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	D
`project_id`	A unique identifier for the proposed project. **Example:**
`project_title`	Title of the project. **I** • `Art Will Make You` • `First Gr`
`project_grade_category`	Grade level of students for which the project is targeted. One of th enumerat • `Grades` • `Gra` • `Gra` • `Grad`
`project_subject_categories`	One or more (comma-separated) subject categories for the proje following enumerated list • `Applied I` • `Care &` • `Health &` • `History &` • `Literacy & I` • `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#Post](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Post) **Exa**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for t **E** • `I` • `Literature & Writing, Social S`
`project_resource_summary`	An explanation of the resources needed for the project. • `My students need hands on literacy materials to sensory`
`project_essay_1`	First applica
`project_essay_2`	Second applica
`project_essay_3`	Third applica
`project_essay_4`	Fourth applica

Feature	D
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
teacher_prefix	Teacher's title. One of the following enumerat • • • • • • T
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the san **Ex**

* 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_4:__ "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 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
```

1.1 Reading Data

In [2]:

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

In [3]:

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefi
x' '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 [4]:

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

# Taking 15k points from project data
project_data = project_data.head(15000)
print(project_data.shape)
```

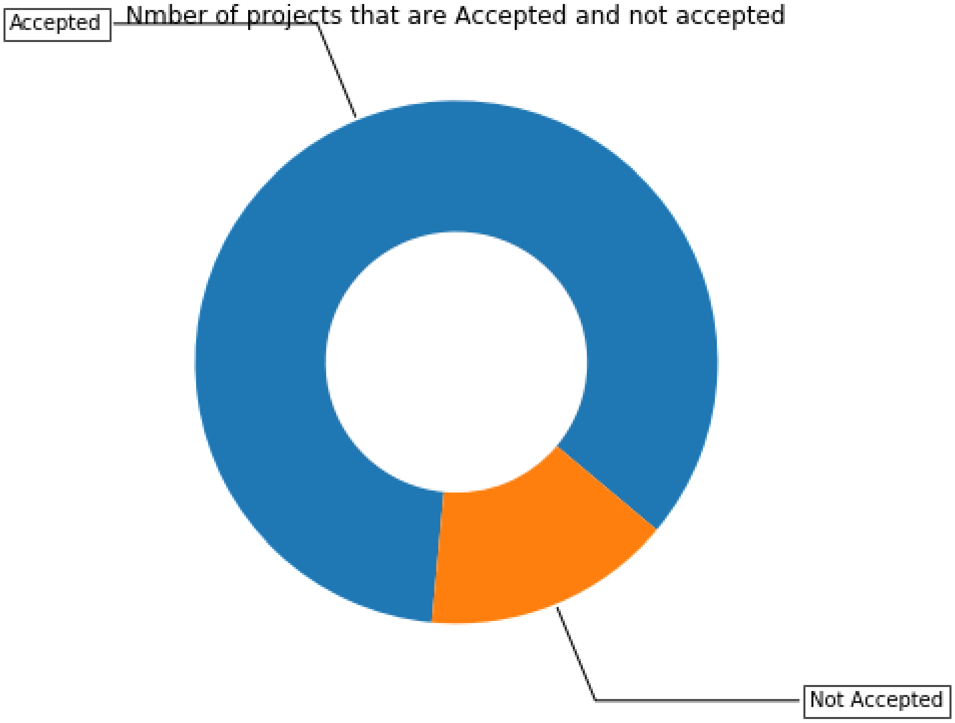
```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
(15000, 17)
```

1.2 Data Analysis

In [5]:

```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labe
ls.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_count
s[1], ", (", (y_value_counts[1]/(y_value_counts[1]+y_value_counts[0]))*1
00,"%)")
print("Number of projects thar are not approved for funding ", y_value_c
ounts[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  12693 , ( 84.61999999
999999 %)
Number of projects thar are not approved for funding  2307 , ( 15.37999
9999999999 %)
```



1.2.1 Univariate Analysis: School State

In [6]:

```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/1938
5591/4084039

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

'''# How to plot US state heatmap: https://datascience.stackexchange.co
m/a/9620

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(18
8,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'r
gb(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')
'''
```

Out[6]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.com/
a/9620\n\nscl = [[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)\']]\n\ndata = [
dict(\n        type=\'choropleth\',\n        colorscale = scl,\n
autocolorscale = False,\n        locations = temp[\'state_code\'],\n
z = temp[\'num_proposals\'].astype(float),\n        locationmode = \'US
A-states\',\n        text = temp[\'state_code\'],\n        marker = dic
t(line = dict (color = \'rgb(255,255,255)\',width = 2)),\n        color
bar = dict(title = "% of pro")\n    ) ]\n\nlayout = dict(\n        titl
e = \'Project Proposals % of Acceptance Rate by US States\',\n        g
eo = dict(\n            scope=\'usa\',\n            projection=dict( ty
pe=\'albers usa\' ),\n            showlakes = True,\n            lakeco
lor = \'rgb(255, 255, 255)\',\n        ),\n    )\n\nfig = go.Figure(dat
a=data, layout=layout)\noffline.iplot(fig, filename=\'us-map-heat-map
\')\n'
```

In [7]:

```python
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2l
etterstabbrev.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
50         WY       0.666667
26         MT       0.677419
7          DC       0.737500
41         SD       0.772727
46         VT       0.777778
==================================================
States with highest % approvals
   state_code  num_proposals
17         KY       0.888889
32         NM       0.909091
16         KS       0.919192
8          DE       0.925000
28         ND       1.000000
```

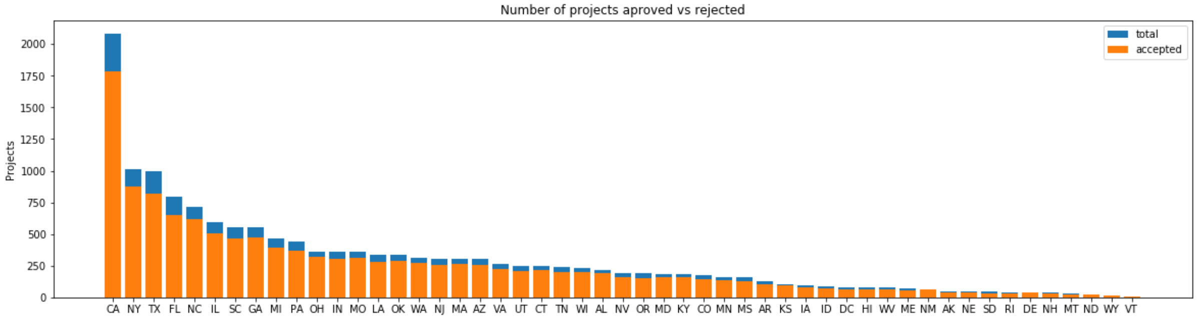
In [8]:

```
#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('Number of projects aproved vs rejected')
    plt.xticks(ind, list(data[xtick].values))
    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()
```

In [9]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False,sortby='total'):
    # 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=[sortby],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 [10]: univariate_barplots(project_data, 'school_state', 'project_is_approved',
         False)
```

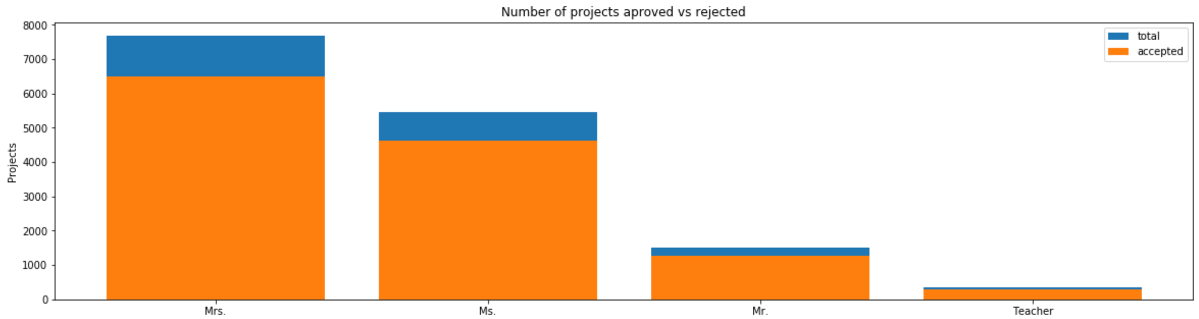


```
   school_state  project_is_approved  total       Avg
4            CA                 1781   2083  0.855017
34           NY                  877   1014  0.864892
43           TX                  817    999  0.817818
9            FL                  653    799  0.817272
27           NC                  616    716  0.860335
==================================================
   school_state  project_is_approved  total       Avg
30           NH                   33     38  0.868421
26           MT                   21     31  0.677419
28           ND                   21     21  1.000000
50           WY                   12     18  0.666667
46           VT                    7      9  0.777778
```

SUMMARY: Every state has greater than 80% success rate in approval

1.2.2 Univariate Analysis: teacher_prefix

```
In [11]: univariate_barplots(project_data, 'teacher_prefix', 'project_is_approve
d' , top=False)
```



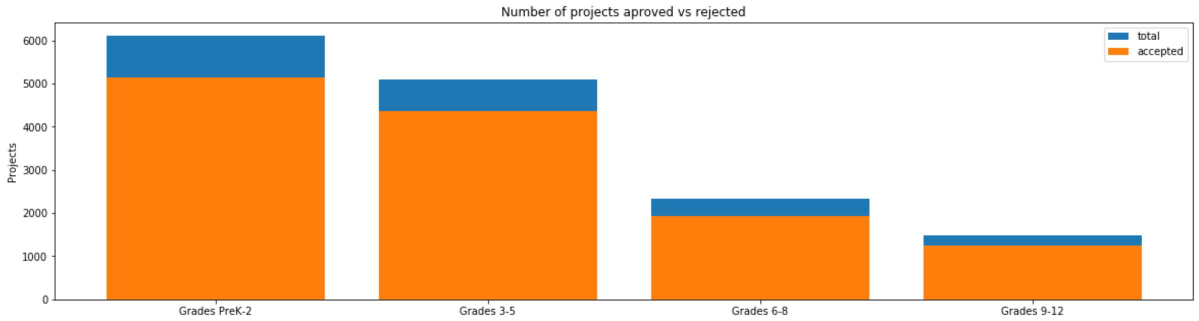
```
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 6506   7687  0.846364
2            Ms.                 4634   5462  0.848407
0            Mr.                 1274   1496  0.851604
3        Teacher                  278    354  0.785311
==================================================
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 6506   7687  0.846364
2            Ms.                 4634   5462  0.848407
0            Mr.                 1274   1496  0.851604
3        Teacher                  278    354  0.785311
```

Observation(s):

1. Approval rate is low for Teacher. and More for Mrs.

1.2.3 Univariate Analysis: project_grade_category

```
In [12]: univariate_barplots(project_data, 'project_grade_category', 'project_is_
         approved', top=False)
```



```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 5146   6112  0.841950
0             Grades 3-5                 4370   5086  0.859221
1             Grades 6-8                 1940   2323  0.835127
2            Grades 9-12                 1237   1479  0.836376
==================================================
```

```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 5146   6112  0.841950
0             Grades 3-5                 4370   5086  0.859221
1             Grades 6-8                 1940   2323  0.835127
2            Grades 9-12                 1237   1479  0.836376
```

Observation(s):

1. Approvale rate is more for PreK-2
2. For all grades approval rate is more than 83%

1.2.4 Univariate Analysis: project_subject_categories

In [13]:

```
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())
```

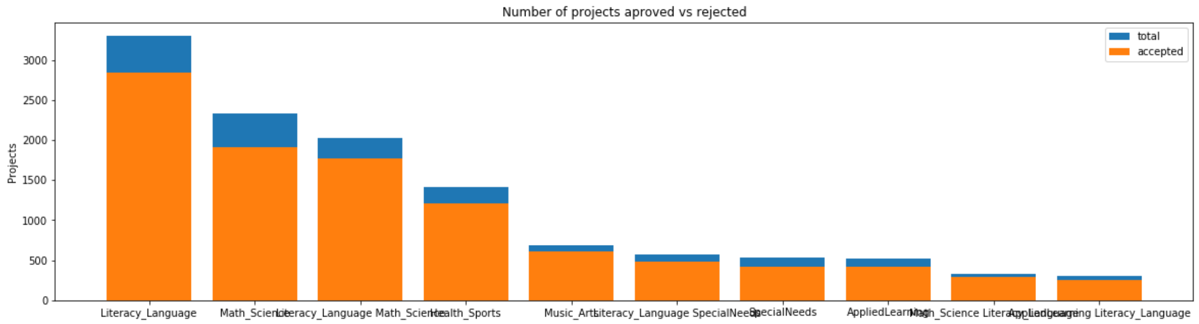
In [14]:

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

Out[14]:

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

In [15]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approv
ed', top=10)
```

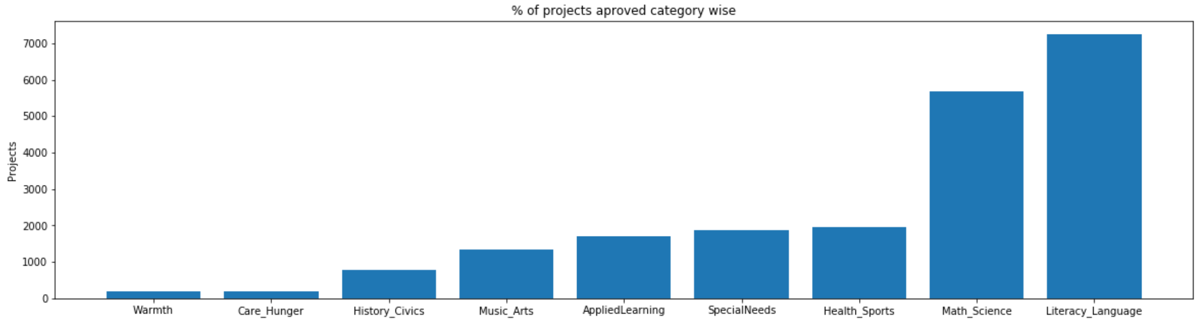


```
                  clean_categories  project_is_approved  total       Av
g
23               Literacy_Language                 2844   3301  0.86155
7
31                    Math_Science                 1915   2329  0.82224
1
27  Literacy_Language Math_Science                 1769   2031  0.87100
0
8                    Health_Sports                 1205   1413  0.85279
5
39                      Music_Arts                  606    693  0.87445
9
==================================================
                     clean_categories  project_is_approved  total
Avg
29     Literacy_Language SpecialNeeds                  484    571  0.84
7636
45                       SpecialNeeds                  423    534  0.79
2135
0                     AppliedLearning                  420    525  0.80
0000
35     Math_Science Literacy_Language                  292    332  0.87
9518
3   AppliedLearning Literacy_Language                  257    304  0.84
5395
```

In [16]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2
2898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

In [17]:

```
# 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 category wise')
plt.xticks(ind, list(sorted_cat_dict.keys()))
plt.show()
```



In [18]:

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

```
Warmth               :       190
Care_Hunger          :       190
History_Civics       :       779
Music_Arts           :      1355
AppliedLearning      :      1711
SpecialNeeds         :      1860
Health_Sports        :      1953
Math_Science         :      5695
Literacy_Language    :      7249
```

Observation(s)

```
1. Literacy_Language and Math_Science has more Approval Rate
2. warmth category has less approval rate
```

1.2.5 Univariate Analysis: project_subject_subcategories ¶

In [19]:

```
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 [20]:

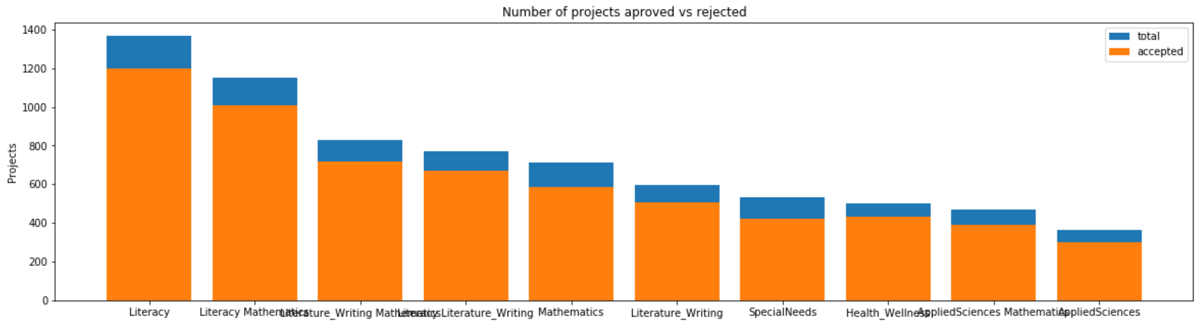
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[20]:

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

In [21]:

```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_app
roved', top=10)
```

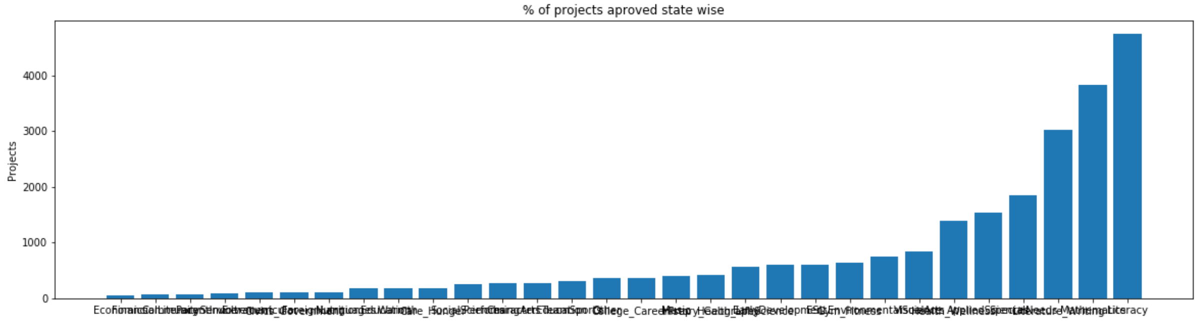


```
                clean_subcategories  project_is_approved  total        A
vg
252                        Literacy                 1197   1367   0.8756
40
254            Literacy Mathematics                 1009   1152   0.8758
68
265  Literature_Writing Mathematics                  718    828   0.8671
50
253     Literacy Literature_Writing                  668    773   0.8641
66
275                     Mathematics                  585    714   0.8193
28
==================================================
             clean_subcategories  project_is_approved  total       Avg
264           Literature_Writing                  507    594  0.853535
315                 SpecialNeeds                  423    534  0.792135
228              Health_Wellness                  433    503  0.860835
17   AppliedSciences Mathematics                  390    467  0.835118
0                AppliedSciences                  303    364  0.832418
```

In [22]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/2
2898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [23]:
```
# 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 [24]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :        47
FinancialLiteracy    :        71
CommunityService     :        77
ParentInvolvement    :        98
Extracurricular      :        99
Civics_Government    :       104
ForeignLanguages     :       106
NutritionEducation   :       179
Warmth               :       190
Care_Hunger          :       190
SocialSciences       :       250
PerformingArts       :       267
CharacterEducation   :       280
TeamSports           :       305
Other                :       364
College_CareerPrep   :       368
Music                :       400
History_Geography    :       414
Health_LifeScience   :       563
EarlyDevelopment     :       599
ESL                  :       602
Gym_Fitness          :       640
EnvironmentalScience :       744
VisualArts           :       834
Health_Wellness      :      1399
AppliedSciences      :      1535
SpecialNeeds         :      1860
Literature_Writing   :      3026
Mathematics          :      3835
Literacy             :      4749
```

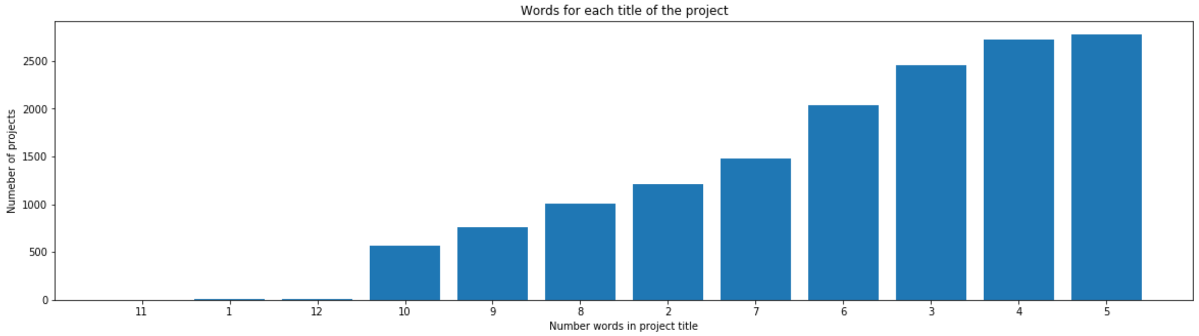
Observation(s):

1. Literacy and Mathematics subcategories has more approval rate

1.2.6 Univariate Analysis: Text features (Title)

In [25]:

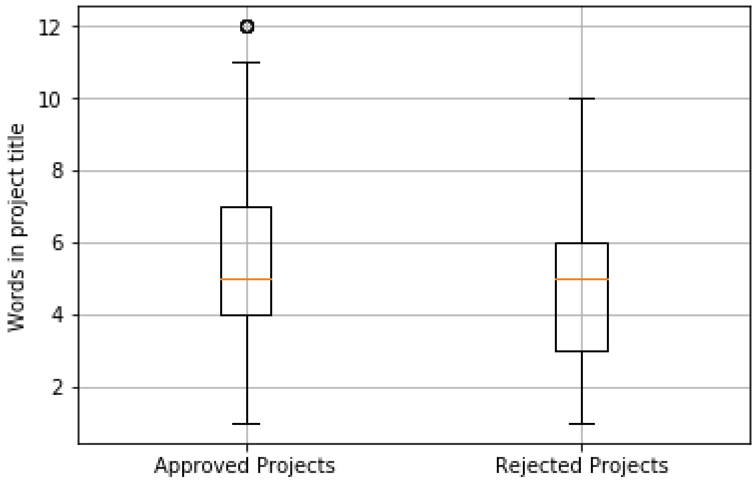
```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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



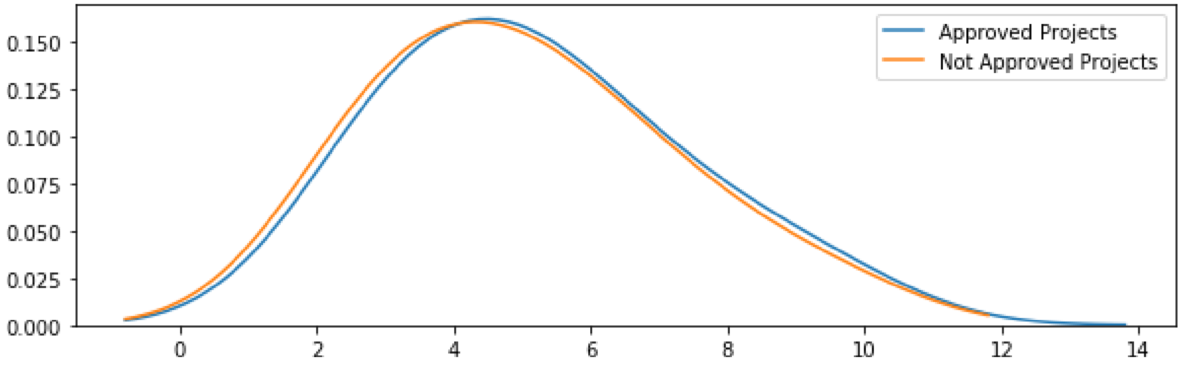
In [26]:

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

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

In [27]:
```python
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



In [28]:
```python
plt.figure(figsize=(10,3))
sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)
sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6)
plt.legend()
plt.show()
```



Observation(s)

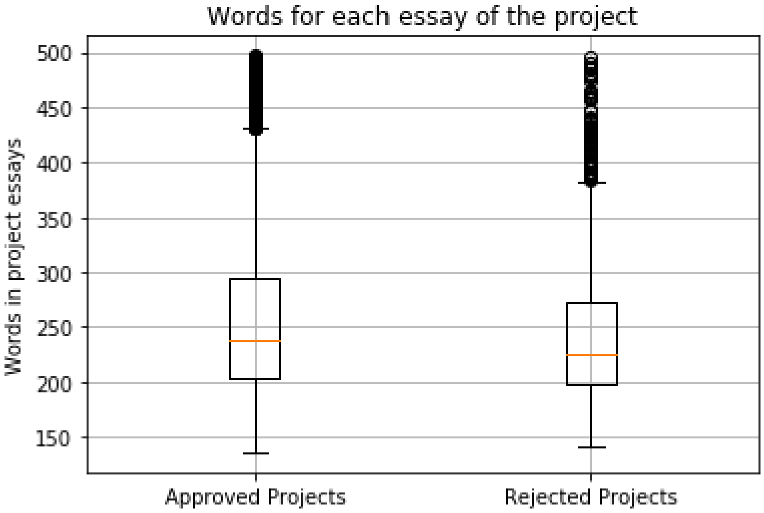
```
    1. The median's for Approved project and Rejected projects is slightly same
    2. If the Title contains more numbers of words it has more chances of getting approved.
```

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

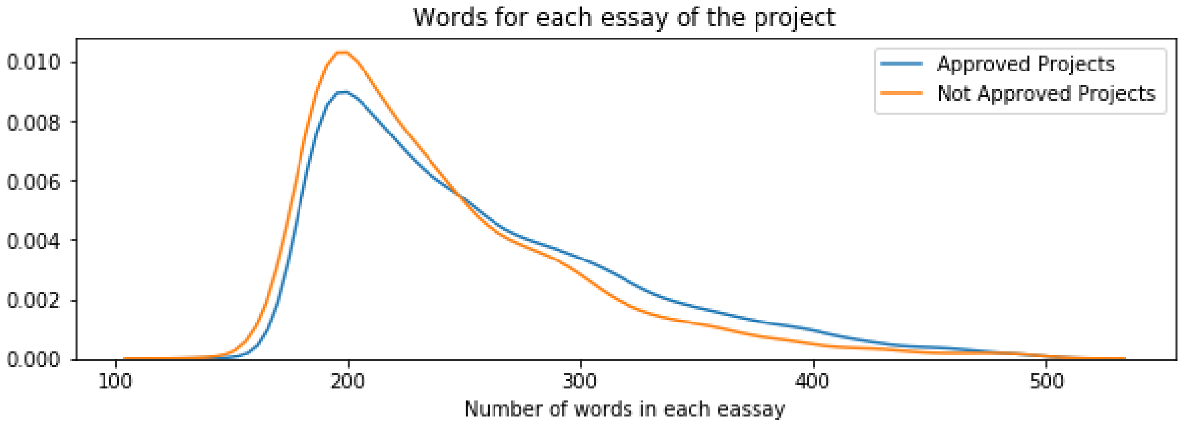
In [29]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [30]:
```
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 [31]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.htm
l
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 essays')
plt.grid()
plt.show()
```



In [32]:
```
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()
```



Observations(s)

```
1. Approved projects has more number of words in the essay's
```

1.2.8 Univariate Analysis: Cost per project

In [33]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[33]:

	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 [34]:
```
# 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[34]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [35]:

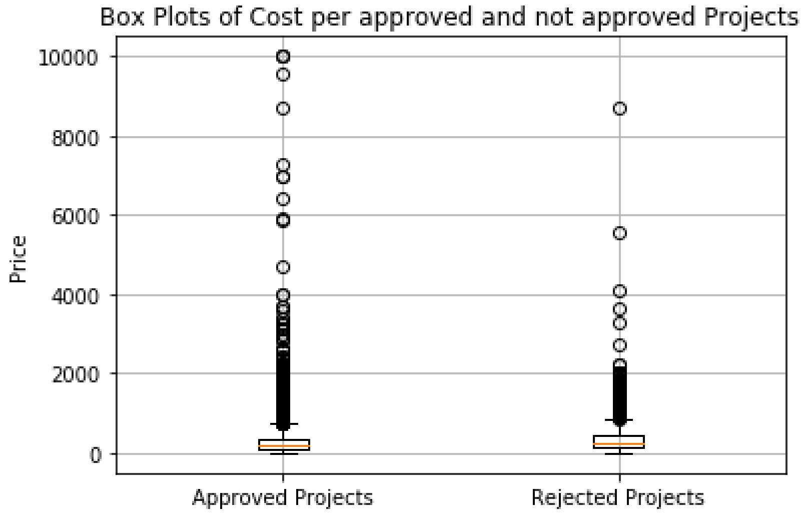
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [36]:

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

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

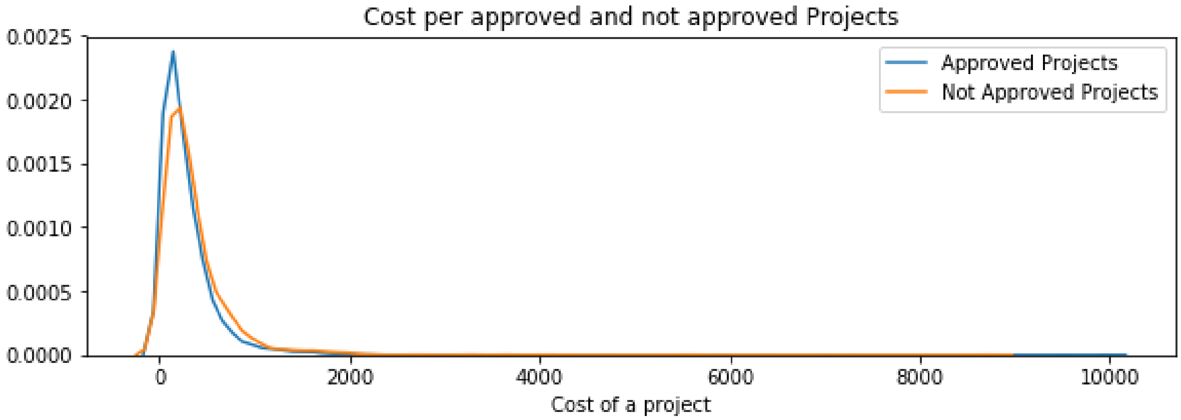
In [37]:

```
# 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('Price')
plt.grid()
plt.show()
```



In [38]:

```
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 [39]:

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

#If you get a ModuleNotFoundError error , install prettytable using: pip
3 install prettytable

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

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

```
+------------+-------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+-------------------+-----------------------+
|     0      |        1.44       |          4.96         |
|     5      |       13.912      |         40.303        |
|     10     |       33.99       |         74.848        |
|     15     |       57.876      |         99.957        |
|     20     |       78.14       |        119.982        |
|     25     |       99.99       |        140.965        |
|     30     |      116.882      |        158.952        |
|     35     |      136.702      |        183.054        |
|     40     |       157.0       |        209.934        |
|     45     |       176.4       |        233.907        |
|     50     |       197.87      |         256.39        |
|     55     |      222.972      |        283.239        |
|     60     |       254.0       |         313.21        |
|     65     |      284.432      |        352.169        |
|     70     |      320.434      |        392.992        |
|     75     |       367.77      |        435.285        |
|     80     |      412.992      |        497.706        |
|     85     |      479.016      |        606.088        |
|     90     |      592.782      |        729.922        |
|     95     |      801.056      |         967.0         |
|    100     |       9999.0      |        8719.69        |
+------------+-------------------+-----------------------+
```

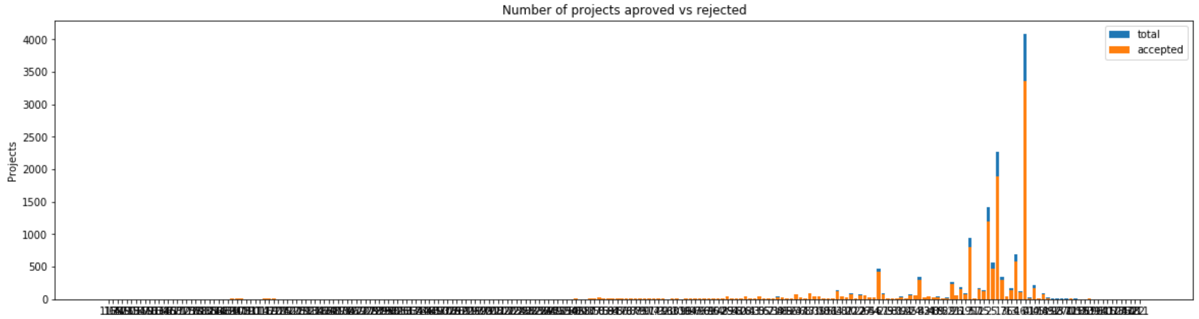
Observation(s)

```
    1. Approved projects tend to have less cost compared to Not approved projects
```

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

Please do this on your own based on the data analysis that was done in the above cells

In [40]:
```
# Taking all the datapoints and sorting based on the Average
# Added a parameter in univariate_barplots method to sort based on Avera
ge
univariate_barplots(project_data, 'teacher_number_of_previously_posted_p
rojects', 'project_is_approved', top=False,sortby='Avg')
```

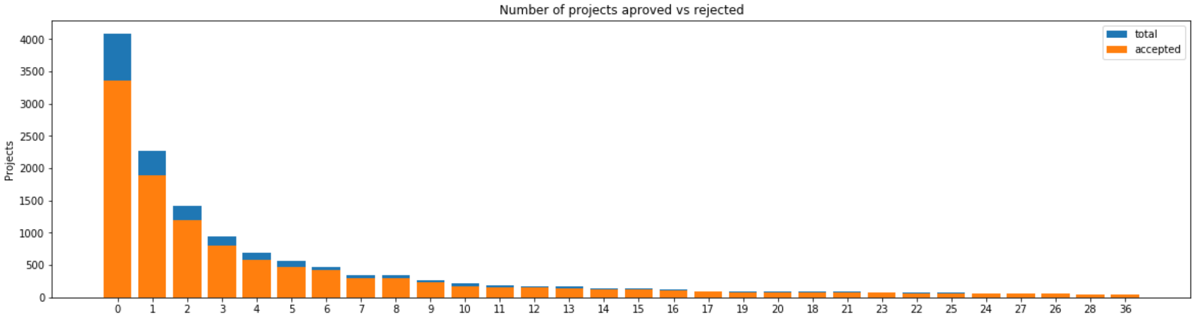


```
     teacher_number_of_previously_posted_projects  project_is_approved  
total  \
113                                           113                    4  
4   
155                                           165                    2  
2   
143                                           147                    1  
1   
144                                           149                    2  
2   
145                                           150                    3  
3   

     Avg  
113  1.0  
155  1.0  
143  1.0  
144  1.0  
145  1.0  
==================================================
     teacher_number_of_previously_posted_projects  project_is_approved  
total  \
132                                           135                    1  
2   
137                                           140                    1  
2   
163                                           178                    1  
2   
215                                           322                    0  
1   
181                                           211                    0  
1   

     Avg  
132  0.5  
137  0.5  
163  0.5  
215  0.0  
181  0.0  
```

Observation(s)

```
1. The success rate lies between 50% to 100% except for one datapoint
2. Number of previuosly posted projects doesn't impact the successrate
```

In [41]:
```
# Taking top 30 datapoints and sorting based on the Total number of proj
ects submitted
univariate_barplots(project_data, 'teacher_number_of_previously_posted_p
rojects', 'project_is_approved', top=30)
```

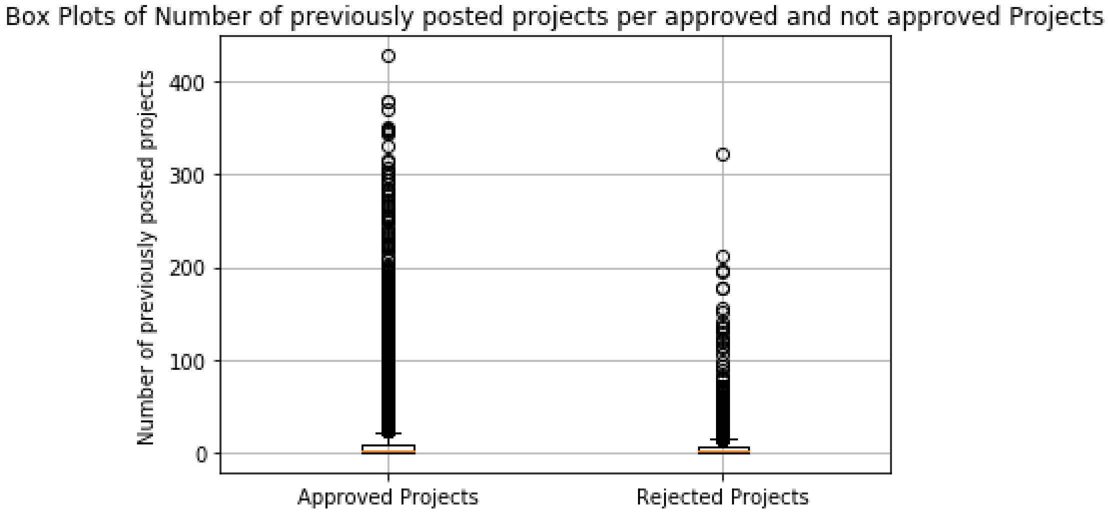


```
   teacher_number_of_previously_posted_projects  project_is_approved  t
otal  \
0                                             0                 3361
4086
1                                             1                 1890
2265
2                                             2                 1194
1425
3                                             3                  799
939
4                                             4                  574
695

        Avg
0  0.822565
1  0.834437
2  0.837895
3  0.850905
4  0.825899
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved
total  \
24                                            24                   55
63
27                                            27                   55
62
26                                            26                   53
62
28                                            28                   47
50
36                                            36                   40
48

         Avg
24  0.873016
27  0.887097
26  0.854839
28  0.940000
36  0.833333
```

Observations

```
    1. Most of the projects are submitted by new teachers and has success ra
te of 82%
```

In [42]:
```
approved_prvsly_posted = project_data[project_data['project_is_approved'
]==1]['teacher_number_of_previously_posted_projects'].values

rejected_prvsly_posted = project_data[project_data['project_is_approved'
]==0]['teacher_number_of_previously_posted_projects'].values
plt.boxplot([approved_prvsly_posted, rejected_prvsly_posted])
plt.title('Box Plots of Number of previously posted projects per approve
d and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Number of previously posted projects')
plt.grid()
plt.show()
```



Obsevation(s)

```
    1. Most of the approved and non approved projects are submitted by teach
ers who has previously sumbitted less than or equal to 5 projects
```

1.2.10 Univariate Analysis: project_resource_summary

Please do this on your own based on the data analysis that was done in the above cells

Check if the `presence of the numerical digits` in the `project_resource_summary` effects the acceptance of the project or not. If you observe that `presence of the numerical digits` is helpful in the classification, please include it for further process or you can ignore it.

1.2.10.1 Univariate Analysis: Based on presence of the numerical digits

In [43]:

```
# convert string one to int 1 in python: https://stackoverflow.com/questions/493174/is-there-a-way-to-convert-number-words-to-integers
def is_number(x):
    if type(x) == str:
        x = x.replace(',', '')
    try:
        float(x)
    except:
        return False
    return True

def text2int (textnum, numwords={}):
    units = [
        'zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight',
        'nine', 'ten', 'eleven', 'twelve', 'thirteen', 'fourteen', 'fifteen',
        'sixteen', 'seventeen', 'eighteen', 'nineteen',
    ]
    tens = ['', '', 'twenty', 'thirty', 'forty', 'fifty', 'sixty', 'seventy', 'eighty', 'ninety']
    scales = ['hundred', 'thousand', 'million', 'billion', 'trillion']
    ordinal_words = {'first':1, 'second':2, 'third':3, 'fifth':5, 'eighth':8, 'ninth':9, 'twelfth':12}
    ordinal_endings = [('ieth', 'y'), ('th', '')]

    if not numwords:
        numwords['and'] = (1, 0)
        for idx, word in enumerate(units): numwords[word] = (1, idx)
        for idx, word in enumerate(tens): numwords[word] = (1, idx * 10)
        for idx, word in enumerate(scales): numwords[word] = (10 ** (idx * 3 or 2), 0)

    textnum = textnum.replace('-', ' ')

    current = result = 0
    curstring = ''
    onnumber = False
    lastunit = False
    lastscale = False

    def is_numword(x):
        if is_number(x):
            return True
        if word in numwords:
            return True
        return False

    def from_numword(x):
        if is_number(x):
            scale = 0
            x = x.replace('.', '')
            increment = int(x.replace(',', ''))
            return scale, increment
        return numwords[x]
    for word in textnum.split():
        if word in ordinal_words:
            scale, increment = (1, ordinal_words[word])
            current = current * scale + increment
            if scale > 100:
                result += current
                current = 0
            onnumber = True
            lastunit = False
            lastscale = False
        else:
            for ending, replacement in ordinal_endings:
                if word.endswith(ending):
                    word = "%s%s" % (word[:-len(ending)], replacement)

            if (not is_numword(word)) or (word == 'and' and not lastscal
e) or (word == 'Infinity'):
                if onnumber:
                    # Flush the current number we are building
                    curstring += repr(result + current) + " "
                curstring += word + " "
                result = current = 0
                onnumber = False
                lastunit = False
                lastscale = False
            else:
                scale, increment = from_numword(word)
                onnumber = True

                if lastunit and (word not in scales):
                    # Assume this is part of a string of individual numb
ers to
                    # be flushed, such as a zipcode "one two three four
 five"
                    curstring += repr(result + current)
                    result = current = 0

                if scale > 1:
                    current = max(1, current)

                current = current * scale + increment
                if scale > 100:
                    result += current
                    current = 0

                lastscale = False
                lastunit = False
                if word in scales:
                    lastscale = True
                elif word in units:
                    lastunit = True

    if onnumber:
        curstring += repr(result + current)

    return curstring
```

In [44]:

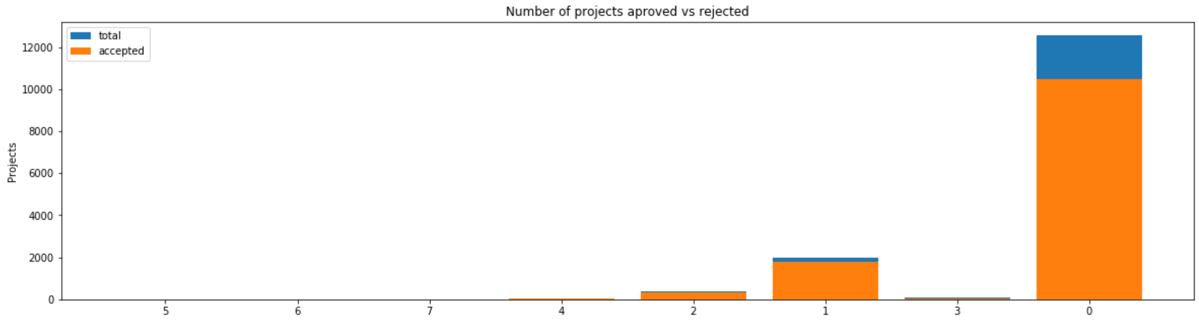
```
# Getting the count of digits present in project_resource_summary

# Empty list to get the digits count
digits_count_list = []

# Iterating the list for every summary present in the project_data
for summary in project_data ['project_resource_summary'].values:

    # setting the digit cout to zero for every new summary
    digit_count = 0

    # Convert the Text represenation of words on the summary to Int repr
esnetaion
    summary = text2int (summary)

    # Iterating for every word present in the summary
    for word in list (summary.split()):

        # if the word is a digit
        if word.isdigit():

            # Increase the counter
            digit_count +=1

    # After the completing all the words in summary append the count to
 digits_count_list
    digits_count_list.append(digit_count)

# Add the Digits count list to the project_data
project_data['count_of_digits_in_summary'] = digits_count_list


# test
project_data.head(5)
```

Out[44]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
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	

5 rows × 21 columns

In [45]:
```
# Taking all datapoints and sorting based on the success rate
univariate_barplots(project_data, 'count_of_digits_in_summary', 'project
_is_approved', top=False, sortby='Avg')
```



```
   count_of_digits_in_summary  project_is_approved  total       Avg
5                           5                    7      7  1.000000
6                           6                    2      2  1.000000
7                           7                    1      1  1.000000
4                           4                   19     20  0.950000
2                           2                  336    367  0.915531
==================================================
   count_of_digits_in_summary  project_is_approved  total       Avg
4                           4                   19     20  0.950000
2                           2                  336    367  0.915531
1                           1                 1767   1965  0.899237
3                           3                   63     71  0.887324
0                           0                10498  12567  0.835362
```

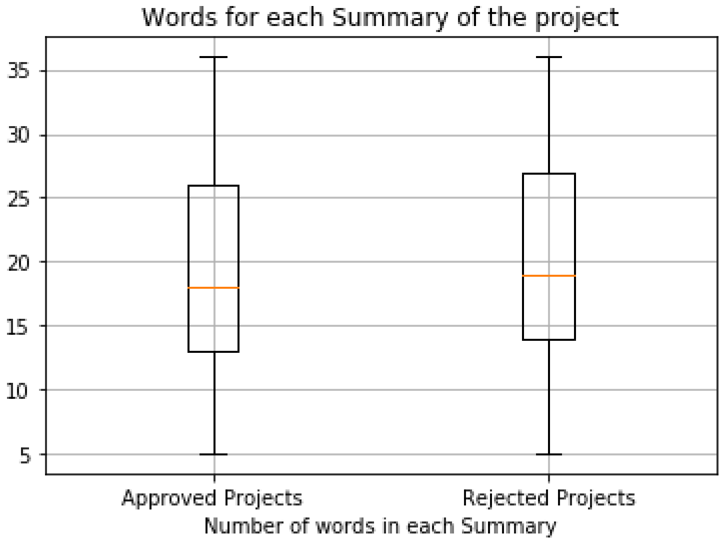
Obesevation(s):

```
    1. For the projects which has more numbers in the project summary has Mo
re approval rate
    2. The projects which doesn't have numbers in the project summary has 8
3% approval rate (very less compared to others)
    3. Most of the project summaries has one or two numbers in the project s
ummary and has 89% approval rate
```

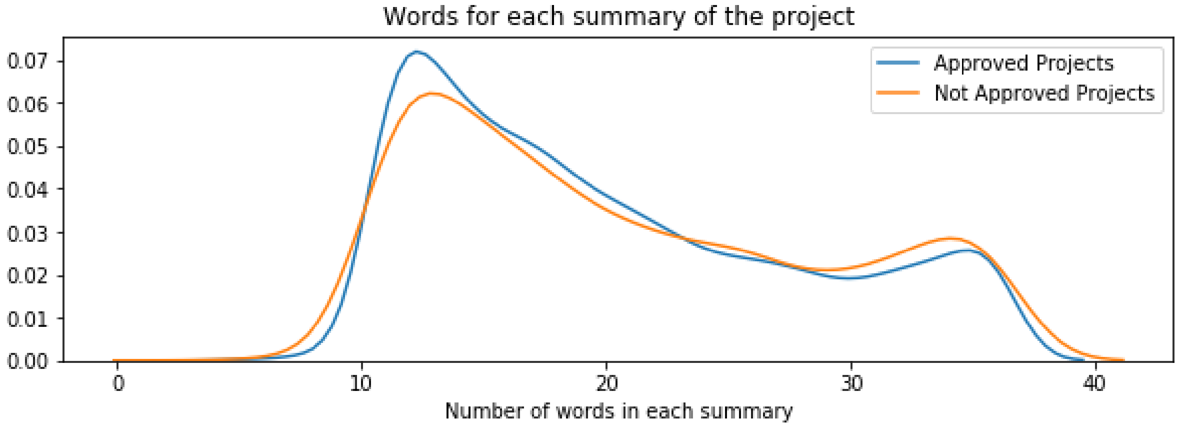
1.2.10.2 Univariate Analysis: Based on number of words present in summary

In [46]:
```
approved_word_count = project_data[project_data['project_is_approved']==
1]['project_resource_summary'].str.split().apply(len)
approved_word_count = approved_word_count.values

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

In [47]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.htm
l
plt.boxplot([approved_word_count, rejected_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.title('Words for each Summary of the project')
plt.xlabel('Number of words in each Summary')
plt.grid()
plt.show()
```



In [48]:
```
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 Projec
ts")
plt.title('Words for each summary of the project')
plt.xlabel('Number of words in each summary')
plt.legend()
plt.show()
```



Observation(s):

```
    1. Approved and Rejected projects tend to have same number of words in t
he summary
    2. Most number of project has word count between 5 to 20 and they've mor
e approval rate
    3. Median for Approved and Not approved projects have aproximately same
 median
```

1.3 Text preprocessing

1.3.1 Essay Text

In [49]:
```
project_data.head(2)
```

Out[49]:

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

2 rows × 21 columns

In [50]:
```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[2000])
print("="*50)
print(project_data['essay'].values[9999])
print("="*50)
```

```
My students are English learners that are working on English as their s
econd or third languages. We are a melting pot of refugees, immigrants,
and native-born Americans bringing the gift of language to our school.
\r\n\r\n We have over 24 languages represented in our English Learner p
rogram with students at every level of mastery.  We also have over 40 c
ountries represented with the families within our school.  Each student
brings a wealth of knowledge and experiences to us that open our eyes t
o new cultures, beliefs, and respect.\"The limits of your language are
the limits of your world.\"-Ludwig Wittgenstein  Our English learner's
have a strong support system at home that begs for more resources.  Man
y times our parents are learning to read and speak English along side o
f their children.  Sometimes this creates barriers for parents to be ab
le to help their child learn phonetics, letter recognition, and other r
eading skills.\r\n\r\nBy providing these dvd's and players, students ar
e able to continue their mastery of the English language even if no one
at home is able to assist.  All families with students within the Level
1 proficiency status, will be a offered to be a part of this program.
These educational videos will be specially chosen by the English Learne
r Teacher and will be sent home regularly to watch.  The videos are to
help the child develop early reading skills.\r\n\r\nParents that do not
have access to a dvd player will have the opportunity to check out a dv
d player to use for the year.  The plan is to use these videos and educ
ational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this y
ear all love learning, at least most of the time. At our school, 97.3%
of the students receive free or reduced price lunch. Of the 560 student
s, 97.3% are minority students. \r\nThe school has a vibrant community
that loves to get together and celebrate. Around Halloween there is a w
hole school parade to show off the beautiful costumes that students wea
r. On Cinco de Mayo we put on a big festival with crafts made by the st
udents, dances, and games. At the end of the year the school hosts a ca
rnival to celebrate the hard work put in during the school year, with a
dunk tank being the most popular activity.My students will use these fi
ve brightly colored Hokki stools in place of regular, stationary, 4-leg
ged chairs. As I will only have a total of ten in the classroom and not
enough for each student to have an individual one, they will be used in
a variety of ways. During independent reading time they will be used as
special chairs students will each use on occasion. I will utilize them
in place of chairs at my small group tables during math and reading tim
es. The rest of the day they will be used by the students who need the
highest amount of movement in their life in order to stay focused on sc
hool.\r\n\r\nWhenever asked what the classroom is missing, my students
always say more Hokki Stools. They can't get their fill of the 5 stools
we already have. When the students are sitting in group with me on the
Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki St
ools are the first to be taken. There are always students who head over
to the kidney table to get one of the stools who are disappointed as th
ere are not enough of them. \r\n\r\nWe ask a lot of students to sit for
7 hours a day. The Hokki stools will be a compromise that allow my stud
ents to do desk work and move at the same time. These stools will help
students to meet their 60 minutes a day of movement by allowing them to
activate their core muscles for balance while they sit. For many of my
students, these chairs will take away the barrier that exists in school
s for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environmen
t with plain walls, rows of desks, and a teacher in front of the room?
A typical day in our room is nothing like that. I work hard to create a
warm inviting themed room for my students look forward to coming to eac
h day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and gir
ls of mixed races in Arkansas.\r\nThey attend a Title I school, which m
eans there is a high enough percentage of free and reduced-price lunch
to qualify. Our school is an \"open classroom\" concept, which is very
unique as there are no walls separating the classrooms. These 9 and 10
year-old students are very eager learners; they are like sponges, absor
bing all the information and experiences and keep on wanting more.With
these resources such as the comfy red throw pillows and the whimsical n
autical hanging decor and the blue fish nets, I will be able to help cr
eate the mood in our classroom setting to be one of a themed nautical e
nvironment. Creating a classroom environment is very important in the s
uccess in each and every child's education. The nautical photo props wi
ll be used with each child as they step foot into our classroom for the
first time on Meet the Teacher evening. I'll take pictures of each chil
d with them, have them developed, and then hung in our classroom ready
for their first day of 4th grade.  This kind gesture will set the tone
before even the first day of school! The nautical thank you cards will
be used throughout the year by the students as they create thank you ca
rds to their team groups.\r\n\r\nYour generous donations will help me t
o help make our classroom a fun, inviting, learning environment from da
y one.\r\n\r\nIt costs lost of money out of my own pocket on resources
to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==================================================
Describing my students isn't an easy task.  Many would say that they ar
e inspirational, creative, and hard-working.  They are all unique - uni
que in their interests, their learning, their abilities, and so much mo
re.  What they all have in common is their desire to learn each day, de
spite difficulties that they encounter.  \r\nOur classroom is amazing -
because we understand that everyone learns at their own pace.  As the t
eacher, I pride myself in making sure my students are always engaged, m
otivated, and inspired to create their own learning! \r\nThis project i
s to help my students choose seating that is more appropriate for them,
developmentally.  Many students tire of sitting in chairs during lesson
s, and having different seats available helps to keep them engaged and
learning.\r\nFlexible seating is important in our classroom, as many of
our students struggle with attention, focus, and engagement.  We curren
tly have stability balls for seating, as well as regular chairs, but th
ese stools will help students who have trouble with balance, or find it
difficult to sit on a stability ball for a long period of time.  We are
excited to try these stools as a part of our engaging classroom communi
ty!nannan
==================================================
\"Creative Greatness\" is this school year's mantra to inspire my stude
nts to reach for the stars. I'm excited about ushering in an enthusiasm
and passion for growth in the visual arts department and inspiring stud
ents to consider and apply the purpose of art outside of the classroom.
\r\n\r\nMy art students and art club members are not just \"taking\" ar
t class, but are using their creativity to engage in school-wide beauti
fication projects and community initiatives. Help us to explore a great
er variety of art media and technology in my Art 1  classes to ignite s
tudent's interest in furthering their studies in art. Our large student
body limits funding to the arts, so charitable donations are crucial to
```

```
our growth into Advanced Placement and College and Career Readiness pro
grams in the arts.Our class will create personalized and unique interac
tive notebooks to encourage the development of independent learners and
writers. Interactive notebooks are not just used for class notes, but a
lso for daily learning activities that require students to process the
information presented in class and then organize the content in a manne
r that will reinforce their learning. \r\nInteractive Notebooks are a c
ross curricular tool that supports literacy in all content areas. In ou
r art class, these notebooks are used not only as an affordable sketchb
ook option, but also as an \"all things art\" guide that students can c
ontinue to reference throughout the school year and as they continue st
udies of more advanced art courses. We use our interactive notebooks to
write art critiques in response to viewing the works of famous artists
and to write art statements in response to the student's personal artwo
rk. We also use interactive notebooks to build vocabulary skills with e
ngaging activities to learn about the elements and principles of art to
go far beyond just defining the terms.  Students are required to chose
thinking maps that best organize the information presented in the lesso
n to teach lifelong skills of literacy and note-taking. \r\nStudents' i
nterest in using interactive notebooks is positively impacted when they
are able to be creative and personalize the look of their notebooks. En
gagement will no doubt be dramatically increased with fun and colorful
notebook covers and pages for each lesson. With this note-taking proces
s, students will learn organization, color coding, summarizing, and oth
er important skills while creating personalized portfolios of their ind
ividual learning that they can reference throughout the year.nannan
==================================================
```

In [51]:

```python
# 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 [52]:
```
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

```
Describing my students is not an easy task.  Many would say that they a
re inspirational, creative, and hard-working.  They are all unique - un
ique in their interests, their learning, their abilities, and so much m
ore.  What they all have in common is their desire to learn each day, d
espite difficulties that they encounter.  \r\nOur classroom is amazing
- because we understand that everyone learns at their own pace.  As the
teacher, I pride myself in making sure my students are always engaged,
motivated, and inspired to create their own learning! \r\nThis project
is to help my students choose seating that is more appropriate for the
m, developmentally.  Many students tire of sitting in chairs during les
sons, and having different seats available helps to keep them engaged a
nd learning.\r\nFlexible seating is important in our classroom, as many
of our students struggle with attention, focus, and engagement.  We cur
rently have stability balls for seating, as well as regular chairs, but
these stools will help students who have trouble with balance, or find
it difficult to sit on a stability ball for a long period of time.  We
are excited to try these stools as a part of our engaging classroom com
munity!nannan
==================================================
```

In [53]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove
-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
Describing my students is not an easy task.  Many would say that they a
re inspirational, creative, and hard-working.  They are all unique - un
ique in their interests, their learning, their abilities, and so much m
ore.  What they all have in common is their desire to learn each day, d
espite difficulties that they encounter.    Our classroom is amazing -
because we understand that everyone learns at their own pace.  As the t
eacher, I pride myself in making sure my students are always engaged, m
otivated, and inspired to create their own learning!   This project is
to help my students choose seating that is more appropriate for them, d
evelopmentally.  Many students tire of sitting in chairs during lesson
s, and having different seats available helps to keep them engaged and
learning.  Flexible seating is important in our classroom, as many of o
ur students struggle with attention, focus, and engagement.  We current
ly have stability balls for seating, as well as regular chairs, but the
se stools will help students who have trouble with balance, or find it
difficult to sit on a stability ball for a long period of time.  We are
excited to try these stools as a part of our engaging classroom communi
ty!nannan
```

In [54]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Describing my students is not an easy task Many would say that they are
inspirational creative and hard working They are all unique unique in t
heir interests their learning their abilities and so much more What the
y all have in common is their desire to learn each day despite difficul
ties that they encounter Our classroom is amazing because we understand
that everyone learns at their own pace As the teacher I pride myself in
making sure my students are always engaged motivated and inspired to cr
eate their own learning This project is to help my students choose seat
ing that is more appropriate for them developmentally Many students tir
e of sitting in chairs during lessons and having different seats availa
ble helps to keep them engaged and learning Flexible seating is importa
nt in our classroom as many of our students struggle with attention foc
us and engagement We currently have stability balls for seating as well
as regular chairs but these stools will help students who have trouble
with balance or find it difficult to sit on a stability ball for a long
period of time We are excited to try these stools as a part of our enga
ging classroom community nannan
```

In [55]:

```
# 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", 'it
s', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'thi
s', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'ha
ve', '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', 'i
nto', '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', 'ho
w', '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", 'd
idn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is
n'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 [56]:

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

```
100%|██████████| 15000/15000 [00:07<00:00, 1896.52it/s]
```

In [57]:

```
# after preprocesing
preprocessed_essays[2000]
```

Out[57]:

```
'describing students not easy task many would say inspirational creativ
e hard working they unique unique interests learning abilities much wha
t common desire learn day despite difficulties encounter our classroom
amazing understand everyone learns pace as teacher i pride making sure
students always engaged motivated inspired create learning this project
help students choose seating appropriate developmentally many students
tire sitting chairs lessons different seats available helps keep engage
d learning flexible seating important classroom many students struggle
attention focus engagement we currently stability balls seating well re
gular chairs stools help students trouble balance find difficult sit st
ability ball long period time we excited try stools part engaging class
room community nannan'
```

1.3.2 Project title Text

In [58]:

```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for title in tqdm(project_data['project_title'].values):
    sent = decontracted(title)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 15000/15000 [00:00<00:00, 42837.22it/s]
```

In [59]:
```
# after preprocessing
preprocessed_titles[1000]
```

Out[59]:
```
'sailing into super 4th grade year'
```

1. 4 Preparing data for models

In [60]:
```
project_data.columns
```

Out[60]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_stat
e',
       'project_submitted_datetime', 'project_grade_category', 'project
_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_appr
oved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'qu
antity',
       'count_of_digits_in_summary'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data

- quantity : numerical
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

1.4.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/ (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)

In [61]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())


categories_one_hot = vectorizer.transform(project_data['clean_categories'].values)
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearn
ing', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Langua
ge']
Shape of matrix after one hot encodig  (15000, 9)
```

In [62]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot = vectorizer.transform(project_data['clean_subcategories'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'FinancialLiteracy', 'CommunityService', 'ParentInvolveme
nt', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutri
tionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingA
rts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPre
p', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopme
nt', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Healt
h_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing',
'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (15000, 30)
```

In [63]:

```
# Please do the similar feature encoding with state, teacher_prefix and
 project_grade_category also

# Feature encoding for state

vectorizer = CountVectorizer(lowercase=False, binary=True)
print (project_data['school_state'].head(5))
vectorizer.fit(project_data['school_state'].values)
print (vectorizer.get_feature_names())

states_one_hot = vectorizer.transform(project_data['school_state'].value
s)
print("Shape of matrix after one hot encodig ",states_one_hot.shape)
```

```
0     IN
1     FL
2     AZ
3     KY
4     TX
Name: school_state, dtype: object
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'H
I', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI',
'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY',
'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT',
'WA', 'WI', 'WV', 'WY']
Shape of matrix after one hot encodig  (15000, 51)
```

In [64]:

```
# Feature encoding for teacher_prefix

# Found some NaN values for teacher_prefix so applying 'most_frequent' i
mputer
# and replacing the values with Most feequently occured values

# from sklearn.impute import SimpleImputer

# imp_mean = SimpleImputer(missing_values=np.nan, strategy='most_frequen
t')
# project_data = imp_mean.fit_transform(project_data['teacher_prefix'])
```

In [65]:

```
print ('Nan Values:',project_data['teacher_prefix'].isnull().sum())
# Replacing the NaN values with most frequently used value of teacher prefix

project_data.loc[project_data['teacher_prefix'].isnull(),'teacher_prefix']='Mrs.'
print ('After Imputing:',project_data['teacher_prefix'].isnull().sum())


vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'])
print (vectorizer.get_feature_names())

teacher_prfx_one_hot = vectorizer.transform(project_data['teacher_prefix'])
print("Shape of matrix after one hot encodig ",teacher_prfx_one_hot.shape)
```

```
Nan Values: 1
After Imputing: 0
['Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (15000, 4)
```

In [66]:

```
# Feature encoding for project_grade_category

my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.split(','))

prjctgrd_dict = dict(my_counter)
sorted_prjctgrd_dict = dict(sorted(prjctgrd_dict.items(), key=lambda kv: kv[1]))

vectorizer = CountVectorizer(vocabulary=list(sorted_prjctgrd_dict.keys()),lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print (vectorizer.get_feature_names())

project_grade_category_one_hot = vectorizer.transform(project_data['project_grade_category'].values)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot.shape)
```

```
['Grades 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2']
Shape of matrix after one hot encodig  (15000, 4)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [67]:

```
# 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  (15000, 7465)
```

1.4.2.2 Bag of Words on `project_title`

In [68]:

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

```
Shape of matrix after one hot encodig  (15000, 912)
```

1.4.2.3 TFIDF vectorizer

In [69]:

```
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  (15000, 7465)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [70]:

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

```
Shape of matrix after one hot encodig  (15000, 912)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [71]:

```
'''
# 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 ou
r coupus", \
      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.c
om/how-to-use-pickle-to-save-and-load-variables-in-python/

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

```
'''
```

Out[71]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230
349/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove
Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}
\n    for line in tqdm(f):\n        splitLine = line.split()\n        w
ord = splitLine[0]\n        embedding = np.array([float(val) for val in
splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",le
n(model)," words loaded!")\n    return model\nmodel = loadGloveModel
(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n
\nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  w
ords loaded!\n\n# ============================\n\nwords = []\nfor i in
preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in preproce
d_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in th
e coupus", len(words))\nwords = set(words)\nprint("the unique words in
the coupus", len(words))\n\ninter_words = set(model.keys()).intersectio
n(words)\nprint("The number of words that are present in both glove vec
tors and our coupus",       len(inter_words),"(",np.round(len(inter_wor
ds)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(mo
del.keys())\nfor i in words:\n    if i in words_glove:\n        words_c
ourpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n
\n\n# stronging variables into pickle files python: http://www.jessicay
ung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimpo
rt pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump
(words_courpus, f)\n\n\n'
```

In [72]:

```
# stronging variables into pickle files python: http://www.jessicayung.c
om/how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [73]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored i
n this list
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/rev
iew
    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%|██████████| 15000/15000 [00:04<00:00, 3417.84it/s]

15000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [74]:

```
# average Word2Vec
# compute average word2vec for each title.
avg_w2v_vectors_title = []; # the avg-w2v for each title is stored in th
is list
for sentence in tqdm(preprocessed_titles): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/rev
iew
    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_title.append(vector)

print(len(avg_w2v_vectors_title))
print(len(avg_w2v_vectors_title[0]))
```

```
100%|██████████| 15000/15000 [00:00<00:00, 55256.41it/s]

15000
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [113]:
```python
# 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 [114]:
```python
# 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[1:100]): # 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 value((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%|██████████| 99/99 [00:00<00:00, 357.27it/s]

99
300
```

1.4.2.9 Using Pretrained Models: TFIDF weighted W2V on `project_title`

In [115]:
```python
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
# 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 [123]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm(preprocessed_titles): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentenc
e/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 th
e tf value((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_titles.append(vector)

print(len(tfidf_w2v_vectors_titles))
print(len(tfidf_w2v_vectors_titles[0]))
```

```
100%|██████████| 15000/15000 [00:00<00:00, 25444.26it/s]

15000
300
```

1.4.3 Vectorizing Numerical features

In [79]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/gener
ated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 21
3.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding t
he mean and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(pr
ice_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values
.reshape(-1, 1))
```

```
Mean : 297.8444793333333, Standard deviation : 383.6922825999444
```

In [80]:
```
price_standardized
```

Out[80]:
```
array([[-0.37333167],
       [ 0.00301158],
       [ 0.57078427],
       ...,
       [-0.31906943],
       [-0.26024625],
       [-0.48414964]])
```

1.4.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [81]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(15000, 9)
(15000, 30)
(15000, 7465)
(15000, 1)
```

In [82]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/408403
9
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and
a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_
standardized))
X.shape
```

Out[82]:
```
(15000, 7505)
```

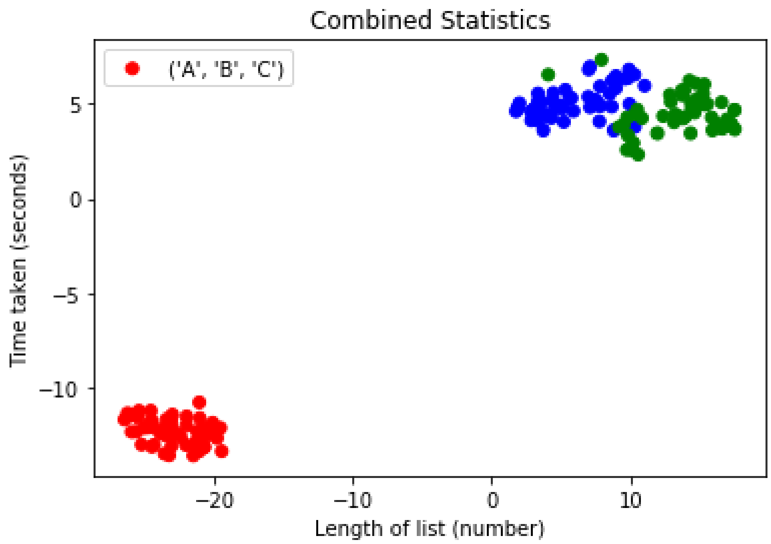
Assignment 2: Apply TSNE

If you are using any code snippet from the internet, you have to provide the reference/citations, as we did in the above cells. Otherwise, it will be treated as plagiarism without citations.

1. In the above cells we have plotted and analyzed many features. Please observe the plots and write the observations in markdown cells below every plot.
2. EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects
3. Build the data matrix using these features
    - school_state : categorical data (one hot encoding)
    - clean_categories : categorical data (one hot encoding)
    - clean_subcategories : categorical data (one hot encoding)
    - teacher_prefix : categorical data (one hot encoding)
    - project_grade_category : categorical data (one hot encoding)
    - project_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)
    - price : numerical
    - teacher_number_of_previously_posted_projects : numerical
4. Now, plot FOUR t-SNE plots with each of these feature sets.
    A. categorical, numerical features + project_title(BOW)
    B. categorical, numerical features + project_title(TFIDF)
    C. categorical, numerical features + project_title(AVG W2V)
    D. categorical, numerical features + project_title(TFIDF W2V)
5. Concatenate all the features and Apply TNSE on the final data matrix
6. Note 1: The TSNE accepts only dense matrices
7. Note 2: Consider only 5k to 6k data points to avoid memory issues. If you run into memory error issues, reduce the number of data points but clearly state the number of datat-poins you are using

In [83]:

```python
# this is the example code for TSNE
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

iris = datasets.load_iris()
x = iris['data']
y = iris['target']

tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue', 2:'green'}
legends = {0:'A', 1:'B', 2:'C'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]), label=('A','B','C'))
plt.legend()
plt.title("Combined Statistics")
plt.xlabel("Length of list (number)")
plt.ylabel("Time taken (seconds)")

plt.show()
```

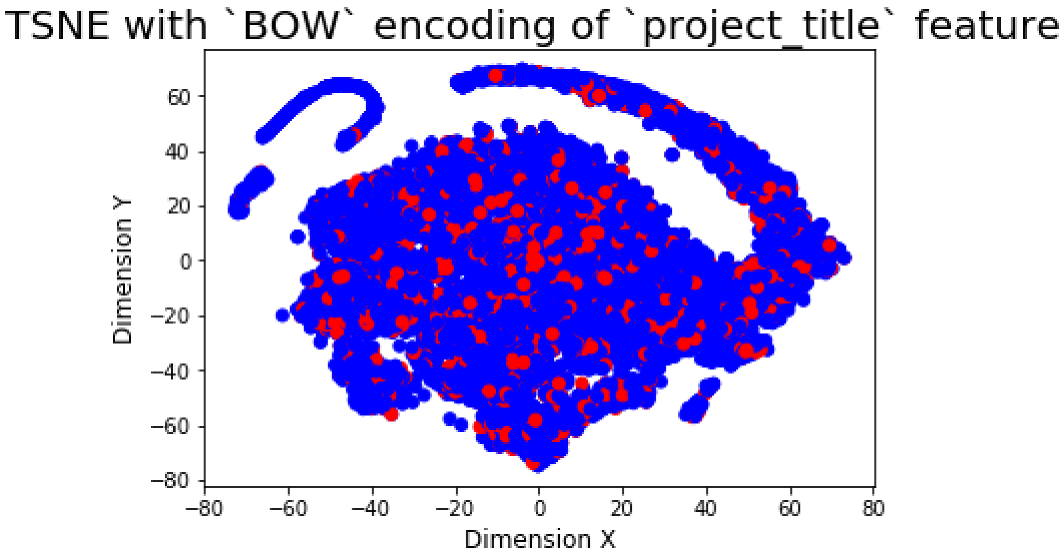


2.1 TSNE with `BOW` encoding of `project_title` feature

In [125]:

```
# please write all of the code with proper documentation and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label


# preparing datamatrix with categorical, numerical features + project_title(BOW)

print (states_one_hot.shape)
print (categories_one_hot.shape)
print (sub_categories_one_hot.shape)
print (teacher_prfx_one_hot.shape)
print (project_grade_category_one_hot.shape)
print (price_standardized.shape)
print (title_bow.shape)

previously_posted_projects = project_data['teacher_number_of_previously_posted_projects'].values

print (previously_posted_projects.reshape(-1,1).shape)


bow_data = hstack((states_one_hot,categories_one_hot,sub_categories_one_hot,teacher_prfx_one_hot,project_grade_category_one_hot,price_standardized,previously_posted_projects.reshape(-1,1),title_bow))

print (bow_data.shape)

tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

X_embedding = tsne.fit_transform(bow_data.toarray())

for_tsne = np.hstack((X_embedding, project_data['project_is_approved'].values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))

plt.xlabel('Dimension X', fontsize=12)
plt.ylabel('Dimension Y', fontsize=12)
plt.title ('TSNE with `BOW` encoding of `project_title` feature ', fontsize=20)
plt.show()
```

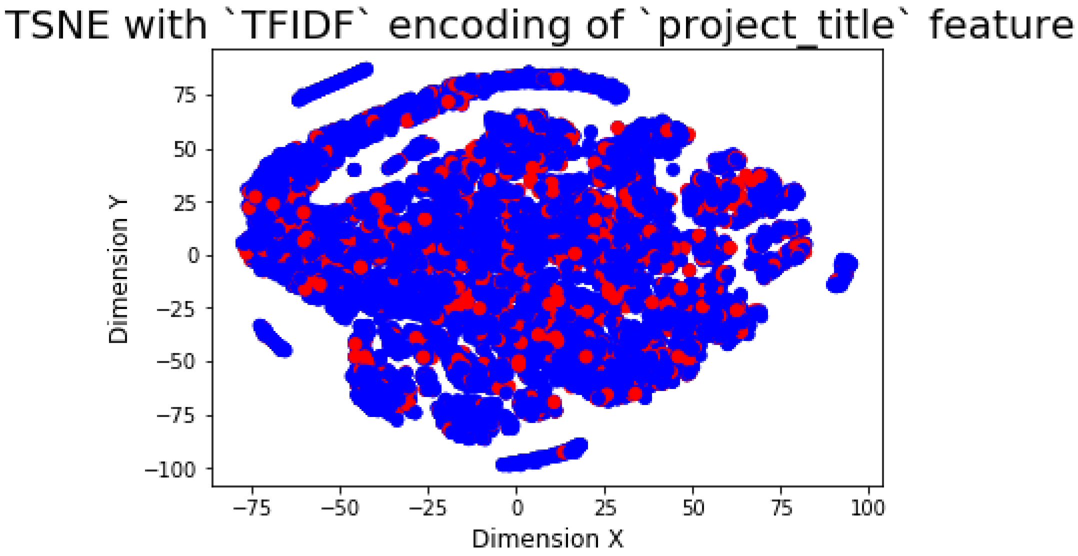
```
(15000, 51)
(15000, 9)
(15000, 30)
(15000, 4)
(15000, 4)
(15000, 1)
(15000, 912)
(15000, 1)
(15000, 1012)
```



2.2 TSNE with `TFIDF` encoding of `project_title` feature

In [109]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

tfidf_data = hstack((states_one_hot,categories_one_hot,sub_categories_one_hot,teacher_prfx_one_hot,project_grade_category_one_hot,price_standardized,previously_posted_projects.reshape(-1,1),title_tfidf))

print (tfidf_data.shape)

tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

X_embedding = tsne.fit_transform(tfidf_data.toarray())

for_tsne = np.hstack((X_embedding, project_data['project_is_approved'].values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))

plt.xlabel('Dimension X', fontsize=12)
plt.ylabel('Dimension Y', fontsize=12)
plt.title ('TSNE with `TFIDF` encoding of `project_title` feature ', fontsize=20)
plt.show()
```

```
(15000, 1012)
```

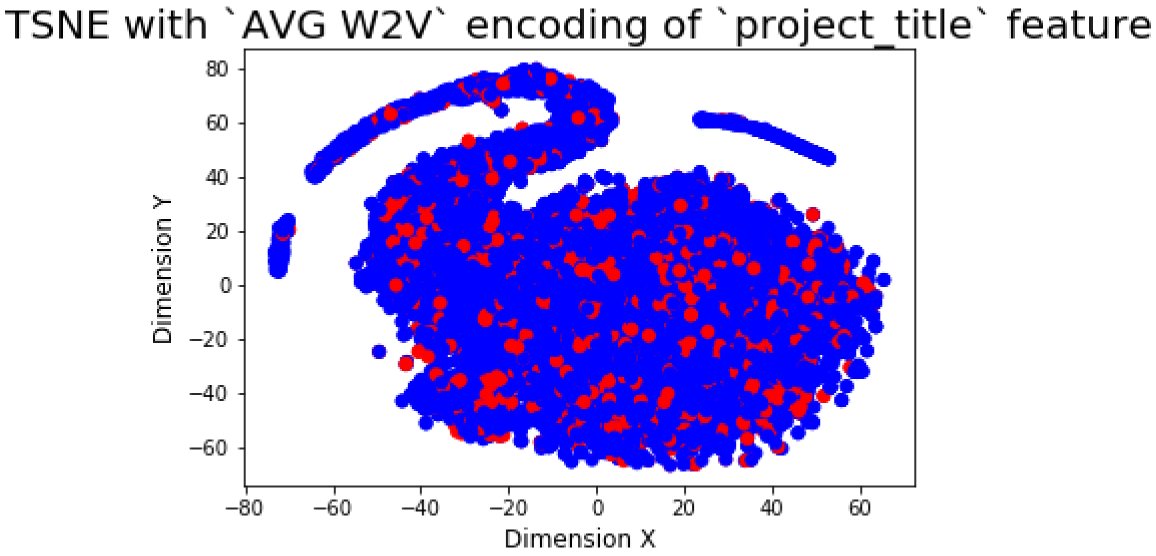


2.3 TSNE with `AVG W2V` encoding of `project_title` feature

In [110]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

avgw2v_data = hstack((states_one_hot,categories_one_hot,sub_categories_one_hot,teacher_prfx_one_hot,project_grade_category_one_hot,price_standardized,previously_posted_projects.reshape(-1,1),avg_w2v_vectors_title))

print (avgw2v_data.shape)

tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

X_embedding = tsne.fit_transform(avgw2v_data.toarray())

for_tsne = np.hstack((X_embedding, project_data['project_is_approved'].values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))

plt.xlabel('Dimension X', fontsize=12)
plt.ylabel('Dimension Y', fontsize=12)
plt.title ('TSNE with `AVG W2V` encoding of `project_title` feature', fontsize=20)
plt.show()
```

```
(15000, 400)
```

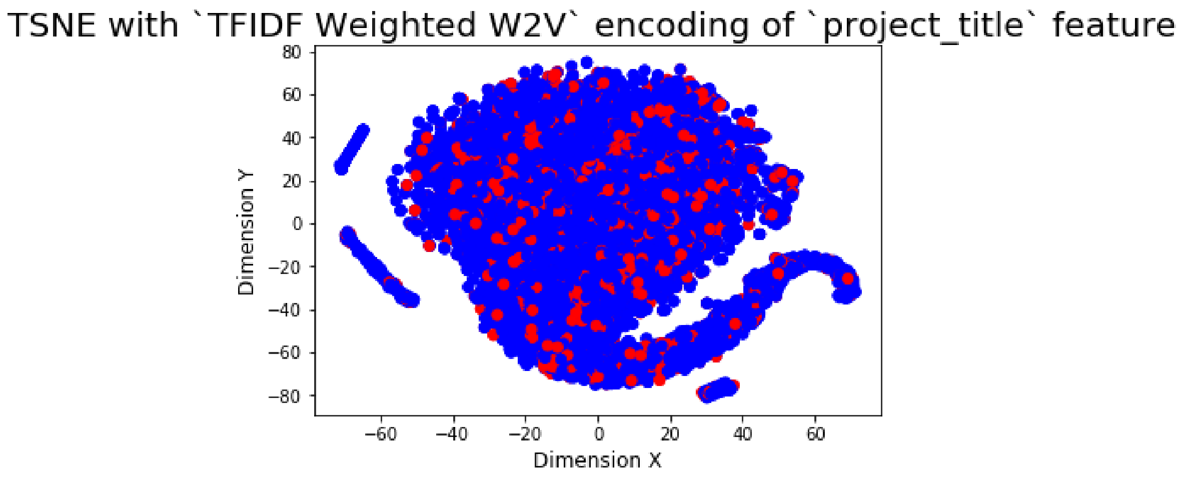


2.4 TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature

In [124]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label




tfidfw2v_data = hstack((states_one_hot,categories_one_hot,sub_categories_one_hot,teacher_prfx_one_hot,project_grade_category_one_hot,price_standardized,previously_posted_projects.reshape(-1,1),tfidf_w2v_vectors_titles))

print (tfidfw2v_data.shape)

tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

X_embedding = tsne.fit_transform(tfidfw2v_data.toarray())

for_tsne = np.hstack((X_embedding, project_data['project_is_approved'].values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))

plt.xlabel('Dimension X', fontsize=12)
plt.ylabel('Dimension Y', fontsize=12)
plt.title ('TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature ', fontsize=20)
plt.show()
```

```
(15000, 400)
```



2.5 Summary

Write few sentences about the results that you obtained and the observations you made.

1. Every state has greater than 80% success rate in approval
2. Approval rate is low for Teacher. and More for Mrs.
3. Approvale rate is more for PreK-2
4. For all grades approval rate is more than 83%
5. Literacy_Language and Math_Science has more Approval Rate
6. Literacy and Mathematics subcategories has more approval rate
7. If the Title contains more numbers of words it has more chances of getting approved.
8. Approved projects has more number of words in the essay's
9. Approved projects tend to have less cost compared to Not approved projects
10. Most of the projects are submitted by new teachers and has success rate of 82%
11. Most number of project has word count between 5 to 20 and they've more approval rate

In [ ]: