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	De
project_id	A unique identifier for the proposed project. **Example:**
project_title	Title of the project. **E** • Art Will Make You • First Gr
project_grade_category	Grade level of students for which the project is targeted. One of the enumerate • Grades • Gra • Gra • Grad
project_subject_categories	One or more (comma-separated) subject categories for the projec following enumerated list • Applied L • Care & • Health & • History & • Literacy & L • Math & • Music & T • Specia • **E** • Music & T • Literacy & Language, Math &
school_state	State where school is located (Two-letter U.S. p (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Posta **Exar**
project_subject_subcategories	One or more (comma-separated) subject subcategories for th **E** • L • Literature & Writing, Social S
project_resource_summary	An explanation of the resources needed for the project. **E** • My students need hands on literacy materials to sensory
project_essay_1	First applicat
project_essay_2	Second applicat
project_essay_3	Third applicat
project_essay_4	Fourth applicat
project_submitted_datetime	Datetime when project application was submitted. **Example:** 201 12:43
teacher_id	A unique identifier for the teacher of the proposed project. bdf8baa8fedef6bfeec7ae4ff

Feature	De
`teacher_prefix`	Teacher's title. One of the following enumerate • • • • • • T
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the sam **Exa**

* 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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

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
```

1. Reading Data

In [2]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?cli
ent_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuserconten
t.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20http
s%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.google
apis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.
photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.read
only&response_type=code

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

In [0]:

```
project_data = pd.read_csv('drive/My Drive/LSTM on Donors/train_data.csv')

resource_data = pd.read_csv('drive/My Drive/LSTM on Donors/resources.csv')
```

In [0]:

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

```
Number of data points in train data (5000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix'
'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [0]:

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

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

Out[0]:

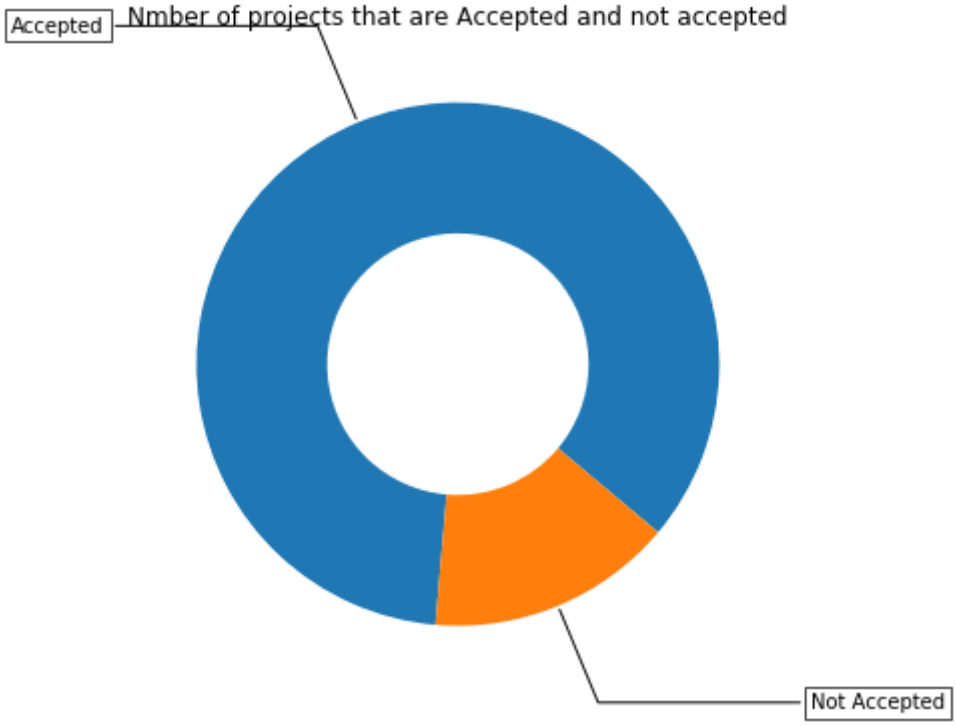
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

Data Analysis

In [0]:

```
# this code is taken from
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-gl
r-gallery-pie-and-polar-charts-pie-and-donut-labels-py


y_value_counts = project_data['project_is_approved'].value_counts()
print("Number of projects thar are approved for funding ", y_value_counts[1], ", (", (y
_value_counts[1]/(y_value_counts[1]+y_value_counts[0]))*100,"%)")
print("Number of projects thar are not approved for funding ", y_value_counts[0], ", ("
, (y_value_counts[0]/(y_value_counts[1]+y_value_counts[0]))*100,"%)")

fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(aspect="equal"))
recipe = ["Accepted", "Not Accepted"]

data = [y_value_counts[1], y_value_counts[0]]

wedges, texts = ax.pie(data, wedgeprops=dict(width=0.5), startangle=-40)

bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
kw = dict(xycoords='data', textcoords='data', arrowprops=dict(arrowstyle="-"),
          bbox=bbox_props, zorder=0, va="center")

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                horizontalalignment=horizontalalignment, **kw)

ax.set_title("Nmber of projects that are Accepted and not accepted")

plt.show()
```

```
Number of projects thar are approved for funding  4237 , ( 84.740000000000
01 %)
Number of projects thar are not approved for funding  763 , ( 15.260000000
000002 %)
```



1.2.1 Univariate Analysis: School State

In [0]:

```
# Pandas dataframe grouby count, mean: https://stackoverflow.com/a/19385591/4084039

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].apply(n
p.mean)).reset_index()
# if you have data which contain only 0 and 1, then the mean = percentage (think about
 it)
temp.columns = ['state_code', 'num_proposals']

# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)'
]]

data = [ dict(
        type='choropleth',
        colorscale = scl,
        autocolorscale = False,
        locations = temp['state_code'],
        z = temp['num_proposals'].astype(float),
        locationmode = 'USA-states',
        text = temp['state_code'],
        marker = dict(line = dict (color = 'rgb(255,255,255)',width = 2)),
        colorbar = dict(title = "% of pro")
    ) ]

layout = dict(
        title = 'Project Proposals % of Acceptance Rate by US States',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)',
        ),
    )

fig = go.Figure(data=data, layout=layout)
# offline.iplot(fig, filename='us-map-heat-map')
```

In [0]:

```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.pdf
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
print('='*50)
print("States with highest % approvals")
print(temp.tail(5))
```

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.500000
41         SD       0.687500
7          DC       0.695652
0          AK       0.705882
50         WY       0.777778
==================================================
States with highest % approvals
   state_code  num_proposals
11         HI       0.964286
16         KS       0.966667
28         ND       1.000000
8          DE       1.000000
30         NH       1.000000
```

In [0]:

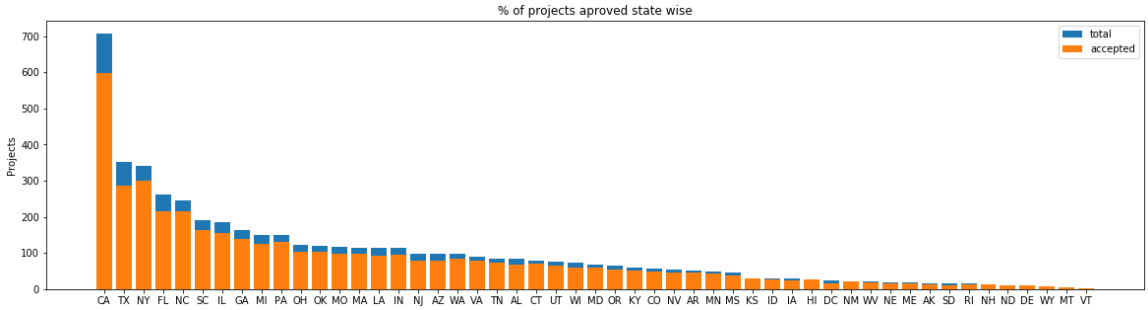
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    plt.figure(figsize=(20,5))
    p1 = plt.bar(ind, data[col3].values)
    p2 = plt.bar(ind, data[col2].values)

    plt.ylabel('Projects')
    plt.title('% of projects aproved state wise')
    plt.xticks(ind, list(data[xtick].values))
    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()
```

In [0]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/4084039
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).reset_index()

    # Pandas dataframe grouby count: https://stackoverflow.com/a/19385591/4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'total':'count'})).reset_index()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':'mean'})).reset_index()['Avg']

    temp.sort_values(by=['total'],inplace=True, ascending=False)

    if top:
        temp = temp[0:top]

    stack_plot(temp, xtick=col1, col2=col2, col3='total')
    print(temp.head(5))
    print("="*50)
    print(temp.tail(5))
```

In [0]:

```
univariate_barplots(project_data, 'school_state', 'project_is_approved', False)
```



	school_state	project_is_approved	total	Avg
4	CA	597	707	0.844413
43	TX	286	352	0.812500
34	NY	299	342	0.874269
9	FL	215	261	0.823755
27	NC	216	246	0.878049

==========

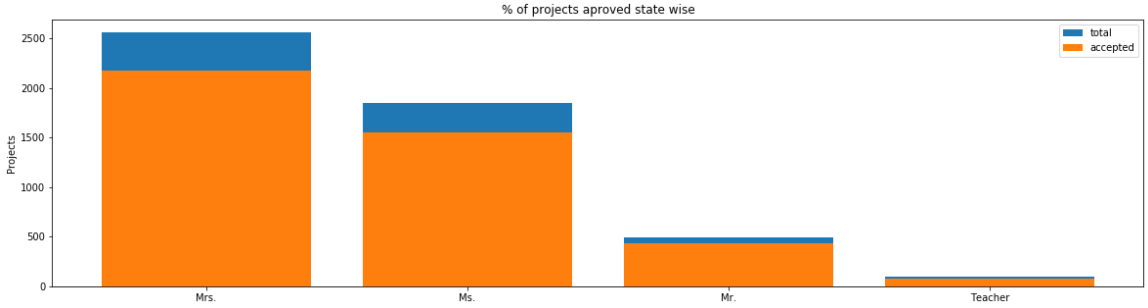
	school_state	project_is_approved	total	Avg
28	ND	11	11	1.000000
8	DE	11	11	1.000000
50	WY	7	9	0.777778
26	MT	5	6	0.833333
46	VT	1	2	0.500000

Every state is having more than 80% success rate in approval

Univariate Analysis: teacher_prefix

In [0]:

```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)
```

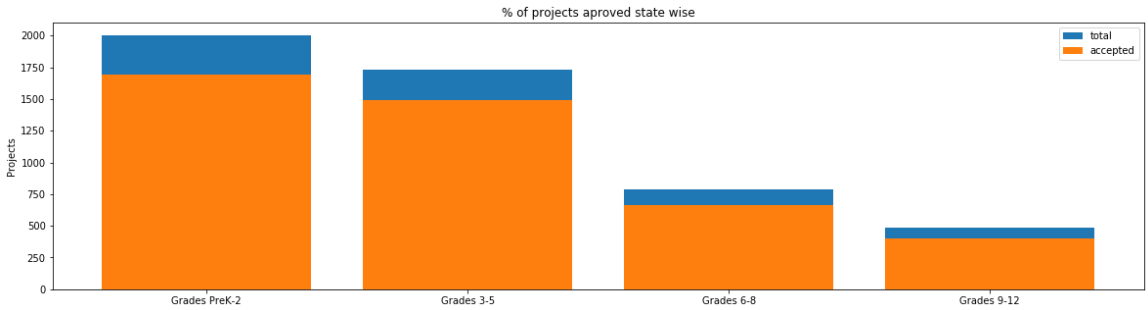



```
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 2173   2560  0.848828
2            Ms.                 1554   1845  0.842276
0            Mr.                  433    495  0.874747
3        Teacher                   77    100  0.770000
==================================================
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 2173   2560  0.848828
2            Ms.                 1554   1845  0.842276
0            Mr.                  433    495  0.874747
3        Teacher                   77    100  0.770000
```

Univariate Analysis: project_grade_category

In [0]:

```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', top=
False)
```




```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 1689   2002  0.843656
0             Grades 3-5                 1491   1729  0.862348
1             Grades 6-8                  660    785  0.840764
2            Grades 9-12                  397    484  0.820248
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 1689   2002  0.843656
0             Grades 3-5                 1491   1729  0.862348
1             Grades 6-8                  660    785  0.840764
2            Grades 9-12                  397    484  0.820248
```

Univariate Analysis: project_subject_categories

In [0]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [0]:

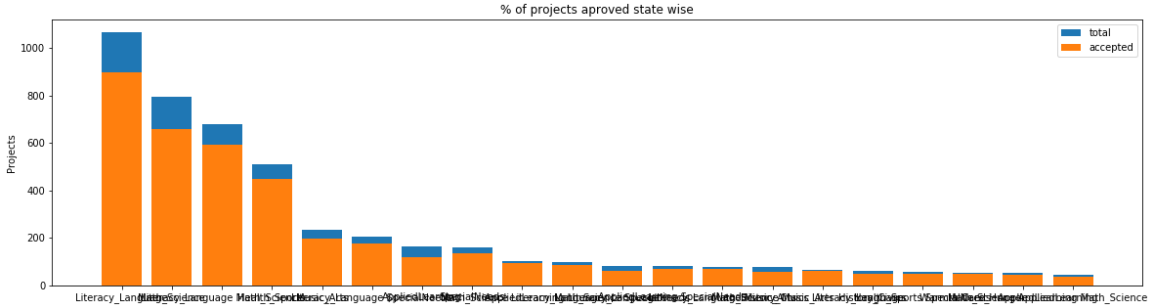
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proj
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

In [0]:

```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=20)
```



	clean_categories	project_is_approved	total	Avg
23	Literacy_Language	900	1067	0.843486
30	Math_Science	659	795	0.828931
26	Literacy_Language Math_Science	594	679	0.874816
8	Health_Sports	447	509	0.878193
37	Music_Arts	199	233	0.854077

==========

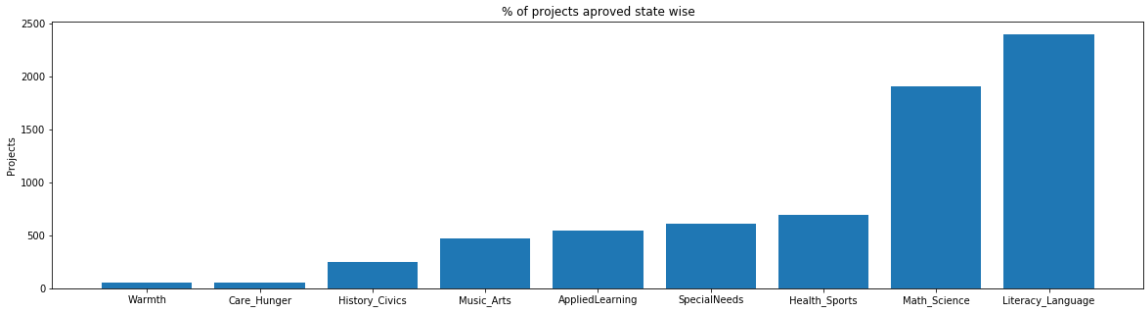
	clean_categories	project_is_approved	total	Avg
16	History_Civics	47	63	0.746032
14	Health_Sports SpecialNeeds	49	57	0.859649
46	Warmth Care_Hunger	47	53	0.886792
31	Math_Science AppliedLearning	44	52	0.846154
4	AppliedLearning Math_Science	35	44	0.795455

In [0]:

```
# 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())
```

In [0]:

```
# 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]))


ind = np.arange(len(sorted_cat_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(sorted_cat_dict.values()))

plt.ylabel('Projects')
plt.title('% of projects aproved state wise')
plt.xticks(ind, list(sorted_cat_dict.keys()))
plt.show()
```



In [0]:

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

```
Warmth               :        58
Care_Hunger          :        58
History_Civics       :       252
Music_Arts           :       476
AppliedLearning      :       547
SpecialNeeds         :       614
Health_Sports        :       697
Math_Science         :      1910
Literacy_Language    :      2400
```

Univariate Analysis: project_subject_subcategories

In [0]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [0]:

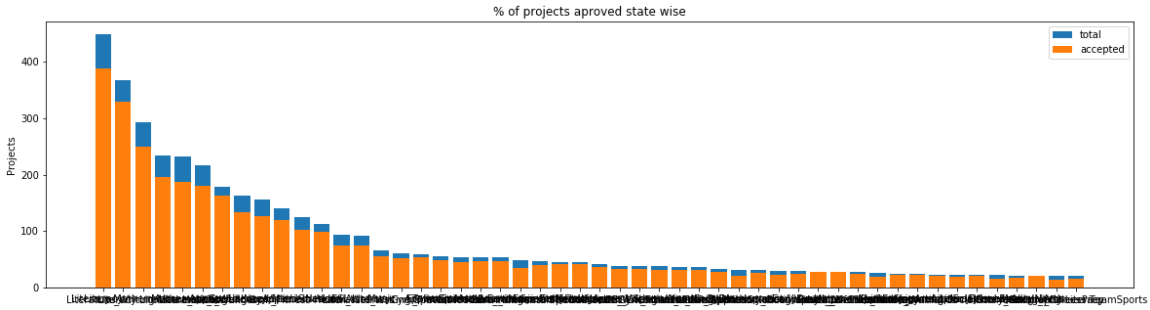
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proj
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

In [0]:

```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top=50)
```



	clean_subcategories	project_is_approved	total	Avg
189	Literacy	389	449	0.866370
191	Literacy Mathematics	329	368	0.894022
201	Literature_Writing Mathematics	250	293	0.853242
190	Literacy Literature_Writing	195	234	0.833333
209	Mathematics	188	232	0.810345

==========

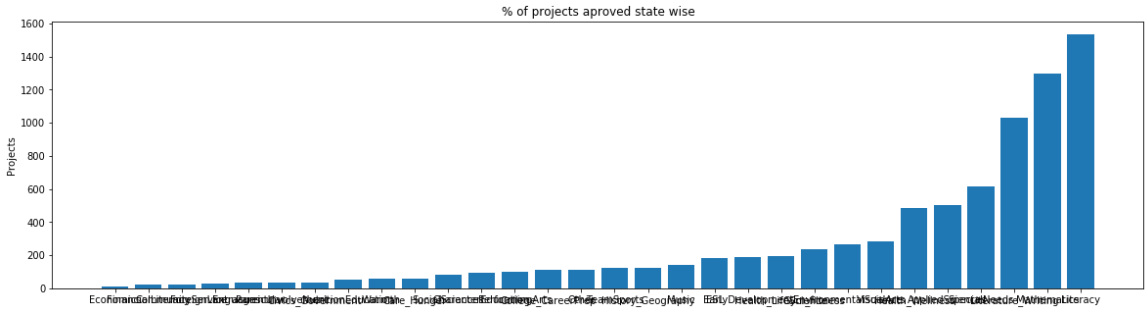
	clean_subcategories	project_is_approved	total	Avg
23	AppliedSciences VisualArts	15	22	0.681818
230	Other SpecialNeeds	17	21	0.809524
181	History_Geography Literacy	20	21	0.952381
56	College_CareerPrep	14	20	0.700000
177	Health_Wellness TeamSports	15	20	0.750000

In [0]:

```
# 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())
```

In [0]:

```
# 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]))


ind = np.arange(len(sorted_sub_cat_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(sorted_sub_cat_dict.values()))

plt.ylabel('Projects')
plt.title('% of projects aproved state wise')
plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
plt.show()
```



In [0]:

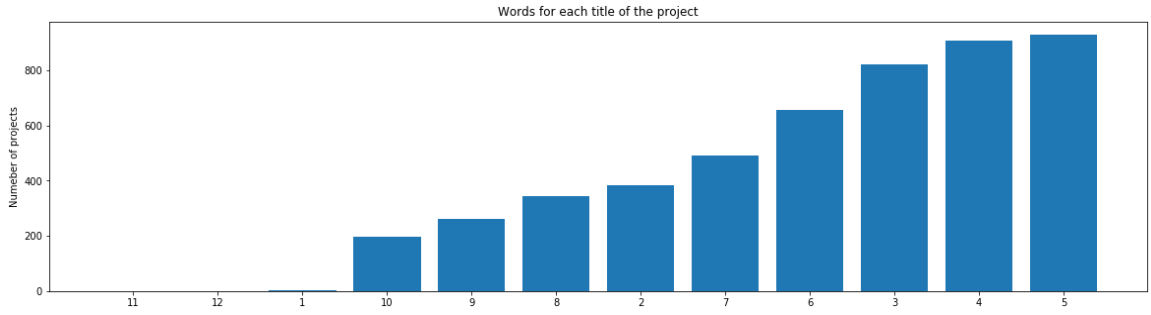
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :        14
FinancialLiteracy    :        23
CommunityService     :        26
ForeignLanguages     :        29
Extracurricular      :        33
ParentInvolvement    :        34
Civics_Government    :        36
NutritionEducation   :        54
Warmth               :        58
Care_Hunger          :        58
SocialSciences       :        82
CharacterEducation   :        95
PerformingArts       :       102
College_CareerPrep   :       113
Other                :       114
TeamSports           :       123
History_Geography    :       124
Music                :       142
ESL                  :       182
EarlyDevelopment     :       189
Health_LifeScience   :       196
Gym_Fitness          :       237
EnvironmentalScience :       265
VisualArts           :       282
Health_Wellness      :       486
AppliedSciences      :       504
SpecialNeeds         :       614
Literature_Writing   :      1032
Mathematics          :      1295
Literacy             :      1534
```

Univariate Analysis: Text features (Title)

In [0]:

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
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]))


ind = np.arange(len(word_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(word_dict.values()))

plt.ylabel('Numeber of projects')
plt.title('Words for each title of the project')
plt.xticks(ind, list(word_dict.keys()))
plt.show()
```

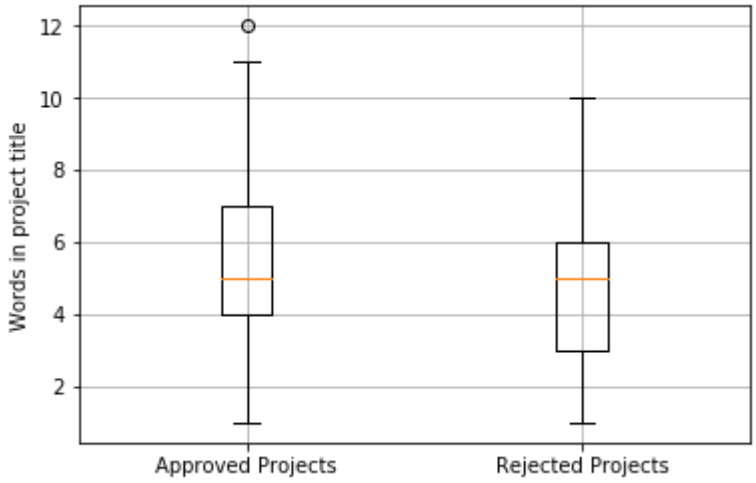


In [0]:

```
approved_word_count = project_data[project_data['project_is_approved']==1]['project_title'].str.split().apply(len)
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0]['project_title'].str.split().apply(len)
rejected_word_count = rejected_word_count.values
```

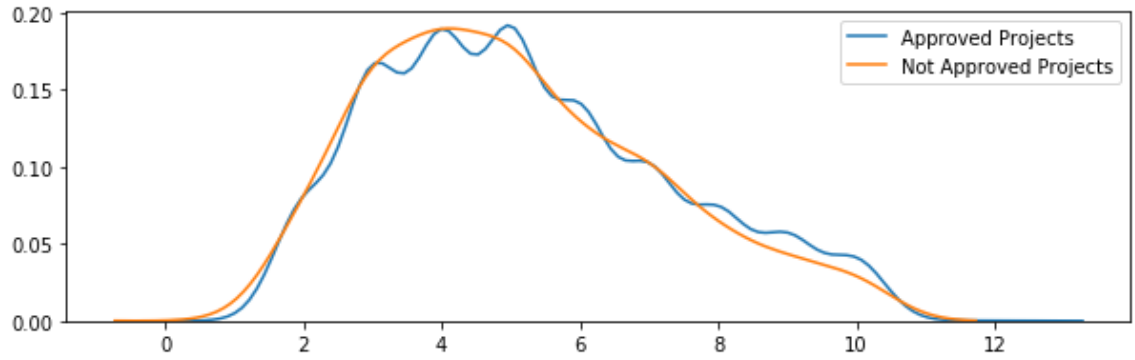
In [0]:

```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



In [0]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.legend()
plt.show()
```



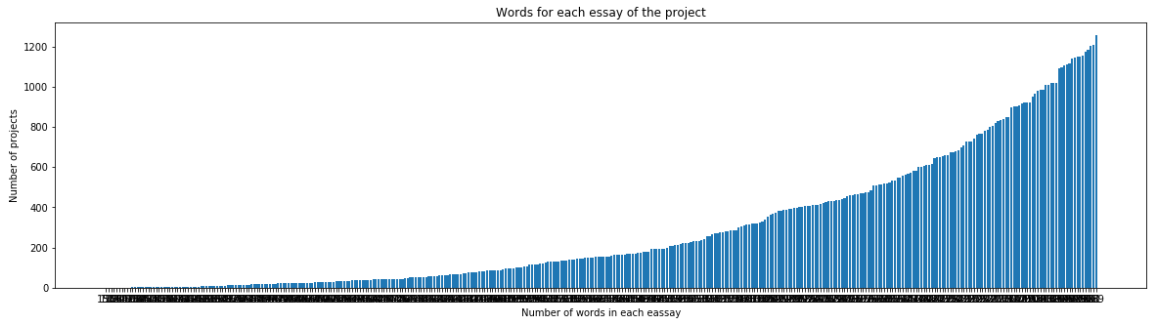
Univariate Analysis: Text features (Project Essay's)

In [0]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

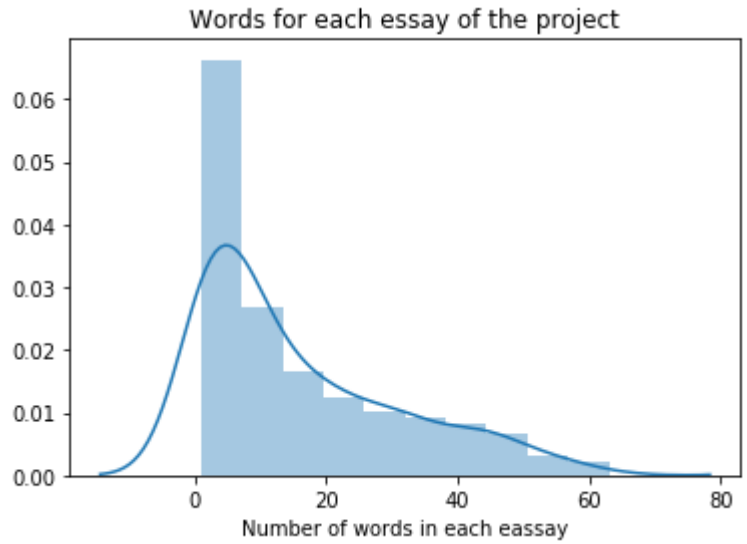
In [0]:

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
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]))


ind = np.arange(len(word_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(word_dict.values()))

plt.ylabel('Number of projects')
plt.xlabel('Number of words in each eassay')
plt.title('Words for each essay of the project')
plt.xticks(ind, list(word_dict.keys()))
plt.show()
```



In [0]:

```
sns.distplot(word_count.values)
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.show()
```

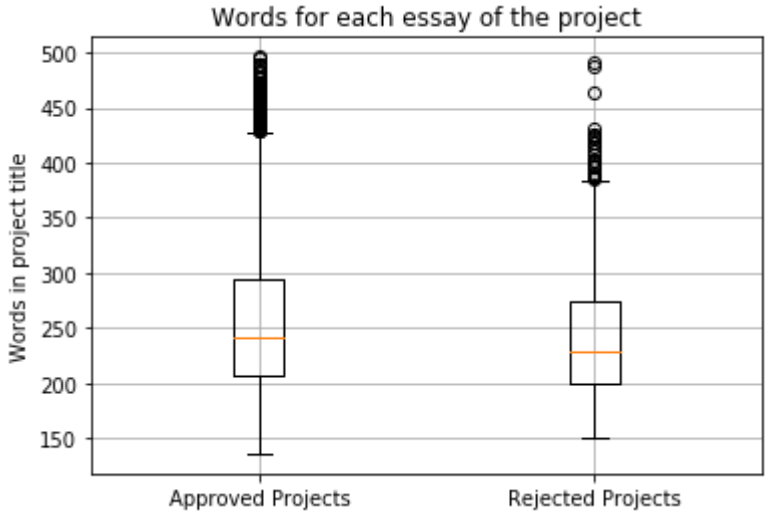


In [0]:

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

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

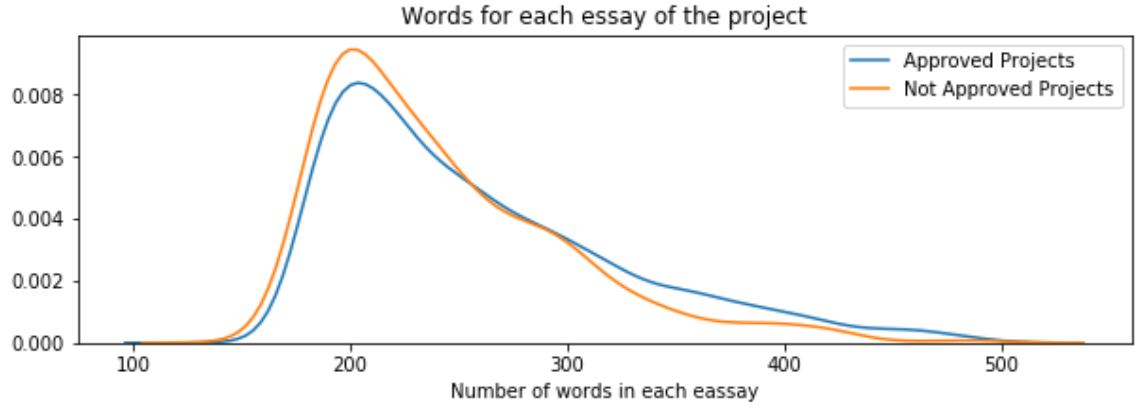
In [0]:

```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



In [0]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```



Univariate Analysis: Cost per project

In [0]:

```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[0]:

	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 [0]:

```
# 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[0]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [0]:

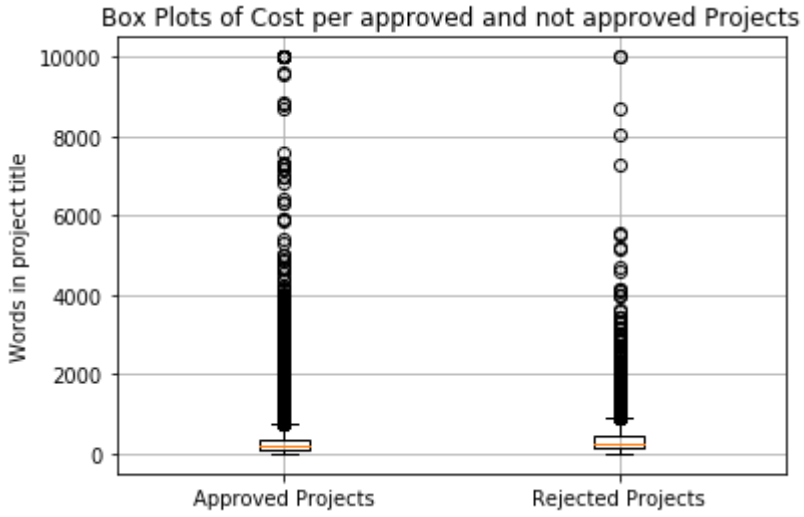
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [0]:

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

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

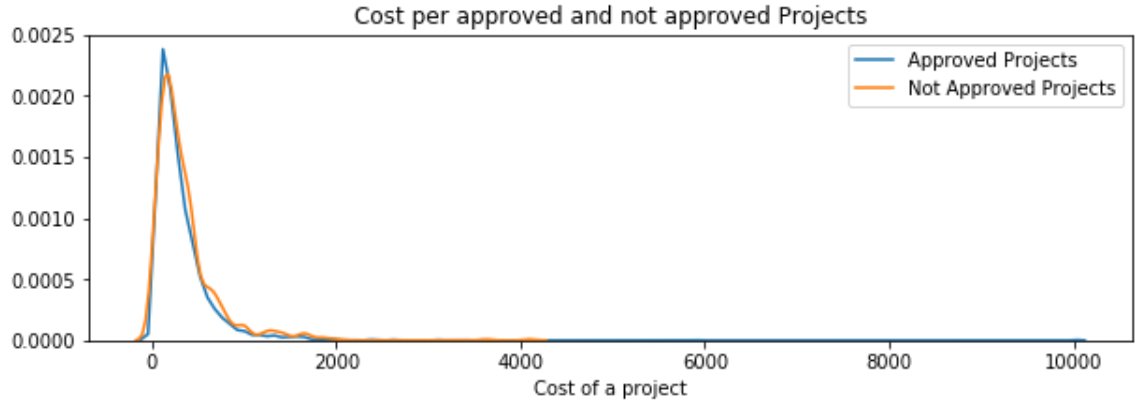
In [0]:

```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



In [0]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_price, hist=False, label="Approved Projects")
sns.distplot(rejected_price, hist=False, label="Not Approved Projects")
plt.title('Cost per approved and not approved Projects')
plt.xlabel('Cost of a project')
plt.legend()
plt.show()
```



In [0]:

```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Percentile", "Approved Projects", "Not Approved Projects"]

for i in range(0,101,5):
    x.add_row([i,np.round(np.percentile(approved_price,i), 3), np.round(np.percentile(r
ejected_price,i), 3)])
print(x)
```

```
+------------+------------------+----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+------------------+----------------------+
|     0      |        0.66       |          1.97        |
|     5      |       13.59       |          41.9        |
|     10     |       33.88       |         73.67        |
|     15     |        58.0       |         99.109       |
|     20     |       77.38       |         118.56       |
|     25     |       99.95       |        140.892       |
|     30     |       116.68      |         162.23       |
|     35     |      137.232      |        184.014       |
|     40     |       157.0       |        208.632       |
|     45     |      178.265      |        235.106       |
|     50     |       198.99      |        263.145       |
|     55     |       223.99      |         292.61       |
|     60     |       255.63      |        325.144       |
|     65     |      285.412      |         362.39       |
|     70     |      321.225      |         399.99       |
|     75     |      366.075      |        449.945       |
|     80     |       411.67      |        519.282       |
|     85     |       479.0       |        618.276       |
|     90     |       593.11      |        739.356       |
|     95     |      801.598      |        992.486       |
|    100     |       9999.0      |         9999.0       |
+------------+------------------+----------------------+
```

In [0]:

```
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_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_approve
d',
       'price', 'quantity'],
      dtype='object')
Head of project_data:
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proj
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

2. Preprocessing Categorical Features: project_grade_category

In [0]:

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

Out[0]:

```
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

In [0]:

```
# 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[0]:

```
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 [0]:

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

Out[0]:

```
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
Health & Sports, Warmth, Care & Hunger           23
Special Needs, 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
```

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

In [0]:

```
project_data['project_subject_categories'] = project_data['project_subject_categories']
.str.replace(' The ','')
project_data['project_subject_categories'] = project_data['project_subject_categories']
.str.replace(' ','')
project_data['project_subject_categories'] = project_data['project_subject_categories']
.str.replace('&','_')
project_data['project_subject_categories'] = project_data['project_subject_categories']
.str.replace(',','_')
project_data['project_subject_categories'] = project_data['project_subject_categories']
.str.lower()
project_data['project_subject_categories'].value_counts()
```

Out[0]:

```
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
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
appliedlearning_warmth_care_hunger          10
music_arts_appliedlearning                  10
literacy_language_warmth_care_hunger         9
music_arts_warmth_care_hunger                2
history_civics_warmth_care_hunger            1
Name: project_subject_categories, dtype: int64
```

4. Preprocessing Categorical Features: teacher_prefix

In [0]:

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

Out[0]:

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

In [0]:

```
# 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
```

> 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 [0]:

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

In [0]:

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

Out[0]:

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

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

In [0]:

```
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()
```

Out[0]:

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

5. Preprocessing Categorical Features: project_subject_subcategories

In [0]:

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

Out[0]:

```
Literacy                                        9486
Literacy, Mathematics                           8325
Literature & Writing, Mathematics               5923
Literacy, Literature & Writing                  5571
Mathematics                                     5379
Literature & Writing                            4501
Special Needs                                   4226
Health & Wellness                               3583
Applied Sciences, Mathematics                   3399
Applied Sciences                                2492
Literacy, Special Needs                         2440
Gym & Fitness, Health & Wellness                2264
ESL, Literacy                                   2234
Visual Arts                                     2217
Music                                           1472
Warmth, Care & Hunger                           1309
Literature & Writing, Special Needs             1306
Gym & Fitness                                   1195
Health & Wellness, Special Needs                1189
Mathematics, Special Needs                      1187
Environmental Science                           1079
Team Sports                                     1061
Applied Sciences, Environmental Science          984
Environmental Science, Health & Life Science     964
Music, Performing Arts                           948
Early Development                                905
Environmental Science, Mathematics               838
Other                                            831
Health & Life Science                            827
Health & Wellness, Nutrition Education           797
                                                ...
Environmental Science, Team Sports                 2
Foreign Languages, Gym & Fitness                   2
Character Education, Economics                     2
Economics, Literature & Writing                    2
Social Sciences, Team Sports                       2
Applied Sciences, Warmth, Care & Hunger            2
History & Geography, Warmth, Care & Hunger         1
Literature & Writing, Nutrition Education          1
Civics & Government, Nutrition Education           1
Other, Warmth, Care & Hunger                       1
Financial Literacy, Performing Arts                1
Community Service, Gym & Fitness                   1
College & Career Prep, Warmth, Care & Hunger       1
Gym & Fitness, Warmth, Care & Hunger               1
Economics, Foreign Languages                       1
Extracurricular, Financial Literacy                1
Community Service, Music                           1
Economics, Music                                   1
ESL, Economics                                     1
Parent Involvement, Team Sports                    1
Community Service, Financial Literacy              1
ESL, Team Sports                                   1
Economics, Nutrition Education                     1
Economics, Other                                   1
Gym & Fitness, Social Sciences                     1
Financial Literacy, Foreign Languages              1
Civics & Government, Foreign Languages             1
Gym & Fitness, Parent Involvement                  1
```

```
Civics & Government, Parent Involvement            1
Parent Involvement, Warmth, Care & Hunger          1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

same process we did in project_subject_categories

In [0]:

```
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(' The ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(' ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace('&','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(',','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.lower()
project_data['project_subject_subcategories'].value_counts()
```

Out[0]:

```
literacy                                        9486
literacy_mathematics                            8325
literature_writing_mathematics                  5923
literacy_literature_writing                     5571
mathematics                                     5379
literature_writing                              4501
specialneeds                                    4226
health_wellness                                 3583
appliedsciences_mathematics                     3399
appliedsciences                                 2492
literacy_specialneeds                           2440
gym_fitness_health_wellness                     2264
esl_literacy                                    2234
visualarts                                      2217
music                                           1472
warmth_care_hunger                              1309
literature_writing_specialneeds                 1306
gym_fitness                                     1195
health_wellness_specialneeds                    1189
mathematics_specialneeds                        1187
environmentalscience                            1079
teamsports                                      1061
appliedsciences_environmentalscience             984
environmentalscience_health_lifescience          964
music_performingarts                             948
earlydevelopment                                 905
environmentalscience_mathematics                 838
other                                            831
health_lifescience                               827
health_wellness_nutritioneducation               797
                                                ...
charactereducation_economics                       2
college_careerprep_teamsports                      2
foreignlanguages_gym_fitness                       2
extracurricular_foreignlanguages                   2
socialsciences_teamsports                          2
economics_literature_writing                       2
parentinvolvement_teamsports                       1
esl_teamsports                                     1
civics_government_parentinvolvement                1
civics_government_nutritioneducation               1
financialliteracy_performingarts                   1
parentinvolvement_warmth_care_hunger               1
gym_fitness_parentinvolvement                      1
history_geography_warmth_care_hunger               1
communityservice_financialliteracy                 1
esl_economics                                      1
literature_writing_nutritioneducation              1
gym_fitness_socialsciences                         1
other_warmth_care_hunger                           1
civics_government_foreignlanguages                 1
economics_music                                    1
financialliteracy_foreignlanguages                 1
economics_nutritioneducation                       1
communityservice_gym_fitness                       1
college_careerprep_warmth_care_hunger              1
extracurricular_financialliteracy                  1
economics_other                                    1
communityservice_music                             1
```

```
gym_fitness_warmth_care_hunger                1
economics_foreignlanguages                    1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

6. Preprocessing Categorical Features: school_state

In [0]:

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

Out[0]:

```
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
```

convert all of them into small letters

In [0]:

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

Out[0]:

```
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 [0]:

```
# 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 [0]:

```
# 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 [0]:

```
project_data['project_title'].head(5)
```

Out[0]:

```
0     Educational Support for English Learners at Home
1                Wanted: Projector for Hungry Learners
2    Soccer Equipment for AWESOME Middle School Stu...
3                               Techie Kindergarteners
4                               Interactive Math Tools
Name: project_title, dtype: object
```

In [0]:

```
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 [0]:

```
# 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 [0]:

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

```
100%|██████████| 109248/109248 [00:03<00:00, 34988.80it/s]
```

In [0]:

```
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 love reading pure pleasure
34 ball
147 needs chromebook
```

8. Preprocessing Categorical Features: essay

In [0]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [0]:

```
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])
```

printing some random essay
9 Over 95% of my students are on free or reduced lunch. I have a few who are homeless, but despite that, they come to school with an eagerness to l earn. My students are inquisitive eager learners who embrace the challen ge of not having great books and other resources every day. Many of them are not afforded the opportunity to engage with these big colorful pages o f a book on a regular basis at home and they don't travel to the public li brary. \r\nIt is my duty as a teacher to do all I can to provide each stu dent an opportunity to succeed in every aspect of life. \r\nReading is Fun damental! My students will read these books over and over again while boos ting their comprehension skills. These books will be used for read alouds, partner reading and for Independent reading. \r\nThey will engage in readi ng to build their \"Love for Reading\" by reading for pure enjoyment. They will be introduced to some new authors as well as some old favorites. I wa nt my students to be ready for the 21st Century and know the pleasure of h olding a good hard back book in hand. There's nothing like a good book to read! \r\nMy students will soar in Reading, and more because of your cons ideration and generous funding contribution. This will help build stamina and prepare for 3rd grade. Thank you so much for reading our proposal!nann an
---
34 My students mainly come from extremely low-income families, and the maj ority of them come from homes where both parents work full time. Most of m y students are at school from 7:30 am to 6:00 pm (2:30 to 6:00 pm in the a fter-school program), and they all receive free and reduced meals for brea kfast and lunch. \r\n\r\n\r\nI want my students to feel as comfortable in my classroom as they do at home. Many of my students take on multiple role s both at home as well as in school. They are sometimes the caretakers of younger siblings, cooks, babysitters, academics, friends, and most of all, they are developing who they are going to become as adults. I consider it an essential part of my job to model helping others gain knowledge in a po sitive manner. As a result, I have a community of students who love helpin g each other in and outside of the classroom. They consistently look for o pportunities to support each other's learning in a kind and helpful way.I am excited to be experimenting with alternative seating in my classroom th is school year. Studies have shown that giving students the option of wher e they sit in a classroom increases focus as well as motivation. \r\n\r\n By allowing students choice in the classroom, they are able to explore and create in a welcoming environment. Alternative classroom seating has been experimented with more frequently in recent years. I believe (along with m any others), that every child learns differently. This does not only apply to how multiplication is memorized, or a paper is written, but applies to the space in which they are asked to work. I have had students in the past ask \"Can I work in the library? Can I work on the carpet?\" My answer was always, \"As long as you're learning, you can work wherever you want!\" \r \n\r\nWith the yoga balls and the lap-desks, I will be able to increase th e options for seating in my classroom and expand its imaginable space.nann an
---
147 My students are eager to learn and make their mark on the world.\r\n\r \nThey come from a Title 1 school and need extra love.\r\n\r\nMy fourth gr ade students are in a high poverty area and still come to school every day to get their education. I am trying to make it fun and educational for the m so they can get the most out of their schooling. I created a caring envi ronment for the students to bloom! They deserve the best.\r\nThank you!\r \nI am requesting 1 Chromebook to access online interventions, differentia te instruction, and get extra practice. The Chromebook will be used to sup plement ELA and math instruction. Students will play ELA and math games th at are engaging and fun, as well as participate in assignments online. Thi s in turn will help my students improve their skills. Having a Chromebook in the classroom would not only allow students to use the programs at thei

```
r own pace, but would ensure more students are getting adequate time to us
e the programs. The online programs have been especially beneficial to my
students with special needs. They are able to work at their level as well
as be challenged with some different materials. This is making these stude
nts more confident in their abilities.\r\n\r\nThe Chromebook would allow m
y students to have daily access to computers and increase their computing
skills.\r\nThis will change their lives for the better as they become more
successful in school. Having access to technology in the classroom would h
elp bridge the achievement gap.nannan
```

In [0]:

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

```
100%|██████████| 109248/109248 [01:07<00:00, 1623.95it/s]
```

In [0]:

```
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 95 students free reduced lunch homeless despite come school eagerness le
arn students inquisitive eager learners embrace challenge not great books
resources every day many not afforded opportunity engage big colorful page
s book regular basis home not travel public library duty teacher provide s
tudent opportunity succeed every aspect life reading fundamental students
read books boosting comprehension skills books used read alouds partner re
ading independent reading engage reading build love reading reading pure e
njoyment introduced new authors well old favorites want students ready 21s
t century know pleasure holding good hard back book hand nothing like good
book read students soar reading consideration generous funding contributio
n help build stamina prepare 3rd grade thank much reading proposal nannan
--------------------------------------------------
34 students mainly come extremely low income families majority come homes
parents work full time students school 7 30 6 00 pm 2 30 6 00 pm school pr
ogram receive free reduced meals breakfast lunch want students feel comfor
table classroom home many students take multiple roles home well school so
metimes caretakers younger siblings cooks babysitters academics friends de
veloping going become adults consider essential part job model helping oth
ers gain knowledge positive manner result community students love helping
outside classroom consistently look opportunities support learning kind he
lpful way excited experimenting alternative seating classroom school year
studies shown giving students option sit classroom increases focus well mo
tivation allowing students choice classroom able explore create welcoming
environment alternative classroom seating experimented frequently recent y
ears believe along many others every child learns differently not apply mu
ltiplication memorized paper written applies space asked work students pas
t ask work library work carpet answer always long learning work wherever w
ant yoga balls lap desks able increase options seating classroom expand im
aginable space nannan
--------------------------------------------------
147 students eager learn make mark world come title 1 school need extra lo
ve fourth grade students high poverty area still come school every day get
education trying make fun educational get schooling created caring environ
ment students bloom deserve best thank requesting 1 chromebook access onli
ne interventions differentiate instruction get extra practice chromebook u
sed supplement ela math instruction students play ela math games engaging
fun well participate assignments online turn help students improve skills
chromebook classroom would not allow students use programs pace would ensu
re students getting adequate time use programs online programs especially
beneficial students special needs able work level well challenged differen
t materials making students confident abilities chromebook would allow stu
dents daily access computers increase computing skills change lives better
become successful school access technology classroom would help bridge ach
ievement gap nannan
```

In [0]:

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

8. Preprocessing Numerical Values: price

8.1 applying StandardScaler

In [0]:

```
"""
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(project_data['price'].values.reshape(-1, 1))
project_data['std_price']=scaler.transform(project_data['price'].values.reshape(-1, 1)
 )
"""
```

Out[0]:

```
"\nfrom sklearn.preprocessing import StandardScaler\nscaler = StandardScal
er()\nscaler.fit(project_data['price'].values.reshape(-1, 1))\nproject_dat
a['std_price']=scaler.transform(project_data['price'].values.reshape(-1,
1) )\n"
```

8.2 applying MinMaxScaler

In [0]:

```
"""
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(project_data['price'].values.reshape(-1, 1))
project_data['nrm_price']=scaler.transform(project_data['price'].values.reshape(-1, 1))
"""
```

Out[0]:

```
"\nfrom sklearn.preprocessing import MinMaxScaler\n\nscaler = MinMaxScaler
()\nscaler.fit(project_data['price'].values.reshape(-1, 1))\nproject_data
['nrm_price']=scaler.transform(project_data['price'].values.reshape(-1,
1))\n"
```

In [0]:

```
project_data.to_csv('drive/My Drive/LSTM on Donors/preprocess.csv', index = False)
```

---------- Assignment: LSTM on Donors ----------

In [0]:

```
project_data = pd.read_csv('drive/My Drive/LSTM on Donors/preprocess.csv')
```

In [0]:

```
print("Shape of dataframe:", project_data.shape)
print("Number of rows:", project_data.shape[0])
print("Number of columns:", project_data.shape[1], '\n')
print("Column names:")
print(project_data.columns)

print("\nHead 3 of dataframe:\n")
project_data.head(3)
```

```
Shape of dataframe: (109248, 20)
Number of rows: 109248
Number of columns: 20

Column names:
Index(['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_approve
d',
       'price', 'quantity', 'essay'],
      dtype='object')

Head 3 of dataframe:
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proj
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs	in	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr	fl	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms	az	

In [0]:

```
print("\nChecking for null values\n")
project_data.info()
```

```
Checking for null values

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109248 entries, 0 to 109247
Data columns (total 20 columns):
Unnamed: 0                                      109248 non-null int64
id                                              109248 non-null object
teacher_id                                      109248 non-null object
teacher_prefix                                  109248 non-null object
school_state                                    109248 non-null object
project_submitted_datetime                      109248 non-null object
project_grade_category                          109248 non-null object
project_subject_categories                      109248 non-null object
project_subject_subcategories                   109248 non-null object
project_title                                   109248 non-null object
project_essay_1                                 109248 non-null object
project_essay_2                                 109248 non-null object
project_essay_3                                 3758 non-null object
project_essay_4                                 3758 non-null object
project_resource_summary                        109248 non-null object
teacher_number_of_previously_posted_projects    109248 non-null int64
project_is_approved                             109248 non-null int64
price                                           109248 non-null float64
quantity                                        109248 non-null int64
essay                                           109248 non-null object
dtypes: float64(1), int64(4), object(15)
memory usage: 16.7+ MB
```

Observations:

We can see null values in project_essay_3 and project_essay_4. Anyway, we are going to drop those columns along with other columns in the next cell. We can ignore them for now.

In [0]:

```
project_data.columns
```

Out[0]:

```
Index(['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_approve
d',
       'price', 'quantity', 'essay'],
      dtype='object')
```

In [0]:

```
# 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, 61291.22it/s]
```

In [0]:

```
# Dropping unnecessary columns from dataframe
```

In [0]:

```
print("Columns in project_data before removing unnecessary columns\n")
project_data.columns
```

```
Columns in project_data before removing unnecessary columns
```

Out[0]:

```
Index(['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_approve
d',
       'price', 'quantity', 'essay', 'project_summary_numerical'],
      dtype='object')
```

In [0]:

```
project_data_1 = project_data.drop(['Unnamed: 0', 'id', 'teacher_id', 'project_submitte
d_datetime', 'project_title',
                                    'project_essay_1', 'project_essay_2', 'project_essa
y_3', 'project_essay_4',
                                    'project_resource_summary'], axis = 1)
```

In [0]:

```
print("Columns in project_data_1 after removing unnecessary columns\n")
project_data_1.columns
```

```
Columns in project_data_1 after removing unnecessary columns
```

Out[0]:

```
Index(['teacher_prefix', 'school_state', 'project_grade_category',
       'project_subject_categories', 'project_subject_subcategories',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'price', 'quantity', 'essay', 'project_summary_numerical'],
      dtype='object')
```

In [0]:

```
project_data_1 = pd.read_csv('drive/My Drive/LSTM on Donors/preprocess_1.csv')
```

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

In [0]:

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

Splitting into train, cv and test set

In [0]:

```
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, stratify = y)

# 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_st
ate = 42, stratify = y_train)
```

In [8]:

```
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, 10)
Shape of x_cv: (21850, 10)
Shape of x_test: (21850, 10)
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://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/)

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

https://github.com/stanfordnlp/GloVe (https://github.com/stanfordnlp/GloVe)

In [1]:

```
### 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/
```

In [12]:

```
"""
from zipfile import ZipFile

file_name = "glove.42B.300d.zip"

with ZipFile(file_name, 'r') as zip:

    zip.printdir()

    # Extracting all the files
    print('Extracting all the files from zip file')
    zip.extractall()
    print('Done!')
"""
```

Out[12]:

```
'\nfrom zipfile import ZipFile\n\nfile_name = "glove.42B.300d.zip"\n\nwith
ZipFile(file_name, \'r\') as zip: \n     \n    zip.printdir() \n  \n    #
Extracting all the files \n    print(\'Extracting all the files from zip f
ile\')\n    zip.extractall() \n    print(\'Done!\') \n'
```

In [13]:

```
"""
glove_words = {}

with open("glove.42B.300d.txt") as glove:

  for data in glove:
    words = data.split()
    word = words[0]
    vec = np.asarray(words[1:], dtype='float32')
    glove_words[word] = vec
print("Number of words in glove vector:", len(glove_words))

"""
```

Out[13]:

```
'\nglove_words = {}\n\nwith open("glove.42B.300d.txt") as glove:\n  \n  fo
r data in glove:\n    words = data.split()\n    word = words[0]\n    vec =
np.asarray(words[1:], dtype=\'float32\')\n    glove_words[word] = vec\npri
nt("Number of words in glove vector:", len(glove_words))\n\n'
```

Import glove_vectors file

In [0]:

```
with open('drive/My Drive/LSTM on Donors/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  model
```

Defining sequence length, vocabulary size and embedding size.

In [0]:

```
# 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.

In [18]:

```
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!')
```

```
Using TensorFlow backend.

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

In [20]:

```
# 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 = glove_words.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: 45966
Shape of train weight matrix: (45966, 300)
```

Padding document

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

In [22]:

```
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 = 'po
st')

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

In [0]:

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

In [25]:

```
from keras.layers import Embedding, Dense, Flatten, Input, LSTM, Dropout, BatchNormaliz
ation, concatenate


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)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Pleas
e use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. P
lease use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:190: The name tf.get_default_session is deprecate
d. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Pleas
e use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:203: The name tf.Session is deprecated. Please us
e tf.compat.v1.Session instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:216: The name tf.is_variable_initialized is depre
cated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:223: The name tf.variables_initializer is depreca
ted. Please use tf.compat.v1.variables_initializer instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:148: The name tf.placeholder_with_default is depr
ecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:3733: calling dropout (from tensorflow.python.op
s.nn_ops) with keep_prob is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

Embedding layer for Categorical Features

Categorical Feature: teacher_prefix

Embedding layer for teacher_prefix

In [0]:

```
# 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

In [0]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

tr_tea_pre_encode = le.fit_transform(x_tr['teacher_prefix'])
cv_tea_pre_encode = le.transform(x_cv['teacher_prefix'])
test_tea_pre_encode = le.transform(x_test['teacher_prefix'])
```

Categorical feature: school_state

Embedding layer for school_state

In [0]:

```
# 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

In [0]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

tr_sch_encode = le.fit_transform(x_tr['school_state'])
cv_sch_encode = le.transform(x_cv['school_state'])
test_sch_encode = le.transform(x_test['school_state'])
```

Categorical feature: project_grade_category

Creating embedding layer for project_grade_category

In [0]:

```
# 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

In [0]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

tr_pro_gra_encode = le.fit_transform(x_tr['project_grade_category'])
cv_pro_gra_encode = le.transform(x_cv['project_grade_category'])
test_pro_gra_encode = le.transform(x_test['project_grade_category'])
```

Categorical feature: project_subject_categories

Embedding layer for project_subject_categories

In [0]:

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

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

# Creating an embedding layer
emb_pro_sub = Embedding(input_dim = pro_sub_uni, output_dim = emb_pro_sub_size,
                        trainable = True, name = "project_subject_categories_emb")(inp
_pro_sub)

flatten_pro_sub = Flatten()(emb_pro_sub)
```

Label encoding for project_subject_categories

In [0]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

le.fit(x_tr['project_subject_categories'])

x_test["project_subject_categories"] = x_test["project_subject_categories"].map(lambda
a: '<unknown>' if a not in le.classes_ else a)
x_cv["project_subject_categories"] = x_cv["project_subject_categories"].map(lambda a:
'<unknown>' if a not in le.classes_ else a)

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

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

Categorical feature: project_subject_subcategories

Embedding layer for project_subject_subcategories

In [0]:

```
# Unique values
pro_sub_1_uni = x_tr['project_subject_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 = "project_subject_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 = "project_subject_subcategories_emb")(
inp_pro_sub)

flatten_pro_sub_1 = Flatten()(emb_pro_sub_1)
```

Label encoding for project_subject_subcategories

In [0]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

le.fit(x_tr["project_subject_subcategories"])

x_test["project_subject_subcategories"] = x_test["project_subject_subcategories"].map(l
ambda a: '<unknown>' if a not in le.classes_ else a)
x_cv["project_subject_subcategories"] = x_cv["project_subject_subcategories"].map(lambd
a a: '<unknown>' if a not in le.classes_ else a)

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

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

Numerical Features

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

In [0]:

```
# 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_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

In [0]:

```
# 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

In [0]:

```
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

In [0]:

```
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 (-1,1)
emb_num = Dense(100, activation = "relu")(inp_num)
```

Concatenating all the flattened layers

In [0]:

```
 from keras.layers import concatenate

con_lay = concatenate([flatten_1, flatten_tea_pre, flatten_sch, flatten_pro_gra, flatte
n_pro_sub, flatten_pro_sub_1, emb_num])
```

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

Keras model:

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

In [0]:

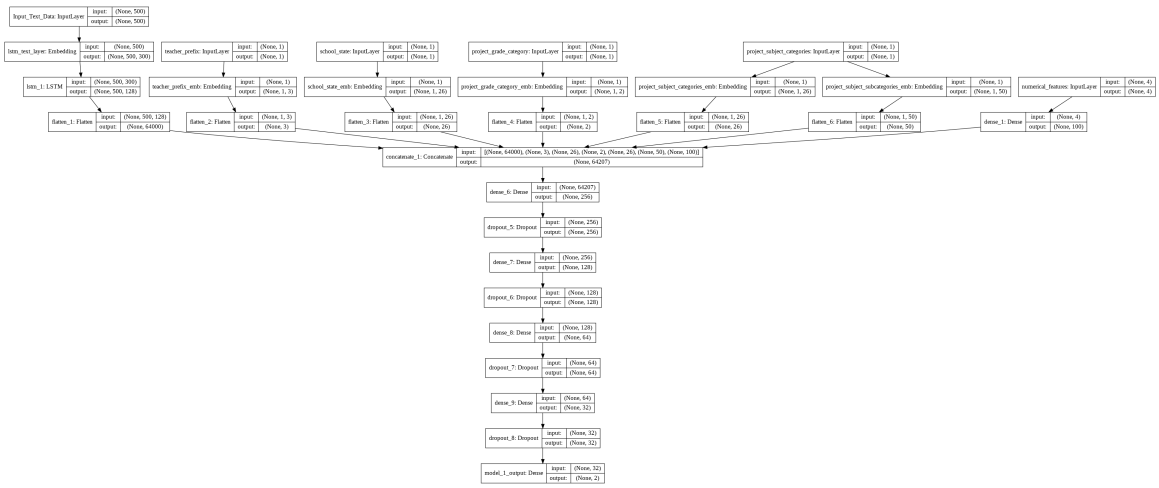
```
from keras.models import Model
from keras import regularizers, initializers

# 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

In [61]:

```
# 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')
```

Out[61]:



Getting all data into list.

In [0]:

```
# Train data
tr_data_1 = [pad_tr, tr_tea_pre_encode, tr_sch_encode, tr_pro_sub_encode, tr_sub_1_encoder, tr_pro_gra_encode, tr_ss]

# CV data
cv_data_1 = [pad_cv, cv_tea_pre_encode, cv_sch_encode, cv_pro_sub_encode, cv_sub_1_encoder, cv_pro_gra_encode, cv_ss]

# Test data
test_data_1 = [pad_test, test_tea_pre_encode, test_sch_encode, test_pro_sub_encode, test_sub_1_encoder, test_pro_gra_encode, test_ss]
```

Chaning type of dependent variable (y) to categorical type

In [0]:

```
from keras.utils import np_utils

y_tr_data_1 = np_utils.to_categorical(y_tr, 2)
y_cv_data_1 = np_utils.to_categorical(y_cv, 2)
y_test_data_1 = np_utils.to_categorical(y_test, 2)
```

AUC-ROC custom function

Source link: https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-roc-and-auc-in-keras (https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-roc-and-auc-in-keras)

In [0]:

```
from sklearn.metrics import roc_auc_score

import tensorflow as tf

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

Creating Callback with Checkpoint, EarlyStopping and Tensorboard

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

In [0]:

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

# Saves the model after every epoch
checkpoint_1 = ModelCheckpoint("model_1.h5", monitor = "val_loss", 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)

# TensorBoard is a visualization tool provided with TensorFlow.
tensorboard_1 = TensorBoard(log_dir = "drive/My Drive/LSTM on Donors/graph_1",
                         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_1 = [checkpoint_1, earlystop_1, tensorboard_1]
```

Compile the data

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

In [0]:

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

In [0]:

```
from keras.optimizers import Adam, RMSprop

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

Fitting model and callback to visualize model

In [0]:

```
history_1 = model_1.fit(tr_data_1, y_tr_data_1, batch_size = 512,
                        epochs = 30, validation_data = (cv_data_1, y_cv_data_1), verbos
e = 1,
                        callbacks = callback_1)
```

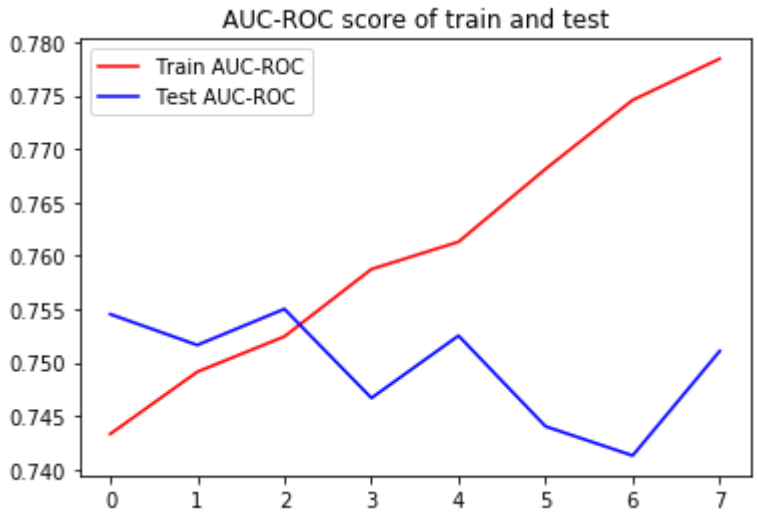
```
Train on 65548 samples, validate on 21850 samples
Epoch 1/30
65548/65548 [==============================] - 164s 3ms/step - loss: 0.461
1 - auroc: 0.7433 - val_loss: 0.5152 - val_auroc: 0.7545

Epoch 00001: val_loss improved from 0.69996 to 0.51524, saving model to mo
del_1.h5
Epoch 2/30
65548/65548 [==============================] - 164s 3ms/step - loss: 0.449
7 - auroc: 0.7491 - val_loss: 0.4804 - val_auroc: 0.7516

Epoch 00002: val_loss improved from 0.51524 to 0.48043, saving model to mo
del_1.h5
Epoch 3/30
65548/65548 [==============================] - 164s 3ms/step - loss: 0.444
8 - auroc: 0.7524 - val_loss: 0.4742 - val_auroc: 0.7550

Epoch 00003: val_loss improved from 0.48043 to 0.47416, saving model to mo
del_1.h5
Epoch 4/30
65548/65548 [==============================] - 165s 3ms/step - loss: 0.441
7 - auroc: 0.7587 - val_loss: 0.4758 - val_auroc: 0.7467

Epoch 00004: val_loss did not improve from 0.47416
Epoch 5/30
65548/65548 [==============================] - 164s 2ms/step - loss: 0.438
1 - auroc: 0.7613 - val_loss: 0.5059 - val_auroc: 0.7525

Epoch 00005: val_loss did not improve from 0.47416
Epoch 6/30
65548/65548 [==============================] - 163s 2ms/step - loss: 0.435
0 - auroc: 0.7681 - val_loss: 0.5583 - val_auroc: 0.7440

Epoch 00006: val_loss did not improve from 0.47416
Epoch 7/30
65548/65548 [==============================] - 163s 2ms/step - loss: 0.432
7 - auroc: 0.7746 - val_loss: 0.5075 - val_auroc: 0.7413

Epoch 00007: val_loss did not improve from 0.47416
Epoch 8/30
65548/65548 [==============================] - 164s 3ms/step - loss: 0.431
5 - auroc: 0.7785 - val_loss: 0.5173 - val_auroc: 0.7511

Epoch 00008: val_loss did not improve from 0.47416
Restoring model weights from the end of the best epoch
Epoch 00008: early stopping
```

Evaluating test data

In [0]:

```
# 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 [==============================] - 20s 925us/step
Test Loss: 0.4756210258596276
Test ROC-AUC score: 0.7528369569461162
```



Observation:

- Test Loss - 0.47
- Test AUC-ROC - 0.752

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

In [0]:

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

In [0]:

```
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_stat
e = 42, stratify = y)

# 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_s
tate = 42, stratify = y_train)
```

In [115]:

```
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, 10)
Shape of x_cv: (21850, 10)
Shape of x_test: (21850, 10)
Shape of y_tr: (65548,)
Shape of y_cv: (21850,)
Shape of y_test: (21850,)
```

Applying TF-IDF vectorizer

In [0]:

```
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

In [118]:

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

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

```
First 10 idf values

[ 7.21535591  5.90846833 11.39740605 11.39740605 11.39740605 10.14464308
 11.39740605  9.60564658 10.70425887 11.39740605]
```

In [0]:

```
# Zipping feature names corresponding to idf_ values

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

In [120]:

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

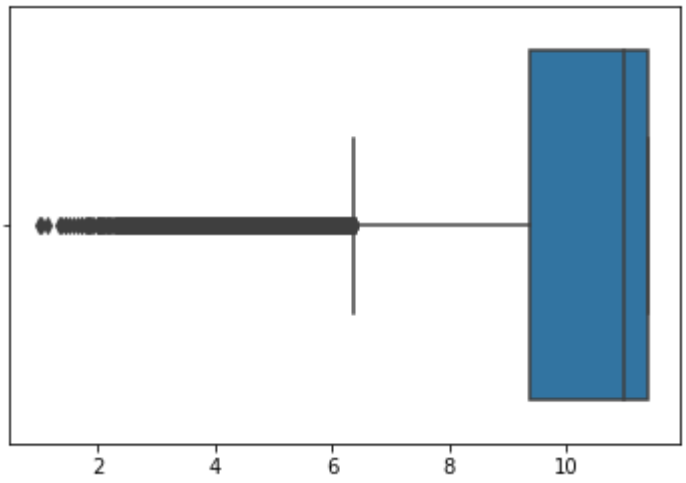
```
First 5 feature names along with idf values:

[(1.0075034040634312, 'students'), (1.0449310470519895, 'nannan'), (1.1630
512280481382, 'school'), (1.3624517377069705, 'learning'), (1.394231549062
3014, 'classroom')]

Last 5 feature names along with idf values:

[(11.397406052985405, 'zundel'), (11.397406052985405, 'zwink'), (11.397406
052985405, 'zx110'), (11.397406052985405, 'zydeco'), (11.397406052985405,
'zynergy')]
```

Box plot

In [122]:

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

```
Box plot for idf values
```



Observation:

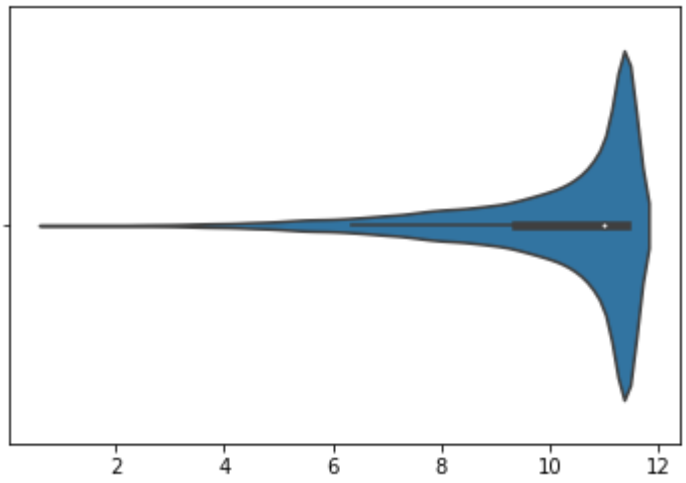
- Quartile 1: IDF values ranges from 0 to 9.3.
- Quartile 2: IDF values ranges from 9.4 to 10.99.
- Quartile 3: IDF values ranges from 11 to 11.39.
- Quartile 4" IDF values ranges from 11.39 to 11.399

Violin plot

In [124]:

```
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

In [125]:

```
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: 10.06835202916157
Median of idf values: 10.99194094487724
Maximum of idf values: 11.397406052985405
Minimum of idf values: 1.0075034040634312
```

In [126]:

```
# 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):
```

Out[126]:

```
2.014903020542265
```

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

In [0]:

```
list_words = []

for i in range(len(feat_idf)):

    if feat_idf[i][0] > 2 and feat_idf[i][0] < 11:
        words = feat_idf[i][1]
        list_words.append(words)
```

In [128]:

```
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: 45937
Number of words after taking IQR: 28110
```

Tokenize:

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

In [130]:

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

In [132]:

```
# 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)

emb_matrix_2 = zeros((word_count_2, emb_dim))
for word, i in t_2.word_index.items():
        emb_vec_2 = glove_words.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: 28111
Shape of train weight matrix: (28111, 300)
```

Padding document

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

In [134]:

```
from keras.preprocessing.sequence import pad_sequences

pad_tr_2 = pad_sequences(tr_seq_2, maxlen = seq_len, padding = 'post', truncating = 'po
st')
pad_cv_2 = pad_sequences(cv_seq_2, maxlen = seq_len, padding = 'post', truncating = 'po
st')
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

In [0]:

```
from keras.layers import Embedding, Dense, Flatten, Input, LSTM, Dropout, BatchNormaliz
ation, concatenate


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

In [0]:

```
from keras.layers import concatenate

con_lay_2 = concatenate([flatten_1_2, flatten_tea_pre, flatten_sch, flatten_pro_gra, fl
atten_pro_sub, flatten_pro_sub_1, emb_num])
```

Keras model:

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

In [0]:

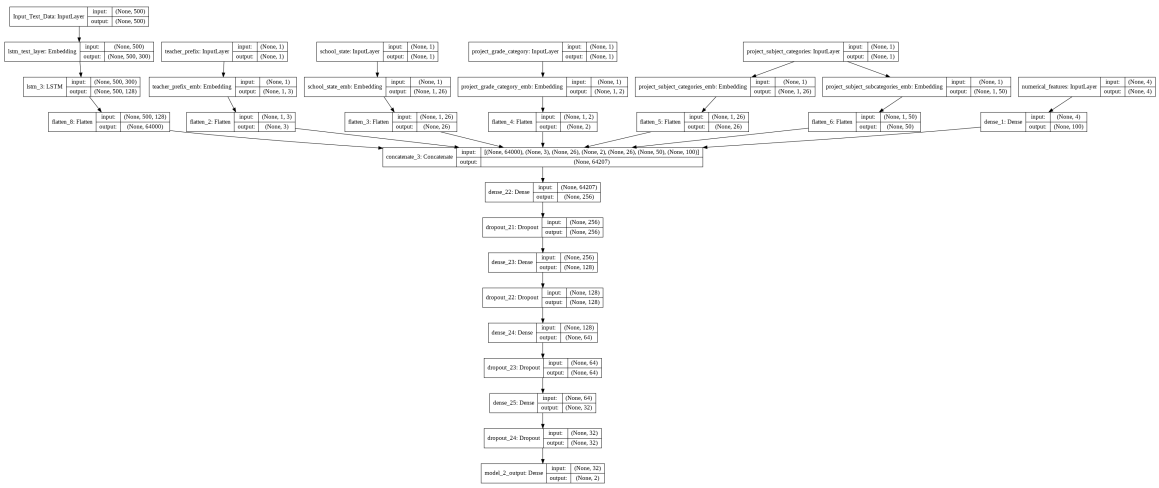
```
from keras.models import Model

# Layer 1
m_2 = Dense(256, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(con_l
ay_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

In [160]:

```
# 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')
```

Out[160]:



Getting all data into list.

In [0]:

```
# 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_gra_encode, tr_ss]

# 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_gra_encode, cv_ss]

# Test data
test_data_2 = [pad_test_2, test_tea_pre_encode, test_sch_encode, test_pro_sub_encode, test_sub_1_encoder, test_pro_gra_encode, test_ss]
```

In [0]:

```
# Chaning 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_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/ (https://keras.io/callbacks/)

In [0]:

```
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_loss', mode = "min", patience = 5,
                            verbose = 1, restore_best_weights = True)

# TensorBoard is a visualization tool provided with TensorFlow.
tensorboard_2 = TensorBoard(log_dir = "drive/My Drive/LSTM on Donors/graph_2",
                         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**

In [0]:

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

In [0]:

```
from keras.optimizers import Adam, RMSprop

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

Fitting model and callback to visualize model

In [179]:

```
history_2 = model_2.fit(tr_data_2, y_tr_data_2, batch_size = 512,
                        epochs = 10, validation_data = (cv_data_2, y_cv_data_2), verbos
e = 1,
                        callbacks = callback_2)
```

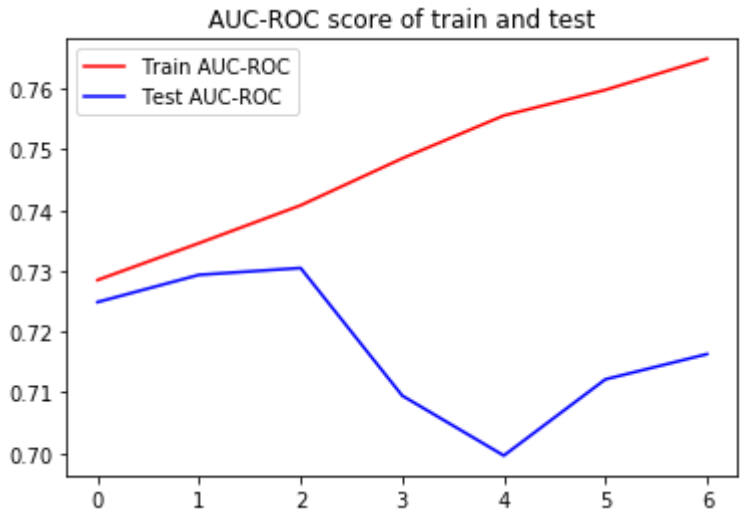
```
Train on 65548 samples, validate on 21850 samples
Epoch 1/10
65548/65548 [==============================] - 165s 3ms/step - loss: 0.454
0 - auroc: 0.7285 - val_loss: 0.4940 - val_auroc: 0.7249

Epoch 00001: val_loss improved from 0.51418 to 0.49400, saving model to mo
del_2.h5
Epoch 2/10
65548/65548 [==============================] - 165s 3ms/step - loss: 0.452
7 - auroc: 0.7346 - val_loss: 0.4765 - val_auroc: 0.7294

Epoch 00002: val_loss improved from 0.49400 to 0.47649, saving model to mo
del_2.h5
Epoch 3/10
65548/65548 [==============================] - 165s 3ms/step - loss: 0.450
4 - auroc: 0.7408 - val_loss: 0.4849 - val_auroc: 0.7305

Epoch 00003: val_loss did not improve from 0.47649
Epoch 4/10
65548/65548 [==============================] - 165s 3ms/step - loss: 0.445
7 - auroc: 0.7485 - val_loss: 0.5639 - val_auroc: 0.7095

Epoch 00004: val_loss did not improve from 0.47649
Epoch 5/10
65548/65548 [==============================] - 165s 3ms/step - loss: 0.445
0 - auroc: 0.7556 - val_loss: 0.5674 - val_auroc: 0.6996

Epoch 00005: val_loss did not improve from 0.47649
Epoch 6/10
65548/65548 [==============================] - 165s 3ms/step - loss: 0.444
8 - auroc: 0.7598 - val_loss: 0.4965 - val_auroc: 0.7122

Epoch 00006: val_loss did not improve from 0.47649
Epoch 7/10
65548/65548 [==============================] - 165s 3ms/step - loss: 0.442
2 - auroc: 0.7650 - val_loss: 0.5127 - val_auroc: 0.7163

Epoch 00007: val_loss did not improve from 0.47649
Restoring model weights from the end of the best epoch
Epoch 00007: early stopping
```

Evaluating test data

In [180]:

```
# 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'], 'r')
plt.plot(history_2.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 [==============================] - 21s 942us/step
Test Loss: 0.4769316001587656
Test ROC-AUC score: 0.7271611909671221
```



Observation:

- Test loss - 0.476
- Test AUC-ROC - 0.727

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

In [0]:

```
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, stratify = y)

# 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_st
ate = 42, stratify = y_train)
```

In [0]:

```
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, 10)
Shape of x_cv: (21850, 10)
Shape of x_test: (21850, 10)
Shape of y_tr: (65548,)
Shape of y_cv: (21850,)
Shape of y_test: (21850,)
```

In [0]:

```
# 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
```

In [0]:

```
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)

print('-'*22)
print("Shapes in 2 dimension.")
print('-'*22)
print("Train shape:", df_cn_tr.shape)
print("CV shape:", df_cn_cv.shape)
print("Test shape:", df_cn_te.shape, '\n')


print('-'*22)
print("Shapes in 3 dimension.")
print('-'*22)
print("Train shape:", tr_exp.shape)
print("CV shape:", cv_exp.shape)
print("Test shape:", te_exp.shape)
```

```
----------------------
Shapes in 2 dimension.
----------------------
Train shape: (65548, 9)
CV shape: (21850, 9)
Test shape: (21850, 9)

----------------------
Shapes in 3 dimension.
----------------------
Train shape: (65548, 9, 1)
CV shape: (21850, 9, 1)
Test shape: (21850, 9, 1)
```

Getting all data into a list

In [0]:

```
# Concatinating 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]
```

In [0]:

```
# 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

In [0]:

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

# 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)

# 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)
```

Concatinating LSTM output and Conv1D output

In [0]:

```
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)

In [0]:

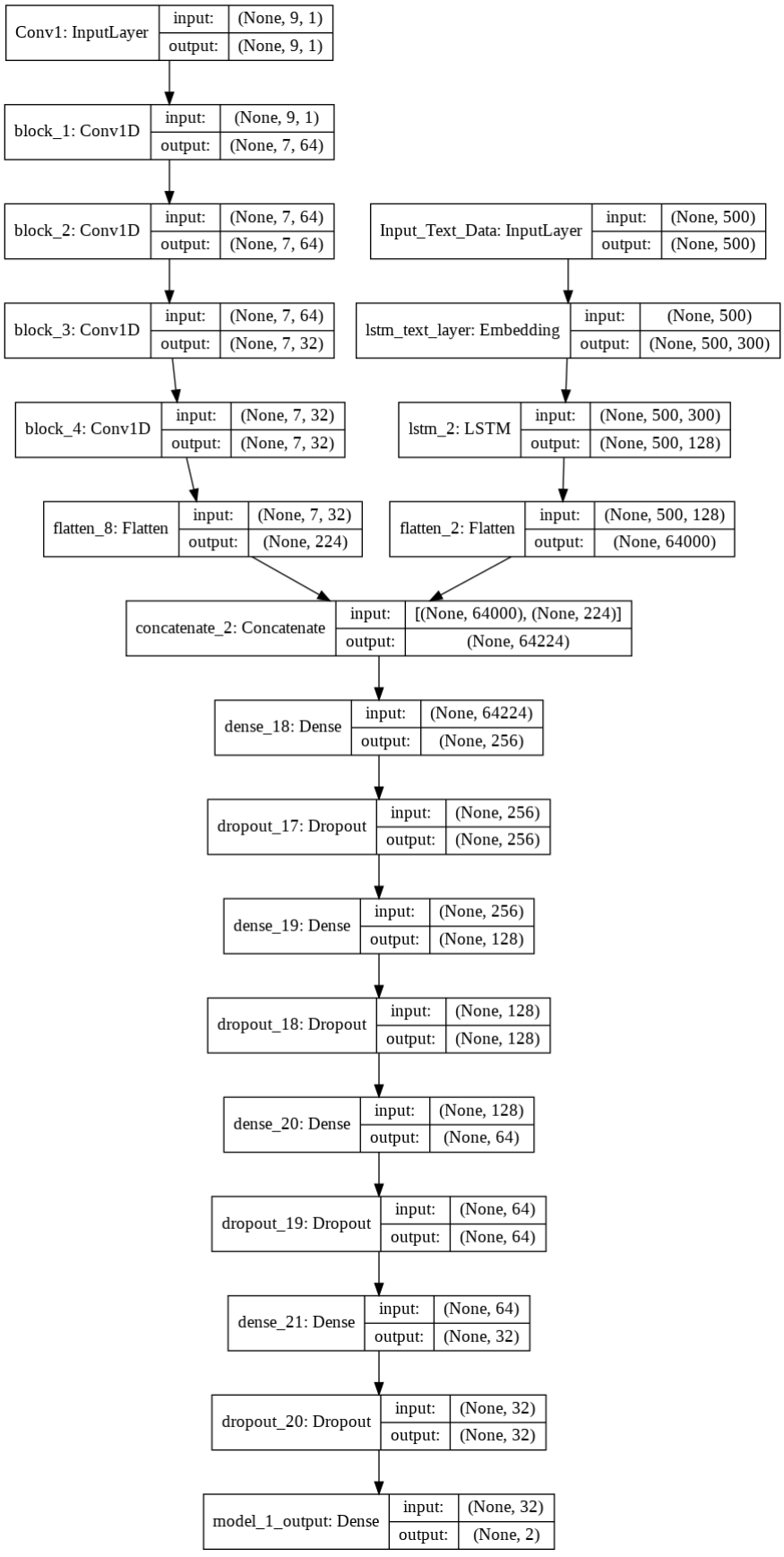
```
from keras.models import Model

# Layer 1
m_3 = Dense(256, activation = 'relu', kernel_regularizer = regularizers.l2(0.01))(con_l
ay_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

In [0]:

```
# 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')
```

Out[0]:


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_8: 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_2: LSTM | input: (None, 500, 300) | output: (None, 500, 128)
flatten_2: Flatten | input: (None, 500, 128) | output: (None, 64000)
concatenate_2: Concatenate | input: [(None, 64000), (None, 224)] | output: (None, 64224)
dense_18: Dense | input: (None, 64224) | output: (None, 256)
dropout_17: Dropout | input: (None, 256) | output: (None, 256)
dense_19: Dense | input: (None, 256) | output: (None, 128)
dropout_18: Dropout | input: (None, 128) | output: (None, 128)
dense_20: Dense | input: (None, 128) | output: (None, 64)
dropout_19: Dropout | input: (None, 64) | output: (None, 64)
dense_21: Dense | input: (None, 64) | output: (None, 32)
dropout_20: Dropout | input: (None, 32) | output: (None, 32)
model_1_output: Dense | input: (None, 32) | output: (None, 2)

Creating Callback with Checkpoint, EarlyStopping and Tensorboard

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

In [0]:

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

# 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_loss', mode = "min", patience = 5,
                            verbose = 1, restore_best_weights = True)

# TensorBoard is a visualization tool provided with TensorFlow.
tensorboard_3 = TensorBoard(log_dir = "drive/My Drive/LSTM on Donors/graph_3",
                         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**

In [0]:

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

In [0]:

```
from keras.optimizers import Adam, RMSprop

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

Fitting model and callback to visualize model

In [0]:

```
history_3 = model_3.fit(tr_data_3, y_tr_data_3, batch_size = 512,
                        epochs = 5, validation_data = (cv_data_3, y_cv_data_3), verbose
= 1,
                        callbacks = callback_3)
```

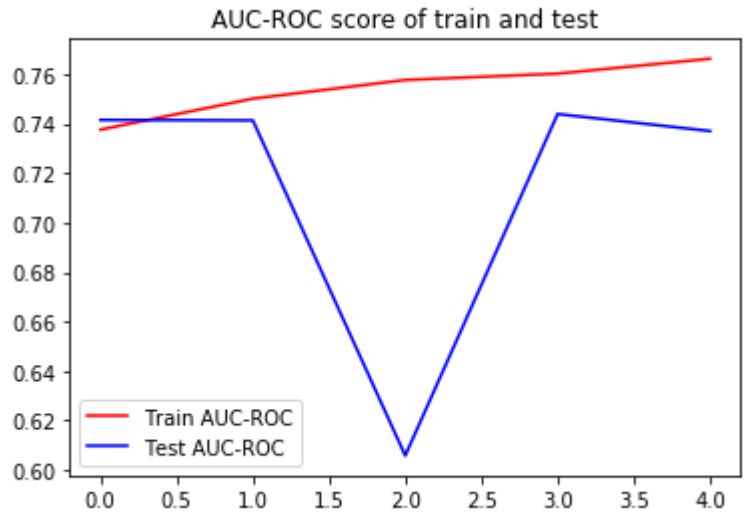
```
Train on 65548 samples, validate on 21850 samples
Epoch 1/5
65548/65548 [==============================] - 170s 3ms/step - loss: 1.531
0 - auroc: 0.7376 - val_loss: 0.6717 - val_auroc: 0.7415

Epoch 00001: val_loss improved from inf to 0.67174, saving model to model_
3.h5
Epoch 2/5
65548/65548 [==============================] - 166s 3ms/step - loss: 0.506
3 - auroc: 0.7501 - val_loss: 0.5018 - val_auroc: 0.7413

Epoch 00002: val_loss improved from 0.67174 to 0.50182, saving model to mo
del_3.h5
Epoch 3/5
65548/65548 [==============================] - 166s 3ms/step - loss: 0.453
6 - auroc: 0.7576 - val_loss: 0.6246 - val_auroc: 0.6058

Epoch 00003: val_loss did not improve from 0.50182
Epoch 4/5
65548/65548 [==============================] - 166s 3ms/step - loss: 0.447
4 - auroc: 0.7602 - val_loss: 0.5775 - val_auroc: 0.7439

Epoch 00004: val_loss did not improve from 0.50182
Epoch 5/5
65548/65548 [==============================] - 165s 3ms/step - loss: 0.438
9 - auroc: 0.7662 - val_loss: 0.5129 - val_auroc: 0.7370

Epoch 00005: val_loss did not improve from 0.50182
```

Evaluating test data

In [0]:

```
# 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'], 'r')
plt.plot(history_3.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 [==============================] - 20s 927us/step
Test Loss: 0.5117486406381944
Test ROC-AUC score: 0.7382677115625296
```



Observation:

- Test loss - 0.51
- Test AUC-ROC - 0.738

Pretty Table

In [2]:

```
from prettytable import PrettyTable

a = PrettyTable()

a.field_names = ['S.No', 'Model', 'Optimizer', 'Dropout', 'Test Loss', 'Test AUC-ROC']

a.add_row([1, 'Model- 1', 'rmsprop', 0.3, 0.47, 0.752])
a.add_row([2, 'Model- 2', 'rmsprop', 0.3, 0.47, 0.727])
a.add_row([3, 'Model- 3', 'rmsprop', 0.3, 0.51, 0.74])

print(a.get_string(title = "LSTM on Donors Result"))
```

```
+------+----------+-----------+---------+-----------+--------------+
| S.No |  Model   | Optimizer | Dropout | Test Loss | Test AUC-ROC |
+------+----------+-----------+---------+-----------+--------------+
|  1   | Model- 1 |  rmsprop  |   0.3   |    0.47   |    0.752     |
|  2   | Model- 2 |  rmsprop  |   0.3   |    0.47   |    0.727     |
|  3   | Model- 3 |  rmsprop  |   0.3   |    0.51   |     0.74     |
+------+----------+-----------+---------+-----------+--------------+
```

CONCLUSION:

a)Model - 1 Optimizer - 'rmsprop' and dropout - 0.3 Test loss - 0.47 Test AUC-ROC - 0.752

b)Model - 2 Optimizer - 'rmsprop' and dropout - 0.3 Test loss - 0.47 Test AUC-ROC - 0.727

c)Model - 3 Optimizer - 'rmsprop' and dropout - 0.3 Test loss - 0.51 Test AUC-ROC - 0.74

Model - 1 performed well when compared to model - 2 and model - 3. Model - 1 resulted with test loss of 0.47 and test AUC-ROC of 0.752.