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	
project_id	A unique identifier for the proposed project. **Examp**
project_title	Title of the proje • Art Will Make • First
project_grade_category	Grade level of students for which the project is targeted. One c enum • Gra • • • G
project_subject_categories	One or more (comma-separated) subject categories for the p following enumerated • Applie • Car • Healt • Histor • Literacy • Math • Music • Spe •
	• Music • Literacy & Language, Math
school_state	State where school is located (Two-letter U. (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#F
project_subject_subcategories	One or more (comma-separated) subject subcategories • • Literature & Writing, Socia
project_resource_summary	An explanation of the resources needed for the proj • My students need hands on literacy materials sens
project_essay_1	First app
project_essay_2	Second app
project_essay_3	Third app
project_essay_4	Fourth app
project_submitted_datetime	Datetime when project application was submitted. **Example:** 12
teacher_id	A unique identifier for the teacher of the proposed pro bdf8baa8fedef6bfeec7ae

Feature	
teacher_prefix	Teacher's title. One of the following enum • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [ ]:

```
%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 chart_studio 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 [6]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [7]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.head(2)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_pr
efix' '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']
```

Out[7]:

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

In [8]:

```
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[8]:

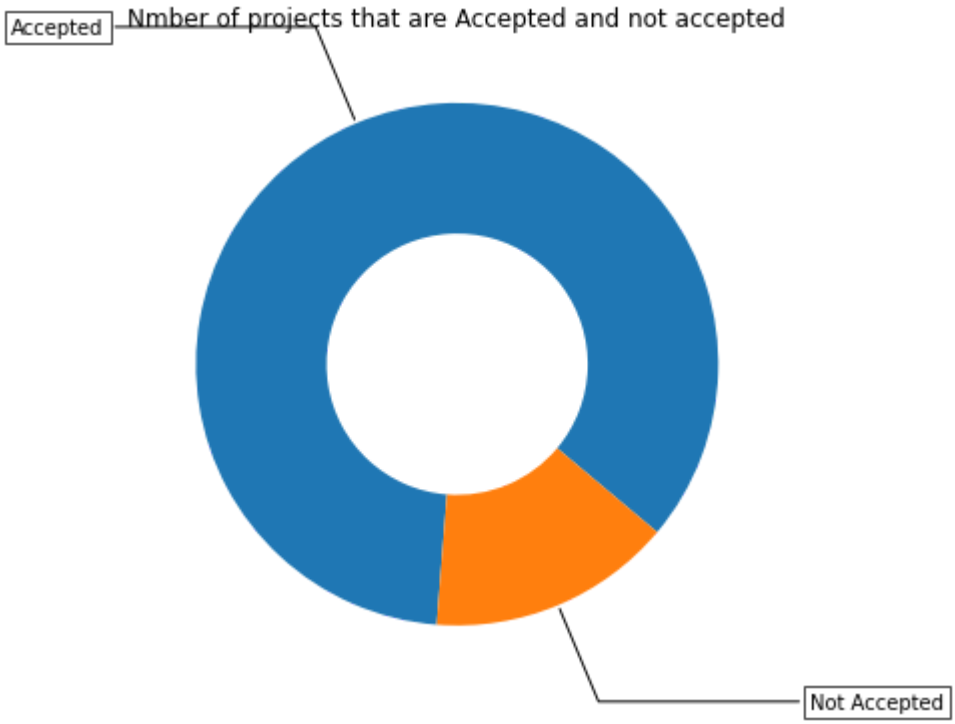
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

2. Data Analysis

In [11]:

```
# this code is taken from
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#
sphx-glr-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  92706 , ( 84.85830
404217927 %)
Number of projects thar are not approved for funding  16542 , ( 15.1
41695957820739 %)
```



2.1 Univariate Analysis: School State

In [ ]:

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

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].apply(np.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 [13]:

```
# 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       0.800000
7          DC       0.802326
43         TX       0.813142
26         MT       0.816327
18         LA       0.831245
==================================================
States with highest % approvals
   state_code  num_proposals
30         NH       0.873563
35         OH       0.875152
47         WA       0.876178
28         ND       0.888112
8          DE       0.897959
```

In [18]:

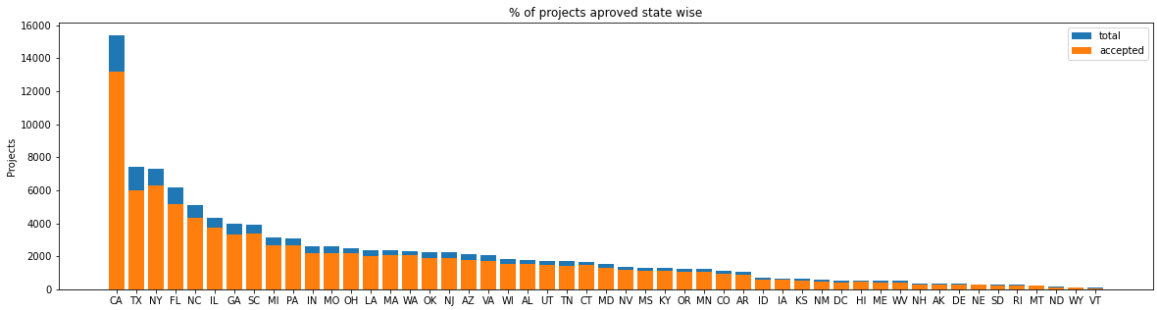
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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 [24]:

```
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 [25]:

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



	school_state	project_is_approved	total	Avg
4	CA	13205	15388	0.858136
43	TX	6014	7396	0.813142
34	NY	6291	7318	0.859661
9	FL	5144	6185	0.831690
27	NC	4353	5091	0.855038

==========

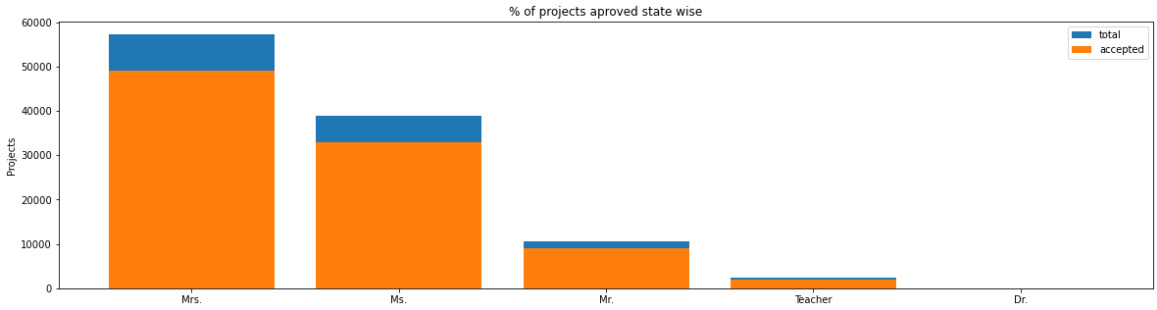
	school_state	project_is_approved	total	Avg
39	RI	243	285	0.852632
26	MT	200	245	0.816327
28	ND	127	143	0.888112
50	WY	82	98	0.836735
46	VT	64	80	0.800000

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

2.2 Univariate Analysis: teacher_prefix

In [26]:

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

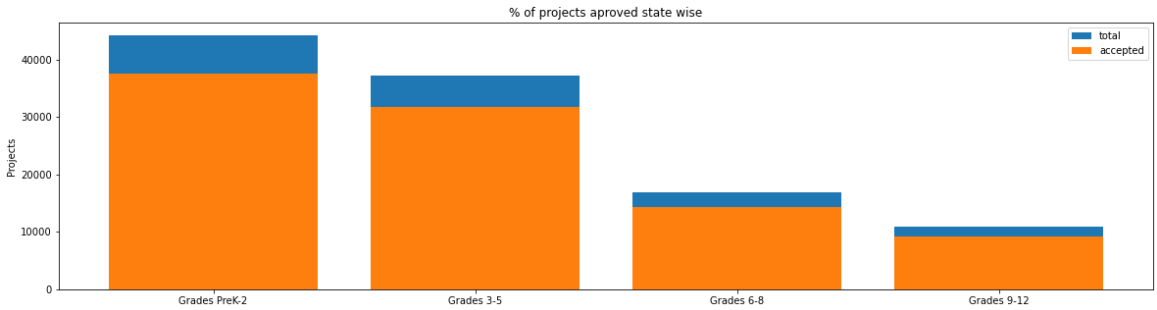



```
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.                    9     13  0.692308
==================================================
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.                    9     13  0.692308
```

2.3 Univariate Analysis: project_grade_category

In [27]:

```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approve
d', top=False)
```




```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                37536  44225  0.848751
0             Grades 3-5                31729  37137  0.854377
1             Grades 6-8                14258  16923  0.842522
2            Grades 9-12                 9183  10963  0.837636
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                37536  44225  0.848751
0             Grades 3-5                31729  37137  0.854377
1             Grades 6-8                14258  16923  0.842522
2            Grades 9-12                 9183  10963  0.837636
```

2.4 Univariate Analysis: project_subject_categories

In [28]:

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

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

In [29]:

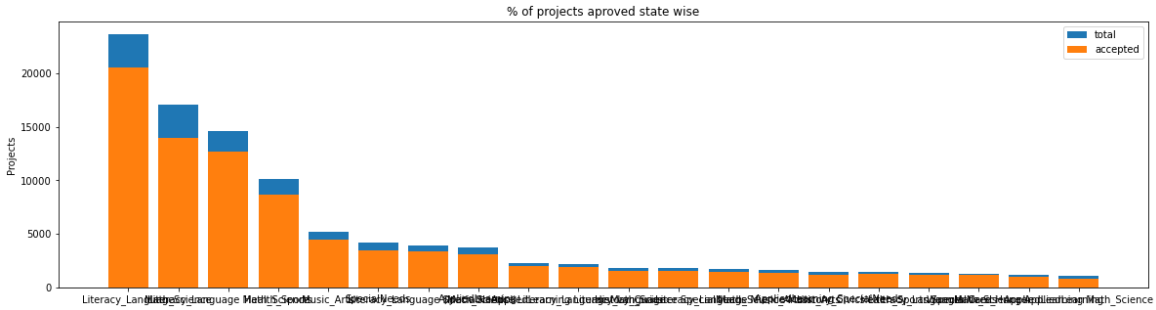
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[29]:

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

In [30]:

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



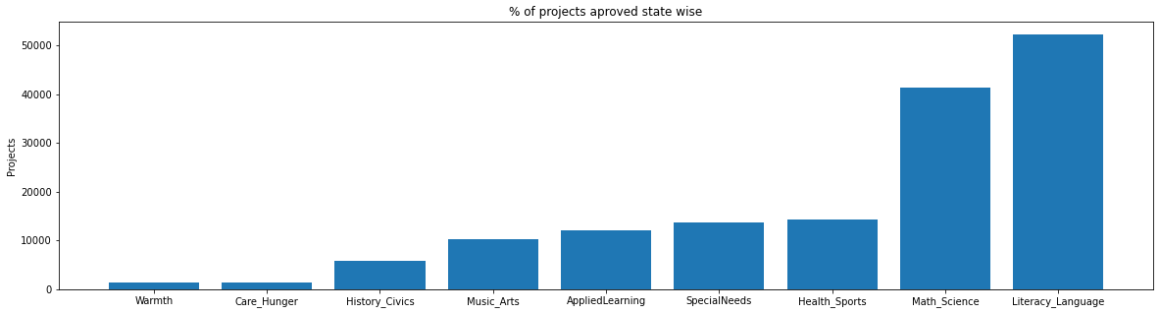

```
                clean_categories  project_is_approved  total
Avg
24              Literacy_Language                20520  23655  0.86
7470
32                   Math_Science                13991  17072  0.81
9529
28  Literacy_Language Math_Science                12725  14636  0.86
9432
8                   Health_Sports                 8640  10177  0.84
8973
40                     Music_Arts                 4429   5180  0.85
5019
==================================================
                  clean_categories  project_is_approved  total
Avg
19  History_Civics Literacy_Language                1271   1421  0.
894441
14        Health_Sports SpecialNeeds                1215   1391  0.
873472
50               Warmth Care_Hunger                 1212   1309  0.
925898
33      Math_Science AppliedLearning                1019   1220  0.
835246
4       AppliedLearning Math_Science                 855   1052  0.
812738
```

In [31]:

```
# 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 [32]:

```
# 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 [33]:

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

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

2.5 Univariate Analysis: project_subject_subcategories

In [34]:

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

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

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

In [35]:

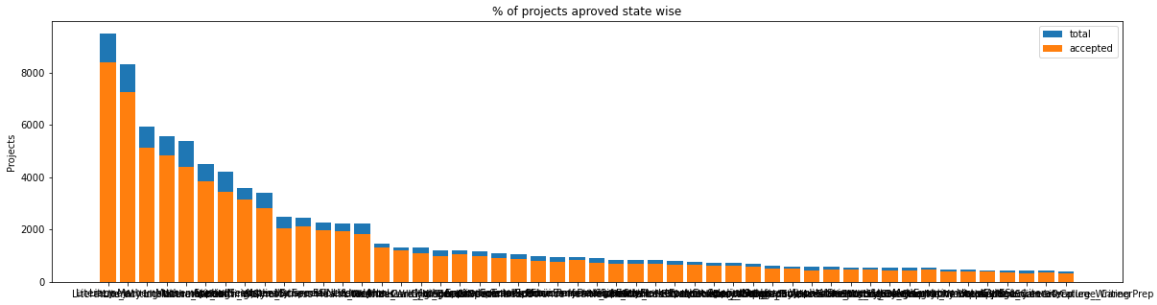
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[35]:

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

In [36]:

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



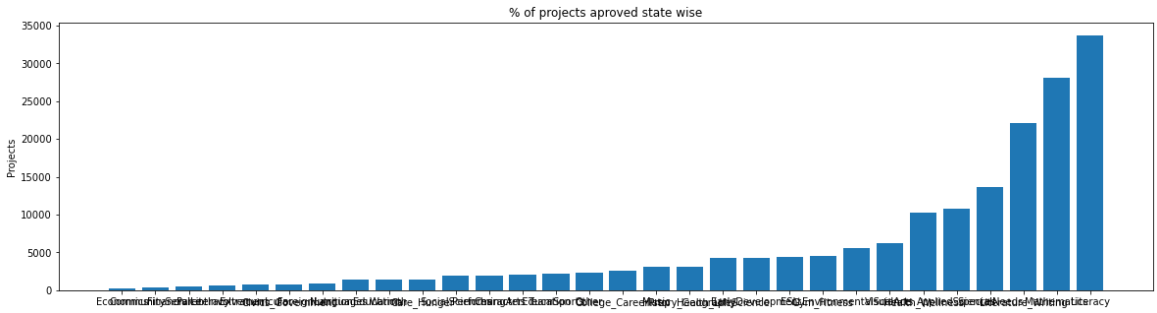

```
                clean_subcategories  project_is_approved  total
Avg
317                         Literacy                 8371   9486  0.8
82458
319             Literacy Mathematics                 7260   8325  0.8
72072
331  Literature_Writing Mathematics                 5140   5923  0.8
67803
318     Literacy Literature_Writing                 4823   5571  0.8
65733
342                      Mathematics                 4385   5379  0.8
15207
==================================================
                    clean_subcategories  project_is_approved  total
Avg
196       EnvironmentalScience Literacy                  389    444
0.876126
127                                ESL                  349    421
0.828979
79                  College_CareerPrep                  343    421
0.814727
17   AppliedSciences Literature_Writing                 361    420
0.859524
3    AppliedSciences College_CareerPrep                 330    405
0.814815
```

In [37]:

```
# 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 [38]:

```
# 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 [39]:

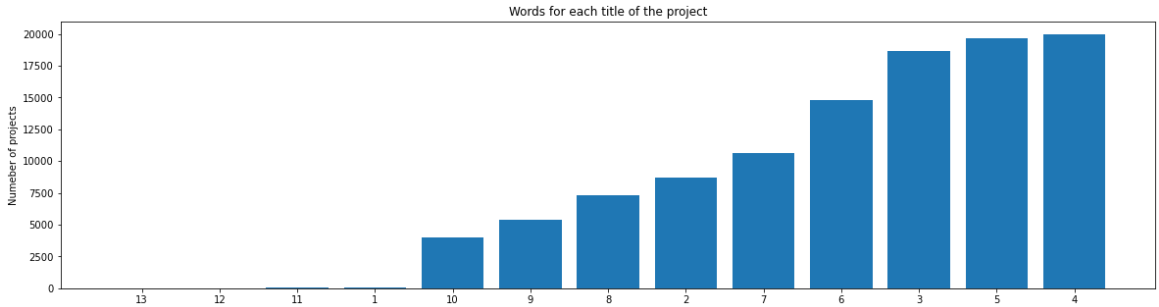
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
NutritionEducation   :      1355
Warmth               :      1388
Care_Hunger          :      1388
SocialSciences       :      1920
PerformingArts       :      1961
CharacterEducation   :      2065
TeamSports           :      2192
Other                :      2372
College_CareerPrep   :      2568
Music                :      3145
History_Geography    :      3171
Health_LifeScience   :      4235
EarlyDevelopment     :      4254
ESL                  :      4367
Gym_Fitness          :      4509
EnvironmentalScience :      5591
VisualArts           :      6278
Health_Wellness      :     10234
AppliedSciences      :     10816
SpecialNeeds         :     13642
Literature_Writing   :     22179
Mathematics          :     28074
Literacy             :     33700
```

2.6 Univariate Analysis: Text features (Title)

In [40]:

```
#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()
```

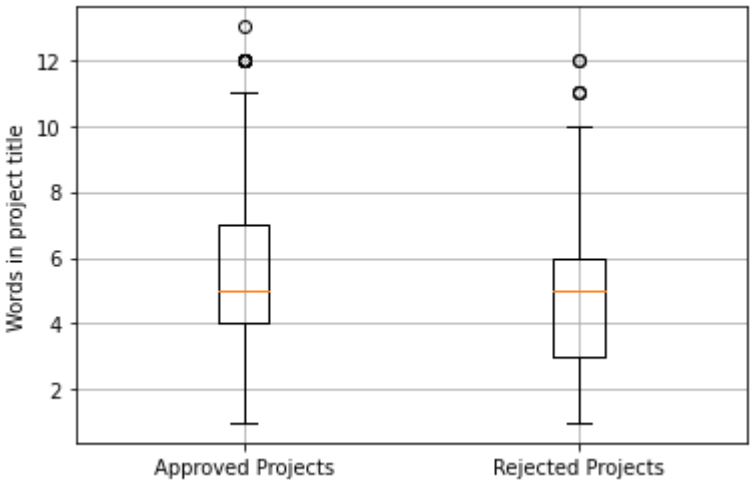


In [41]:

```
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
```

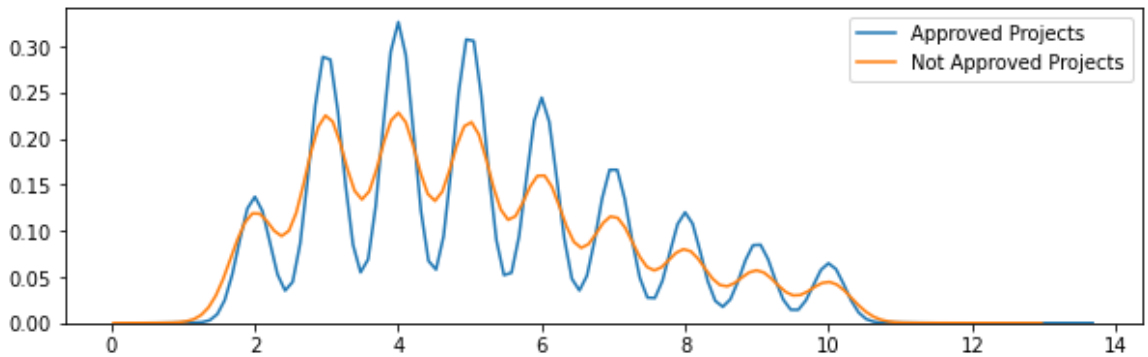
In [42]:

```
# 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 [43]:

```
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()
```



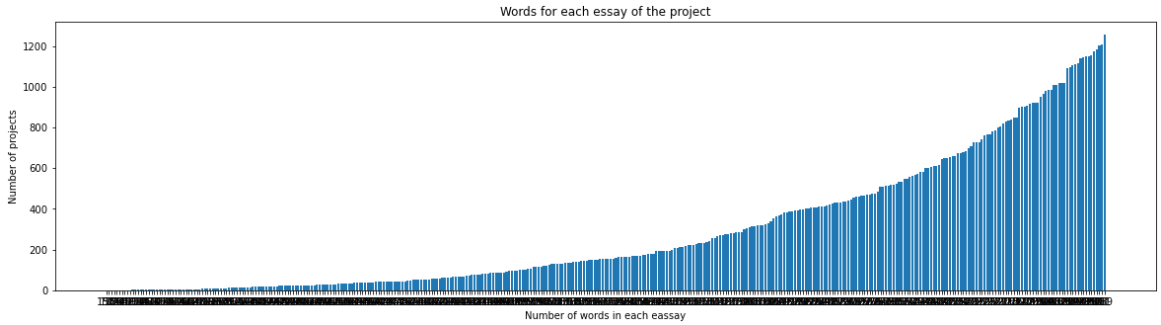
2.7 Univariate Analysis: Text features (Project Essay's)

In [44]:

```
# 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)
```

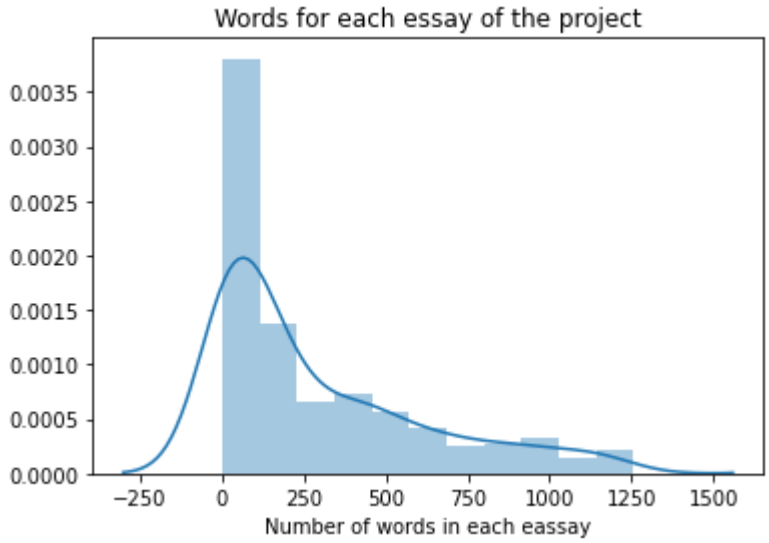
In [45]:

```
#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 [46]:

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

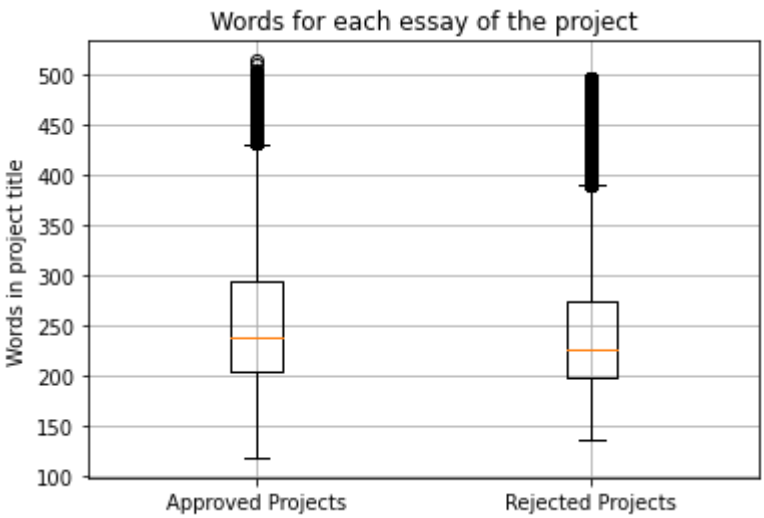


In [49]:

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

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

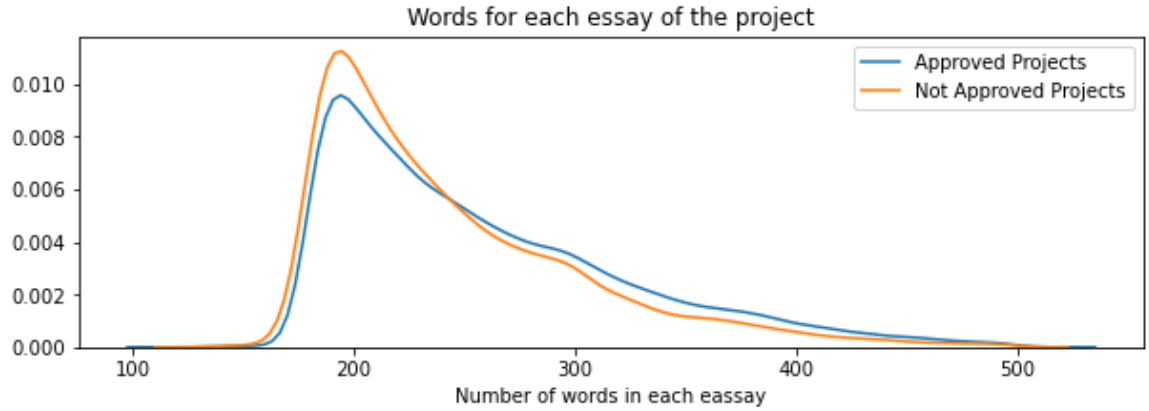
In [50]:

```
# 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 [51]:

```
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()
```



2.8 Univariate Analysis: Cost per project

In [64]:

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

Out[64]:

	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 [65]:

```
# 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[65]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [68]:

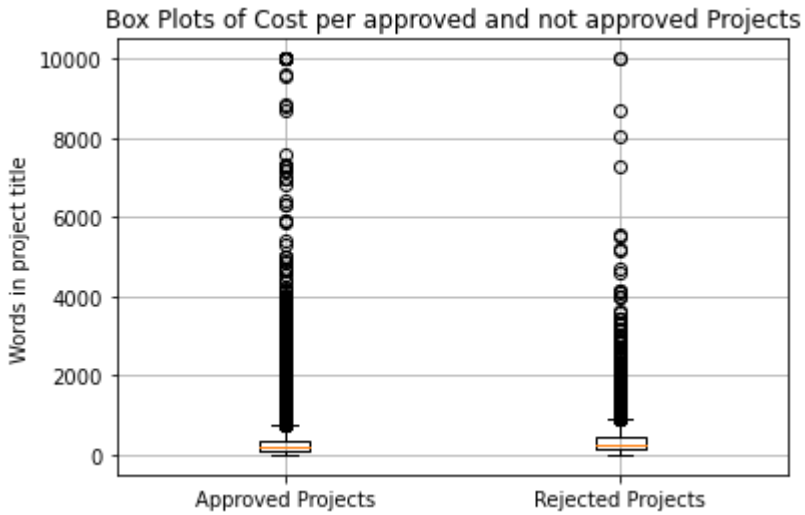
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [69]:

```
approved_price = project_data[project_data['project_is_approved']==1]['price'].v
alues

rejected_price = project_data[project_data['project_is_approved']==0]['price'].v
alues
```

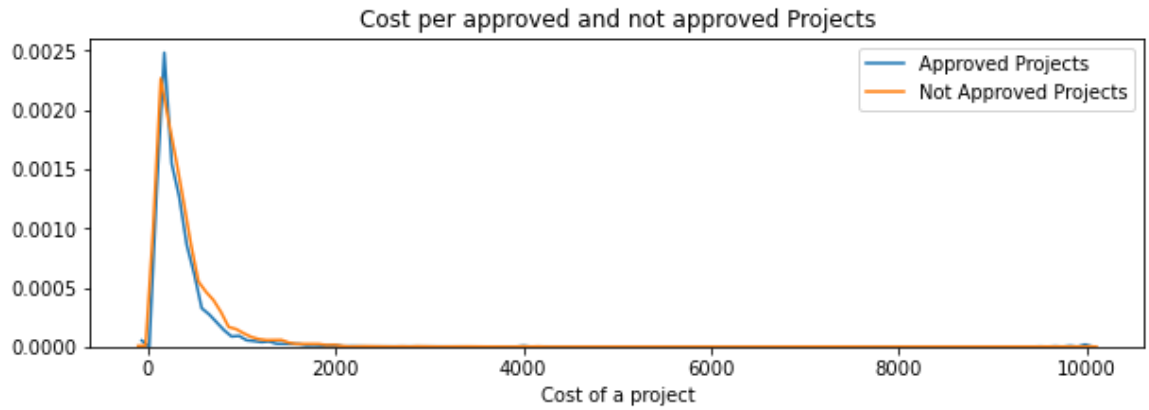
In [70]:

```
# 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 [71]:

```
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 [74]:

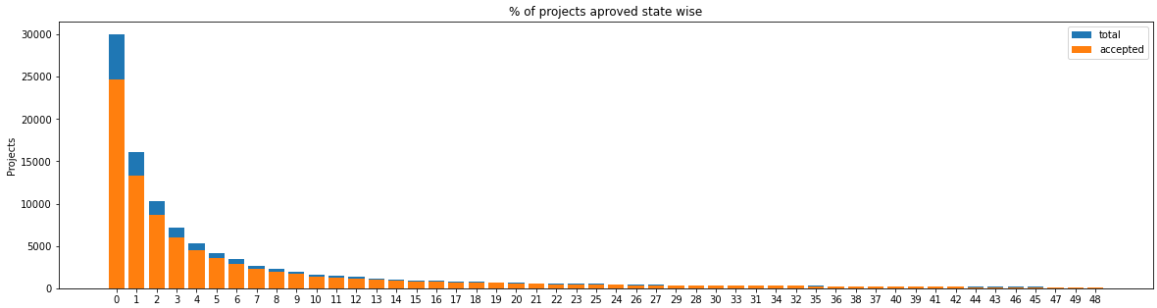
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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.perce
ntile(rejected_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         |
+------------+--------------------+------------------------+
```

2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [84]:

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




```
   teacher_number_of_previously_posted_projects  project_is_approved
total  \
0                                             0                24652
30014
1                                             1                13329
16058
2                                             2                 8705
10350
3                                             3                 5997
7110
4                                             4                 4452
5266

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
    teacher_number_of_previously_posted_projects  project_is_approve
d  total  \
46                                           46                  14
9    164
45                                           45                  14
1    153
47                                           47                  12
9    144
49                                           49                  12
8    143
48                                           48                  13
5    140

         Avg
46  0.908537
45  0.921569
47  0.895833
49  0.895105
48  0.964286
```

Observations :

1. We observe that Maximum number of approved projects have been submitted by teachers with no prior project proposals.
2. New talents are well appreciated hence it is not mandatory for a teacher to have proposed any project prior.
3. We can also notice that very few teachers have higher number of prior projects and the rate of approval is slightly higher for them.

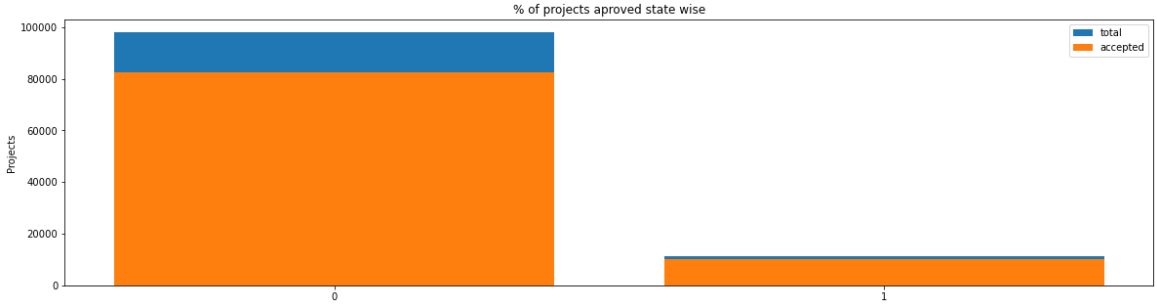
2.10 Univariate Analysis: project_resource_summary

In [104]:

```
#list of project_resource_summary
summaries = []
for s in project_data["project_resource_summary"] :
    summaries.append(s)

#check the presence of the numerical digits in each summary {presence= 1; absenc
e = 0}
numeric_summaries = {}
for i in range(len(summaries)):
    for s in summaries[i].split():
        if s.isdigit() :
            numeric_summaries[i] = 1
            break
        else :
            numeric_summaries[i] = 0

project_data['numeric_summaries'] = numeric_summaries.values()
```

In [105]:

```
univariate_barplots(project_data, 'numeric_summaries', 'project_is_approved', to
p=2)
```




```
   numeric_summaries  project_is_approved  total       Avg
0                  0                82562  98011  0.842375
1                  1                10144  11237  0.902732
==================================================
   numeric_summaries  project_is_approved  total       Avg
0                  0                82562  98011  0.842375
1                  1                10144  11237  0.902732
```

Observations :

1. The project summaries containing numeric values is very less but those projects have a very high acceptance rate of 90%. Mentioning the requirements of certain products suggest clarity in the proposals and hence get higher chance of approval.
2. But most of the project_resource_summary do not have numeric values.

3. Data Preprocessing

In [164]:

```
# only 50000 project_data has taken
processed_data = pd.read_csv('train_data.csv', nrows=50000)
resource_data = pd.read_csv('resources.csv')
processed_data.shape
```

Out[164]:

```
(50000, 17)
```

3.1. Preprocessing Categorical Features: project_grade_category

In [165]:

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

Out[165]:

```
Grades PreK-2    20316
Grades 3-5       16968
Grades 6-8        7750
Grades 9-12       4966
Name: project_grade_category, dtype: int64
```

In [166]:

```
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-on-other-column-value
#remove the spaces, replace the '-' with '_' and convert all the letters to smal
l
processed_data['project_grade_category'] = processed_data['project_grade_categor
y'].str.replace(' ','_')
processed_data['project_grade_category'] = processed_data['project_grade_categor
y'].str.replace('-','_')
processed_data['project_grade_category'] = processed_data['project_grade_categor
y'].str.lower()
processed_data['project_grade_category'].value_counts()
```

Out[166]:

```
grades_prek_2    20316
grades_3_5       16968
grades_6_8        7750
grades_9_12       4966
Name: project_grade_category, dtype: int64
```

3.2.Preprocessing Categorical Features: project_subject_categories

In [168]:

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

Out[168]:

```
Literacy & Language                           10927
Math & Science                                 7695
Literacy & Language, Math & Science            6705
Health & Sports                                4700
Music & The Arts                               2358
Special Needs                                  1913
Literacy & Language, Special Needs             1814
Applied Learning                               1719
Math & Science, Literacy & Language            1041
Applied Learning, Literacy & Language          1018
Math & Science, Special Needs                   871
History & Civics                                839
Literacy & Language, Music & The Arts           794
Math & Science, Music & The Arts                755
Applied Learning, Special Needs                 672
History & Civics, Literacy & Language           651
Health & Sports, Special Needs                  633
Warmth, Care & Hunger                           606
Math & Science, Applied Learning                565
Applied Learning, Math & Science                477
Health & Sports, Literacy & Language            369
Literacy & Language, History & Civics           363
Applied Learning, Music & The Arts              360
Math & Science, History & Civics                282
Literacy & Language, Applied Learning           280
Applied Learning, Health & Sports               264
Math & Science, Health & Sports                 187
History & Civics, Math & Science                171
Special Needs, Music & The Arts                 140
History & Civics, Music & The Arts              135
Health & Sports, Math & Science                 118
History & Civics, Special Needs                 103
Health & Sports, Applied Learning                99
Applied Learning, History & Civics               78
Music & The Arts, Special Needs                  67
Health & Sports, Music & The Arts                66
Literacy & Language, Health & Sports             33
History & Civics, Applied Learning               25
Health & Sports, History & Civics                25
Special Needs, Health & Sports                   14
Health & Sports, Warmth, Care & Hunger           12
Music & The Arts, Health & Sports                10
Music & The Arts, History & Civics                9
Applied Learning, Warmth, Care & Hunger           8
History & Civics, Health & Sports                 8
Math & Science, Warmth, Care & Hunger             7
Special Needs, Warmth, Care & Hunger              6
Music & The Arts, Applied Learning                4
Literacy & Language, Warmth, Care & Hunger        3
Music & The Arts, Warmth, Care & Hunger           1
Name: project_subject_categories, dtype: int64
```

In [169]:

```
# remove 'the' and spaces ; replace '&' with '_', and ',' with '_' ; convert all
the letters to small
processed_data['project_subject_categories'] = processed_data['project_subject_c
ategories'].str.replace(' The ','')
processed_data['project_subject_categories'] = processed_data['project_subject_c
ategories'].str.replace(' ','')
processed_data['project_subject_categories'] = processed_data['project_subject_c
ategories'].str.replace('&','_')
processed_data['project_subject_categories'] = processed_data['project_subject_c
ategories'].str.replace(',','_')
processed_data['project_subject_categories'] = processed_data['project_subject_c
ategories'].str.lower()
processed_data['project_subject_categories'].value_counts()
```

Out[169]:

```
literacy_language                        10927
math_science                              7695
literacy_language_math_science            6705
health_sports                             4700
music_arts                                2358
specialneeds                              1913
literacy_language_specialneeds            1814
appliedlearning                           1719
math_science_literacy_language            1041
appliedlearning_literacy_language         1018
math_science_specialneeds                  871
history_civics                             839
literacy_language_music_arts               794
math_science_music_arts                    755
appliedlearning_specialneeds               672
history_civics_literacy_language           651
health_sports_specialneeds                 633
warmth_care_hunger                         606
math_science_appliedlearning               565
appliedlearning_math_science               477
health_sports_literacy_language            369
literacy_language_history_civics           363
appliedlearning_music_arts                 360
math_science_history_civics                282
literacy_language_appliedlearning          280
appliedlearning_health_sports              264
math_science_health_sports                 187
history_civics_math_science                171
specialneeds_music_arts                    140
history_civics_music_arts                  135
health_sports_math_science                 118
history_civics_specialneeds                103
health_sports_appliedlearning               99
appliedlearning_history_civics              78
music_arts_specialneeds                     67
health_sports_music_arts                    66
literacy_language_health_sports             33
health_sports_history_civics                25
history_civics_appliedlearning              25
specialneeds_health_sports                  14
health_sports_warmth_care_hunger            12
music_arts_health_sports                    10
music_arts_history_civics                    9
history_civics_health_sports                 8
appliedlearning_warmth_care_hunger           8
math_science_warmth_care_hunger              7
specialneeds_warmth_care_hunger              6
music_arts_appliedlearning                   4
literacy_language_warmth_care_hunger         3
music_arts_warmth_care_hunger                1
Name: project_subject_categories, dtype: int64
```

3.3. Preprocessing Categorical Features: teacher_prefix

In [171]:

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

Out[171]:

```
Mrs.       26140
Ms.        17936
Mr.         4859
Teacher     1061
Dr.            2
Name: teacher_prefix, dtype: int64
```

In [172]:

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

```
True
number of nan values 2
```

In [173]:

```
# number of missing values are very less in number
# we can replace it with Mrs. as most of the projects are submitted by Mrs.
processed_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
processed_data['teacher_prefix'].value_counts()
```

Out[173]:

```
Mrs.       26142
Ms.        17936
Mr.         4859
Teacher     1061
Dr.            2
Name: teacher_prefix, dtype: int64
```

In [174]:

```
# Remove '.' and convert all the chars to small
processed_data['teacher_prefix'] = processed_data['teacher_prefix'].str.replace(
'.','')
processed_data['teacher_prefix'] = processed_data['teacher_prefix'].str.lower()
processed_data['teacher_prefix'].value_counts()
```

Out[174]:

```
mrs        26142
ms         17936
mr          4859
teacher     1061
dr             2
Name: teacher_prefix, dtype: int64
```

3.4. Preprocessing Categorical Features: project_subject_subcategories

In [176]:

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

Out[176]:

```
Literacy                                  4434
Literacy, Mathematics                     3833
Literature & Writing, Mathematics         2705
Literacy, Literature & Writing            2570
Mathematics                               2441
                                          ...
Character Education, Economics               1
Civics & Government, Extracurricular         1
Financial Literacy, Foreign Languages        1
Foreign Languages, Performing Arts           1
Community Service, Financial Literacy        1
Name: project_subject_subcategories, Length: 384, dtype: int64
```

In [177]:

```
# remove 'the' and spaces ; replace '&' with '_', and ',' with '_' ; convert all
the letters to small
processed_data['project_subject_subcategories'] = processed_data['project_subjec
t_subcategories'].str.replace(' The ','')
processed_data['project_subject_subcategories'] = processed_data['project_subjec
t_subcategories'].str.replace(' ','')
processed_data['project_subject_subcategories'] = processed_data['project_subjec
t_subcategories'].str.replace('&','_')
processed_data['project_subject_subcategories'] = processed_data['project_subjec
t_subcategories'].str.replace(',','_')
processed_data['project_subject_subcategories'] = processed_data['project_subjec
t_subcategories'].str.lower()
processed_data['project_subject_subcategories'].value_counts()
```

Out[177]:

```
literacy                              4434
literacy_mathematics                  3833
literature_writing_mathematics        2705
literacy_literature_writing           2570
mathematics                           2441
                                      ...
foreignlanguages_performingarts          1
economics_music                          1
financialliteracy_socialsciences         1
communityservice_music                   1
civics_government_esl                    1
Name: project_subject_subcategories, Length: 384, dtype: int64
```

3.5. Preprocessing Categorical Features: school_state

In [179]:

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

Out[179]:

```
CA    7024
NY    3393
TX    3320
FL    2839
NC    2340
IL    1967
SC    1830
GA    1828
MI    1468
PA    1419
OH    1180
IN    1171
MO    1166
WA    1103
LA    1094
MA    1076
OK    1074
NJ    1005
AZ     994
VA     916
WI     833
UT     792
AL     790
CT     774
TN     774
MD     668
NV     665
KY     614
MS     598
OR     577
MN     556
CO     538
AR     446
IA     306
ID     302
KS     285
DC     247
HI     239
NM     236
ME     222
WV     218
DE     155
AK     153
NE     144
SD     142
NH     141
RI     126
MT     106
ND      63
WY      51
VT      32
Name: school_state, dtype: int64
```

In [180]:

```
# convert all of them into small letters
processed_data['school_state'] = processed_data['school_state'].str.lower()
processed_data['school_state'].value_counts()
```

Out[180]:

```
ca    7024
ny    3393
tx    3320
fl    2839
nc    2340
il    1967
sc    1830
ga    1828
mi    1468
pa    1419
oh    1180
in    1171
mo    1166
wa    1103
la    1094
ma    1076
ok    1074
nj    1005
az     994
va     916
wi     833
ut     792
al     790
ct     774
tn     774
md     668
nv     665
ky     614
ms     598
or     577
mn     556
co     538
ar     446
ia     306
id     302
ks     285
dc     247
hi     239
nm     236
me     222
wv     218
de     155
ak     153
ne     144
sd     142
nh     141
ri     126
mt     106
nd      63
wy      51
vt      32
Name: school_state, dtype: int64
```

3.6. Preprocessing Text Features: project_title

In [124]:

```
# https://stackoverflow.com/a/47091490/4084039
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 [125]:

```
# 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', 'itse
lf', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
t', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
s', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becaus
e', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 't
hrough', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al
l', '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'v
e", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "d
idn'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', "should
n't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [182]:

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

Out[182]:

```
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 [183]:

```
print("printing some random reviews")
print(9, processed_data['project_title'].values[9])
print(34, processed_data['project_title'].values[34])
print(143, processed_data['project_title'].values[143])
```

```
printing some random reviews
9 Just For the Love of Reading--\r\nPure Pleasure
34 \"Have A Ball!!!\"
143 Ready To Go With Our MacBook Pro
```

In [137]:

```
# 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 [184]:

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

```
100%|██████████| 50000/50000 [00:03<00:00, 16417.37it/s]
```

In [185]:

```
print("printing some random reviews")
print(9, preprocessed_titles[9])
print(34, preprocessed_titles[34])
print(143, preprocessed_titles[143])
```

```
printing some random reviews
9 love reading pure pleasure
34 ball
143 ready go macbook pro
```

3.7. Preprocessing Text Features: essay

In [187]:

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

In [188]:

```
print("printing some random essay")
print(9, processed_data['essay'].values[9])
print('-'*50)
print(34, processed_data['essay'].values[34])
print('-'*50)
print(143, processed_data['essay'].values[143])
```

```
printing some random essay
9 Over 95% of my students are on free or reduced lunch.  I have a fe
w who are homeless, but despite that, they come to school with an ea
gerness to learn.  My students are inquisitive eager learners who  e
mbrace the challenge 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 of a book on a regular basis at home a
nd they don't travel to the public library.  \r\nIt is my duty as a
teacher to do all I can to provide each student an opportunity to su
cceed in every aspect of life. \r\nReading is Fundamental! My studen
ts will read these books over and over again while boosting their co
mprehension skills. These books will be used for read alouds, partne
r reading and for Independent reading. \r\nThey will engage in readi
ng to build their \"Love for Reading\" by reading for pure enjoymen
t. They will be introduced to some new authors as well as some old f
avorites. I want my students to be ready for the 21st Century and kn
ow the pleasure of holding a good hard back book in hand. There's no
thing like a good book to read!  \r\nMy students will soar in Readin
g, and more because of your consideration and generous funding contr
ibution. This will help build stamina and prepare for 3rd grade. Tha
nk you so much for reading our proposal!nannan
--------------------------------------------------
34 My students mainly come from extremely low-income families, and t
he majority of them come from homes where both parents work full tim
e. Most of my students are at school from 7:30 am to 6:00 pm (2:30 t
o 6:00 pm in the after-school program), and they all receive free an
d reduced meals for breakfast and lunch. \r\n\r\n\r\nI want my stude
nts to feel  as comfortable in my classroom as they do at home. Many
of my students take on multiple roles both at home as well as in sch
ool. They are sometimes the caretakers of younger siblings, cooks, b
abysitters, 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 positive
manner. As a result, I have a community of students who love helping
each other in and outside of the classroom. They consistently look f
or opportunities to support each other's learning in a kind and help
ful way.I am excited to be experimenting with alternative seating in
my classroom this school year. Studies have shown that giving studen
ts the option of where they sit in a classroom increases focus as we
ll as motivation.  \r\n\r\nBy allowing students choice in the classr
oom, they are able to explore and create in a welcoming environment.
Alternative classroom seating has been experimented with more freque
ntly in recent years. I believe (along with many others), that every
child learns differently. This does not only apply to how multiplica
tion is memorized, or a paper is written, but applies to the space i
n 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 w
as always, \"As long as you're learning, you can work wherever you w
ant!\" \r\n\r\nWith the yoga balls and the lap-desks, I will be able
to increase the options for seating in my classroom and expand its i
maginable space.nannan
--------------------------------------------------
143 My students are amazing in every single way. We are an incredibl
y diverse group and every single person has their own background and
story. Although we may be from different cultures, states, or even c
ountries, we are one big family. \r\n\r\nMy students pride themselve
s on the fact that no two people in our class are the same. Each and
every one of us learns in their own unique way and they have come to
embrace their differences and use them to their advantage. \r\n\r\nW
e are not your typical class and we are extremely grateful for that.
These students are a special group of individuals who yearn to learn
```

```
and always strive to better than the day before. \r\n\r\nA MacBook P
ro will serve as additional means of technology within our classroo
m. We use different form of technology every single day in both Math
and Science. This laptop will be used in both small group and indivi
dual settings that allow the students to research materials in Scien
ce. \r\n\r\nStudents will use the laptop as an interactive tool in M
ath and Science. Games, interactive PowerPoints, and so much more wi
ll literally be at their fingertips once we receive the MacBook Pro
laptop. This piece of technology will allow my students to continue
to broaden their  knowledge of any and all subjects.nannan
```

In [189]:

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

```
100%|██████████| 50000/50000 [04:38<00:00, 179.61it/s]
```

In [190]:

```
print("printing some random essay")
print(9, preprocessed_essays[9])
print('-'*50)
print(34, preprocessed_essays[34])
print('-'*50)
print(143, preprocessed_essays[143])
```

```
printing some random essay
9 95 students free reduced lunch homeless despite come school eagern
ess learn students inquisitive eager learners embrace challenge not
great books resources every day many not afforded opportunity engage
big colorful pages book regular basis home not travel public library
duty teacher provide student opportunity succeed every aspect life r
eading fundamental students read books boosting comprehension skills
books used read alouds partner reading independent reading engage re
ading build love reading reading pure enjoyment introduced new autho
rs well old favorites want students ready 21st century know pleasure
holding good hard back book hand nothing like good book read student
s soar reading consideration generous funding contribution help buil
d 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 program receive free reduced meals breakfast lunch want st
udents feel comfortable classroom home many students take multiple r
oles home well school sometimes caretakers younger siblings cooks ba
bysitters academics friends developing going become adults consider
essential part job model helping others gain knowledge positive mann
er result community students love helping outside classroom consiste
ntly look opportunities support learning kind helpful way excited ex
perimenting alternative seating classroom school year studies shown
giving students option sit classroom increases focus well motivation
allowing students choice classroom able explore create welcoming env
ironment alternative classroom seating experimented frequently recen
t years believe along many others every child learns differently not
apply multiplication memorized paper written applies space asked wor
k students past ask work library work carpet answer always long lear
ning work wherever want yoga balls lap desks able increase options s
eating classroom expand imaginable space nannan
--------------------------------------------------
143 students amazing every single way incredibly diverse group every
single person background story although may different cultures state
s even countries one big family students pride fact no two people cl
ass every one us learns unique way come embrace differences use adva
ntage not typical class extremely grateful students special group in
dividuals yearn learn always strive better day macbook pro serve add
itional means technology within classroom use different form technol
ogy every single day math science laptop used small group individual
settings allow students research materials science students use lapt
op interactive tool math science games interactive powerpoints much
literally fingertips receive macbook pro laptop piece technology all
ow students continue broaden knowledge subjects nannan
```

3.8. Preprocessing Numerical features : price

In [203]:

```
# 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[203]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [204]:

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

In [205]:

```
processed_data['price'].head()
```

Out[205]:

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

In [206]:

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

In [207]:

```
processed_data['std_price'].head()
```

Out[207]:

```
0   -0.382681
1   -0.000882
2    0.575122
3   -0.175653
4   -0.611708
Name: std_price, dtype: float64
```

In [208]:

```
# applying MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(processed_data['price'].values.reshape(-1, 1))
processed_data['nrm_price']=scaler.transform(processed_data['price'].values.resh
ape(-1, 1))
```

In [248]:

```
processed_data['nrm_price'].head()
```

Out[248]:

```
0    0.015397
1    0.029839
2    0.051628
3    0.023228
4    0.006733
Name: nrm_price, dtype: float64
```

4. Vectorization

In [213]:

```
data  = pd.read_csv('preprocessed_data.csv', nrows=50000)
data.head(2)
```

Out[213]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_p
0	ca	mrs	grades_prek_2	
1	ut	ms	grades_3_5	

In [250]:

```
data.columns
```

Out[250]:

```
Index(['school_state', 'teacher_prefix', 'project_grade_category',
       'teacher_number_of_previously_posted_projects', 'project_is_a
pproved',
       'clean_categories', 'clean_subcategories', 'essay', 'price'],
      dtype='object')
```

we are going to consider

1) text data

- Essay
- project_title

2) categorical data

- school_state
- subject_categories
- subject_subcategories
- project_grade_category
- teacher_prefix

3) numerical data

- price
- teacher_number_of_previously_posted_projects

4.1 Vectorizing Text data : Essay

4.1.1 Bag of words

In [214]:

```
preprocessed_essays = data['essay'].values
```

In [215]:

```
# We are considering only the words which appeared in at least 10 documents(rows
or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (50000, 12122)
```

4.1.2 TFIDF vectorizer

In [217]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (50000, 12122)
```

4.1.3 Using Pretrained Models: Avg W2V

In [224]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupu
s", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-t
o-use-pickle-to-save-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)


'''
```

Out[224]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38
230349/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading
Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    m
odel = {}\n    for line in tqdm(f):\n        splitLine = line.split
()\n        word = splitLine[0]\n        embedding = np.array([float
(val) for val in splitLine[1:]])\n        model[word] = embedding\n
print ("Done.",len(model)," words loaded!")\n    return model\nmodel
= loadGloveModel(\'glove.42B.300d.txt\')\n\n# ======================
======\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.6
9it/s]\nDone. 1917495  words loaded!\n\n# ========================
==\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.spli
t(\' \'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\'
\'))\nprint("all the words in the coupus", len(words))\nwords = set
(words)\nprint("the unique words in the coupus", len(words))\n\ninte
r_words = set(model.keys()).intersection(words)\nprint("The number o
f words that are present in both glove vectors and our coupus",
len(inter_words),"(",np.round(len(inter_words)/len(words)*100,
3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor
i in words:\n    if i in words_glove:\n        words_courpus[i] = mo
del[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n# strong
ing variables into pickle files python: http://www.jessicayung.com/h
ow-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(wor
ds_courpus, f)\n\n\n'
```

In [225]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-t
o-use-pickle-to-save-and-load-variables-in-python/
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [226]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|██████████| 50000/50000 [02:46<00:00, 301.10it/s]

50000
300
```

4.1.4 Using Pretrained Models: TFIDF weighted W2V

In [227]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [228]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf val
ue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|██████████| 50000/50000 [20:40<00:00, 40.32it/s]

50000
300
```

4.2 Vectorizing Text data : Project_title

4.2.1 Bag of words

In [219]:

```
# We are considering only the words which appeared in at least 5 documents(rows
 or projects).
vectorizer = CountVectorizer(min_df= 5)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encoding ",title_bow.shape)
```

```
Shape of matrix after one hot encoding  (50000, 3230)
```

4.2.2 TFIDF vectorizer

In [220]:

```
vectorizer = TfidfVectorizer(min_df= 5)
title_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encoding ",title_tfidf.shape)
```

```
Shape of matrix after one hot encoding  (50000, 3230)
```

4.3 Vectorizing Categorical Features : school_state, teacher_prefix, clean_categories, project_grade_category

In [245]:

```
# provided we did the cleaning
vectorizer = CountVectorizer(binary=True)
school_state_ohe = vectorizer.fit_transform(data['school_state'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",school_state_ohe.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga',
'hi', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'm
i', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'n
v', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'u
t', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
Shape of matrix after one hot encodig  (50000, 51)
```

In [246]:

```
# provided we did the cleaning
vectorizer = CountVectorizer(binary=True)
teacher_prefix_ohe = vectorizer.fit_transform(data['teacher_prefix'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",teacher_prefix_ohe.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encodig  (50000, 5)
```

In [243]:

```
# provided we did the cleaning
vectorizer = CountVectorizer(binary=True)
clean_categories_ohe = vectorizer.fit_transform(data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",clean_categories_ohe.shape)
```

```
['appliedlearning', 'care_hunger', 'health_sports', 'history_civic
s', 'literacy_language', 'math_science', 'music_arts', 'specialneed
s', 'warmth']
Shape of matrix after one hot encodig  (50000, 9)
```

In [251]:

```
# provided we did the cleaning
vectorizer = CountVectorizer(binary=True)
clean_subcategories_ohe = vectorizer.fit_transform(data['clean_subcategories'].v
alues)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",clean_subcategories_ohe.shape)
```

```
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_gov
ernment', 'college_careerprep', 'communityservice', 'earlydevelopmen
t', 'economics', 'environmentalscience', 'esl', 'extracurricular',
'financialliteracy', 'foreignlanguages', 'gym_fitness', 'health_life
science', 'health_wellness', 'history_geography', 'literacy', 'liter
ature_writing', 'mathematics', 'music', 'nutritioneducation', 'othe
r', 'parentinvolvement', 'performingarts', 'socialsciences', 'specia
lneeds', 'teamsports', 'visualarts', 'warmth']
Shape of matrix after one hot encodig  (50000, 30)
```

In [247]:

```
# provided we did the cleaning
vectorizer = CountVectorizer(binary=True)
project_grade_category_ohe = vectorizer.fit_transform(data['project_grade_catego
ry'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",project_grade_category_ohe.shape)
```

```
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix after one hot encodig  (50000, 4)
```

4.2 Vectorizing Numerical data : Price, teacher_number_of_previously_posted_projects

In [256]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
price_ohe = normalizer.fit_transform(data['price'].values.reshape(-1,1))
print("Shape of matrix after one hot encodig ",price_ohe.shape)
```

```
Shape of matrix after one hot encodig  (50000, 1)
```

In [257]:

```
normalizer = Normalizer()
teacher_no_of_pre_projects_ohe = normalizer.fit_transform(data['teacher_number_o
f_previously_posted_projects'].values.reshape(-1,1))
print("Shape of matrix after one hot encodig ",teacher_no_of_pre_projects_ohe.sh
ape)
```

```
Shape of matrix after one hot encodig  (50000, 1)
```

5. Merging all the features

In [258]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense
matirx :)
X = hstack((school_state_ohe,teacher_prefix_ohe,clean_categories_ohe,clean_subca
tegories_ohe,
            project_grade_category_ohe,price_ohe,teacher_no_of_pre_projects_ohe,
title_bow, text_bow,
            ))
X.shape
```

Out[258]:

```
(50000, 15453)
```