loss: 0.4955 - auroc_3: 0.6784 -

Epoch 00004: val_loss improved from 0.50942 to 0.46656, saving model to model_3.h5
Epoch 5/10
65548/65548 [==============================] - 118s 2ms/step - loss: 0.4711 - auroc_3: 0.6912 -

Epoch 00005: val_loss improved from 0.46656 to 0.45281, saving model to model_3.h5
Epoch 6/10
65548/65548 [==============================] - 119s 2ms/step - loss: 0.4611 - auroc_3: 0.6967 -

Epoch 00006: val_loss improved from 0.45281 to 0.44591, saving model to model_3.h5
Epoch 7/10
65548/65548 [==============================] - 119s 2ms/step - loss: 0.4550 - auroc_3: 0.7047 -

Epoch 00007: val_loss did not improve from 0.44591
Epoch 8/10
65548/65548 [==============================] - 119s 2ms/step - loss: 0.4567 - auroc_3: 0.7026 -

Epoch 00008: val_loss improved from 0.44591 to 0.44441, saving model to model_3.h5
```

```
Epoch 00008: val_loss improved from 0.44591 to 0.44441, saving model to model_3.h5
Epoch 9/10
65548/65548 [==============================] - 119s 2ms/step - loss: 0.4490 - auroc_3: 0.7174 -

Epoch 00009: val_loss improved from 0.44441 to 0.44402, saving model to model_3.h5
Epoch 10/10
65548/65548 [==============================] - 118s 2ms/step - loss: 0.4470 - auroc_3: 0.7182 -

Epoch 00010: val_loss improved from 0.44402 to 0.44052, saving model to model_3.h5
```

Evaluating test data

```
# Evaluating test data
score_3 = model_3.evaluate(te_data_3, y_test_data_3, verbose = 1, batch_size = 512)
print('Test Loss:', score_3[0])
print('Test ROC-AUC score:', score_3[1], '\n')

# Plotting train and test auc roc score
plt.plot(history_3.history['auroc_3'], 'r')
plt.plot(history_3.history['val_auroc_3'], 'b')
plt.title("AUC-ROC score of train and test")
plt.legend({'Train AUC-ROC': 'r', 'Test AUC-ROC':'b'})
plt.show()
```

```
21850/21850 [==============================] - 14s 630us/step
Test Loss: 0.43627406643511774
Test ROC-AUC score: 0.7413306832313538
```



Observation:

- Test loss - 0.436
- Test AUC-ROC - 0.7413

```
%load_ext tensorboard
```

```
%load_ext tensorboard
```

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

```
%tensorboard --logdir logs/fit
```

```
Reusing TensorBoard on port 6006 (pid 4344), started 2:25:38 ago. (Use '!kill 4344' to kill it.
```

TensorBoard SCALARS GRAPHS

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

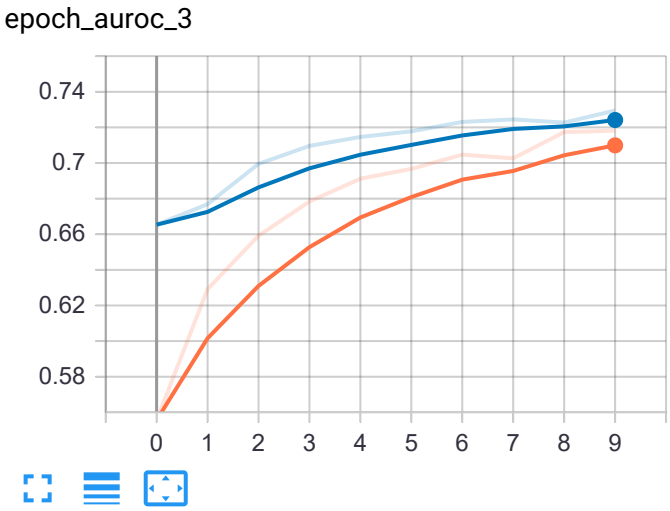
Runs

Write a regex to filter runs

20200423-161914/train

20200423-161914/validation

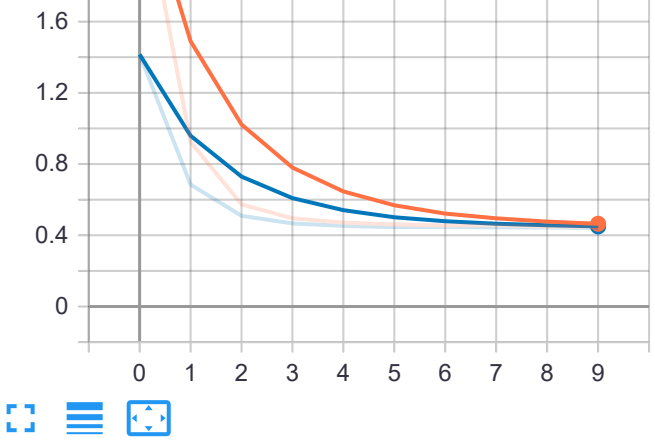
epoch_auroc_3



epoch_loss

TOGGLE ALL RUNS

logs/fit



Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.