DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example**
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targete enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories f following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal ((https://en.wikipedia.org/wiki/List_of_U.S._state_abbr **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategori **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the projec • `My students need hands on literacy mater sensory needs!`

Feature	Description
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Ex** 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed pro bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated value • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted b **Example:** 2

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_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_prefix'
'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

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

Out[4]:

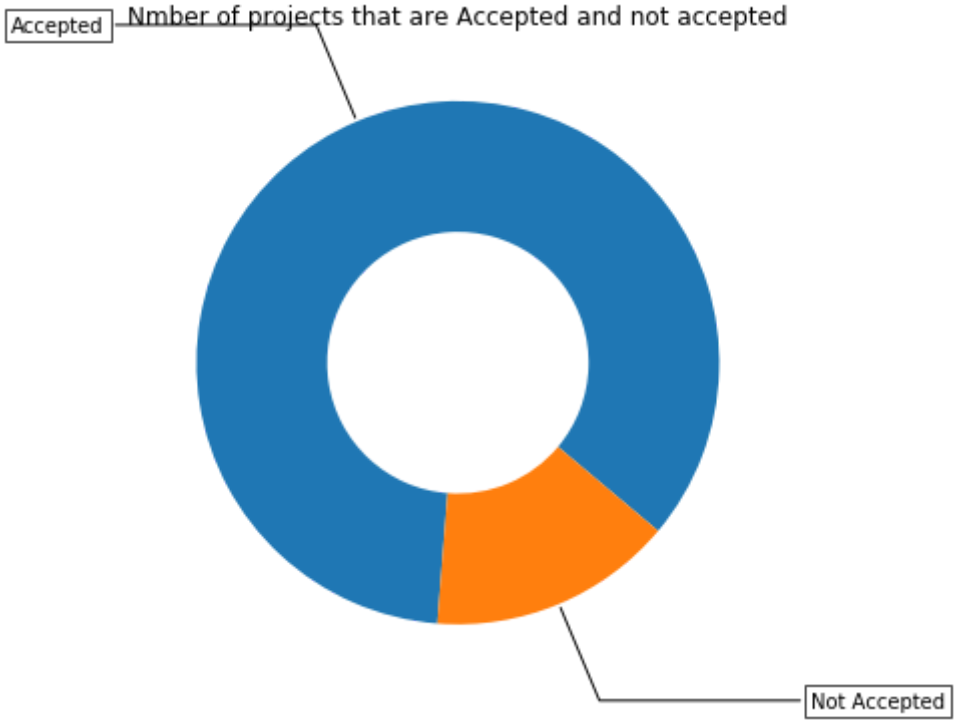
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

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_labels.html#sphx-gl
r-gallery-pie-and-polar-charts-pie-and-donut-labels-py


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

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

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

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

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

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

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

plt.show()
```

```
Number of projects thar are approved for funding  92706 , ( 84.85830404217
927 %)
Number of projects thar are not approved for funding  16542 , ( 15.1416959
57820739 %)
```



In [6]:

```
#observation:for given number of projects 15% projects not approved for funding.while 8
5% projects get approved for funding
```

1.2.1 Univariate Analysis: School State

In [7]:

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

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

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

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,14
3)']]

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[7]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9
620\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, \'r
gb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\ndata = [ dict(\n        t
ype=\'choropleth\',\n        colorscale = scl,\n        autocolorscale = F
alse,\n        locations = temp[\'state_code\'],\n        z = temp[\'num_p
roposals\'].astype(float),\n        locationmode = \'USA-states\',\n
text = temp[\'state_code\'],\n        marker = dict(line = dict (color =
\'rgb(255,255,255)\',width = 2)),\n        colorbar = dict(title = "% of p
ro")\n    ) ]\n\nlayout = dict(\n        title = \'Project Proposals % of
Acceptance Rate by US States\',\n        geo = dict(\n            scope=
\'usa\',\n            projection=dict( type=\'albers usa\' ),\n
showlakes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n
),\n    )\n\nfig = go.Figure(data=data, layout=layout)\noffline.iplot(fig,
filename=\'us-map-heat-map\')\n'
```

In [8]:

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

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

In [9]:

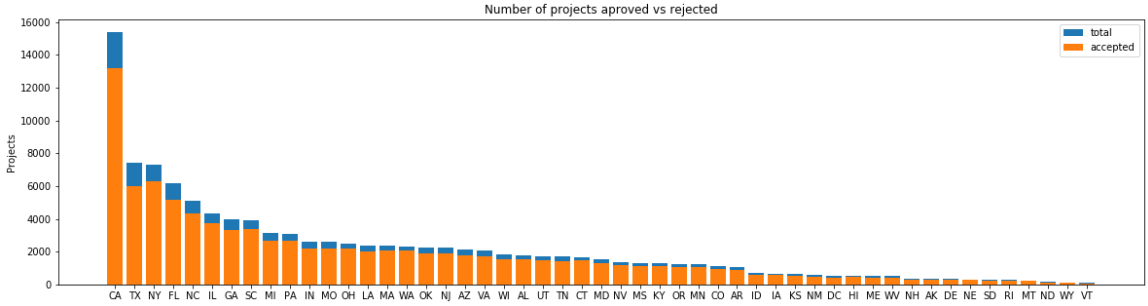
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/ba
r_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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 [10]:

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

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

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

    if top:
        temp = temp[0:top]

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

In [11]:

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




```
   school_state  project_is_approved  total       Avg
4            CA                13205  15388  0.858136
43           TX                 6014   7396  0.813142
34           NY                 6291   7318  0.859661
9            FL                 5144   6185  0.831690
27           NC                 4353   5091  0.855038
==================================================
   school_state  project_is_approved  total       Avg
39           RI                  243    285  0.852632
26           MT                  200    245  0.816327
28           ND                  127    143  0.888112
50           WY                   82     98  0.836735
46           VT                   64     80  0.800000
```

In [12]:

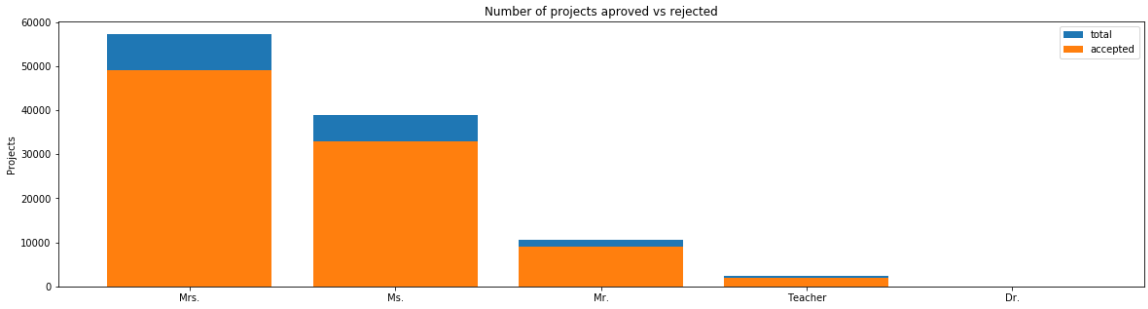
```
#observation:
#1.we can see state CA has posted high number of projects with approval rate of 85%
#2.state VT has posted less number of projects only 80 projects and get 60 projects approved
```

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

1.2.2 Univariate Analysis: teacher_prefix

In [13]:

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




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

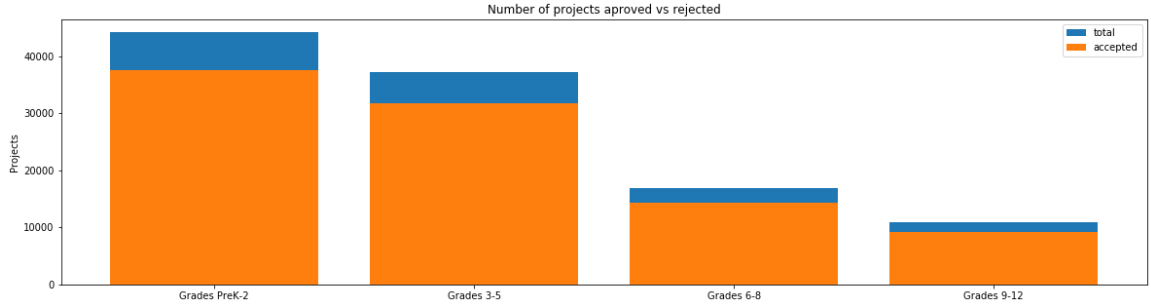
In [14]:

```
#observation:
#1.There are total 57269 projects posted  for teacher_prefix "mrs.". Approval rate is 8
5%
#2.There are only 13 projects posted for teacher_prefix 'Dr.'. Approval rate is 69%
```

1.2.3 Univariate Analysis: project_grade_category

In [15]:

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




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

In [16]:

```
#observation:
#1.for all grade category, there are more than 80% approval rate
#2.for grades prek-2 and 3-5 there are high number of project posted and their approval
rate of project is 85%
#3.lowest approval rate is for grade 9-12 that is 83%
```

1.2.4 Univariate Analysis: project_subject_categories

In [17]:

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

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

In [18]:

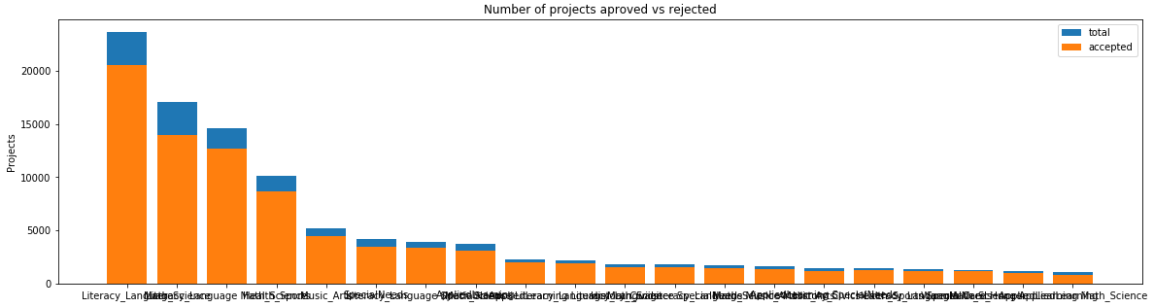
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[18]:

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

In [19]:

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




```
                 clean_categories  project_is_approved  total       Avg
24               Literacy_Language                20520  23655  0.867470
32                    Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12725  14636  0.869432
8                    Health_Sports                 8640  10177  0.848973
40                      Music_Arts                 4429   5180  0.855019
==================================================
                   clean_categories  project_is_approved  total       Avg
19  History_Civics Literacy_Language                 1271   1421  0.894441
14        Health_Sports SpecialNeeds                 1215   1391  0.873472
50                Warmth Care_Hunger                 1212   1309  0.925898
33      Math_Science AppliedLearning                 1019   1220  0.835246
4       AppliedLearning Math_Science                  855   1052  0.812738
```

In [20]:

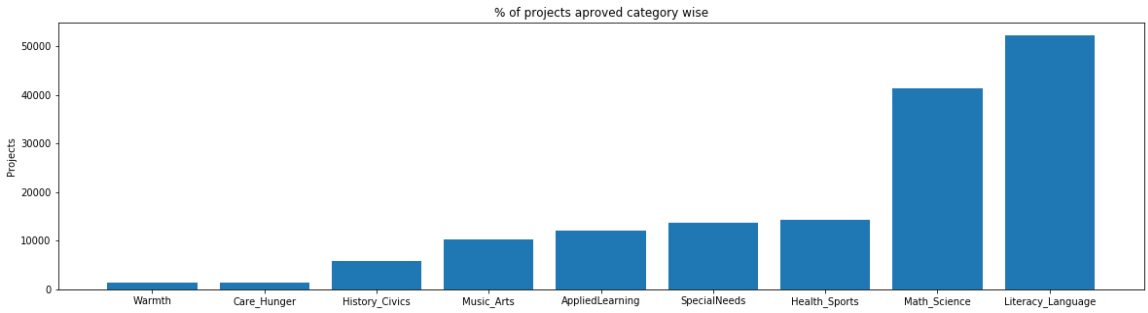
```
#observation:
#1.23655 projects are posted for subject categories-Literacy_Language. Approval rate is
86%
#2.92% is highest approval rate for subject categories-warmth care_hunger.1212 out of 1
309 projects get approved for funding
#3.89% is the second highest approval rate for subject categories-History_Civics Litera
cy_Language. Total 1421 projects are posted for this category
```

In [21]:

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

In [22]:

```
# 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 [23]:

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

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

In [24]:

```
#observation:
#1.maximum number of words count is for literacy_language
#2.minimum number of words count is for warmth & care_hunger
```

1.2.5 Univariate Analysis: project_subject_subcategories

In [25]:

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

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

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

In [26]:

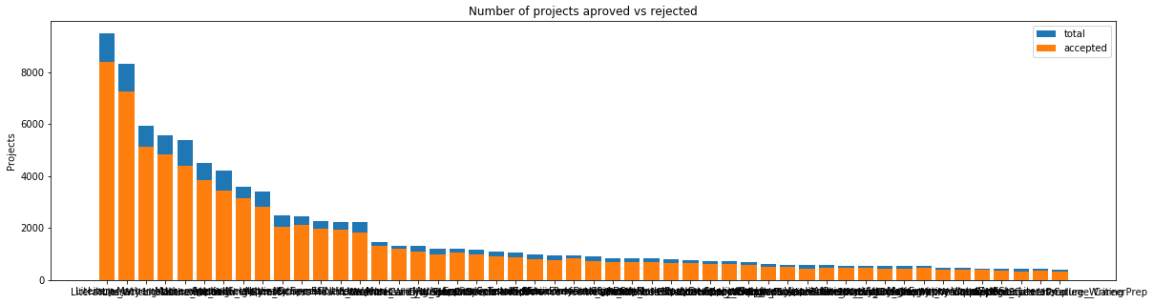
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[26]:

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

In [27]:

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




```
              clean_subcategories  project_is_approved  total       Avg
317                       Literacy                 8371   9486  0.882458
319           Literacy Mathematics                 7260   8325  0.872072
331  Literature_Writing Mathematics                 5140   5923  0.867803
318     Literacy Literature_Writing                 4823   5571  0.865733
342                     Mathematics                 4385   5379  0.815207
==================================================
                   clean_subcategories  project_is_approved  total
Avg
196       EnvironmentalScience Literacy                  389    444  0.876
126
127                                ESL                  349    421  0.828
979
79                  College_CareerPrep                  343    421  0.814
727
17   AppliedSciences Literature_Writing                 361    420  0.859
524
3    AppliedSciences College_CareerPrep                 330    405  0.814
815
```

In [28]:

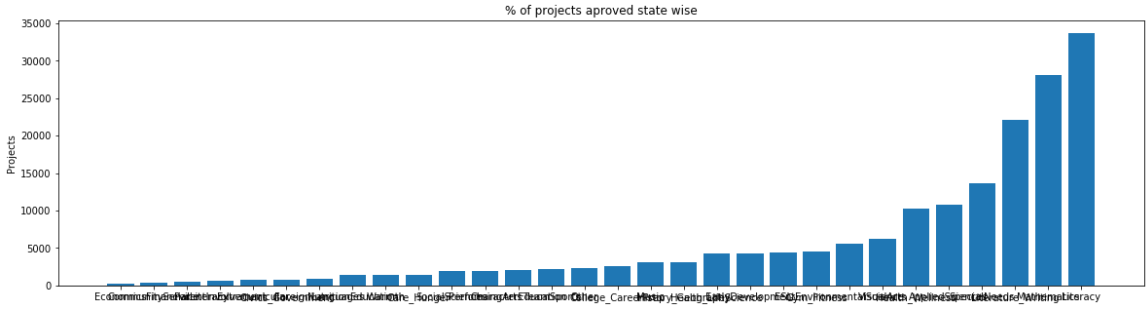
```
#1.Highest approval rate is for subject subcategories-Literacy(i.e. 88%)
#2.lowest approval rate is for subject subcategories-AppliedSciences College_CareerPrep
(i.e. 81%)
```

In [29]:

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

In [30]:

```
# 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 [31]:

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

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

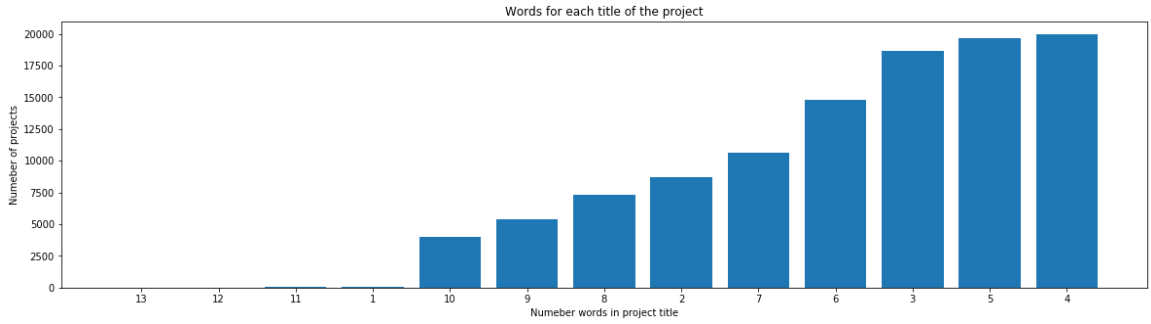
In [32]:

```
#observation:
#1.maximum number of words count is for Literacy(i.e.33700)
#2.minimum number of words count is for Economics(i.e.269)
```

1.2.6 Univariate Analysis: Text features (Title)

In [33]:

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



In [34]:

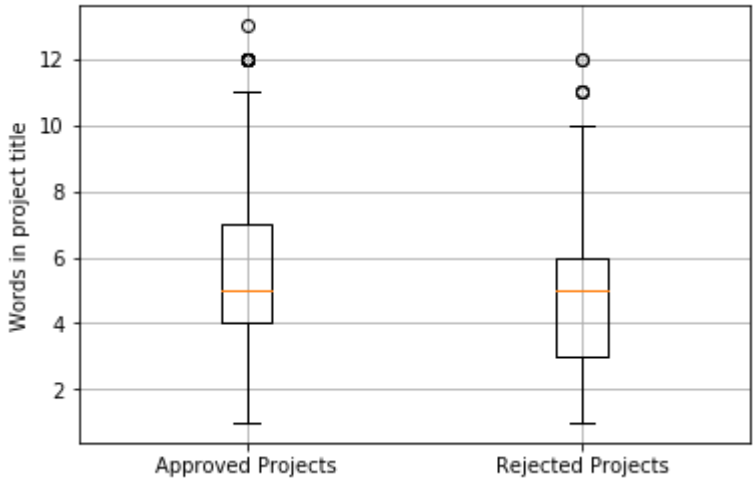
```
#observation:
#1.most of the projects title have 4 and 5 words
#2.very few projects are having 1 and 11 words for the project title
```

In [35]:

```
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 [36]:

```
# 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()
```

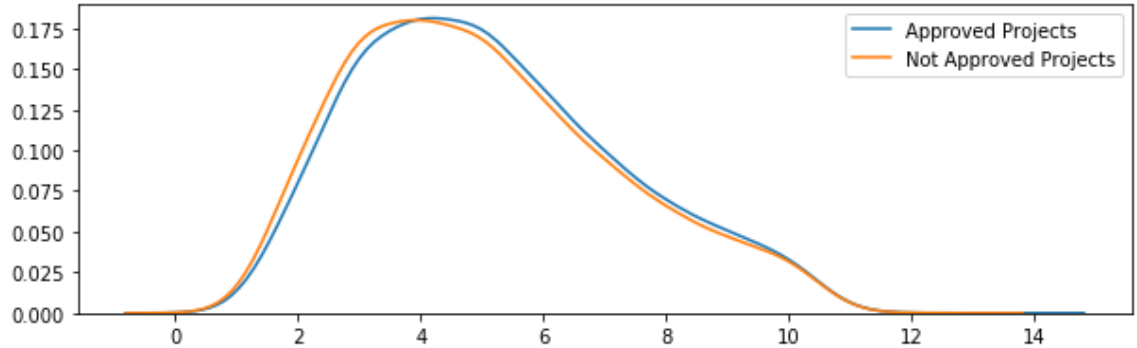


In [37]:

```
#observation:
#1.projects title having number of words between 6 to 8 are mostly approverd.
#2.projects title having number of words between 2 to 4 are mostly rejected.
#3.projects title having number of words between 4 to 6, nothing can be concluded with
 surity by looking the above plot
```

In [38]:

```
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()
```



In [39]:

```
#observations:
#1.pdf for approved projects is slightly ahead of pdf of rejected projects for number o
f words between 4 to 14
#2.pdf for rejected projects is slightly ahead of pdf of approved projects for number o
f words between 0 to 4
```

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

In [40]:

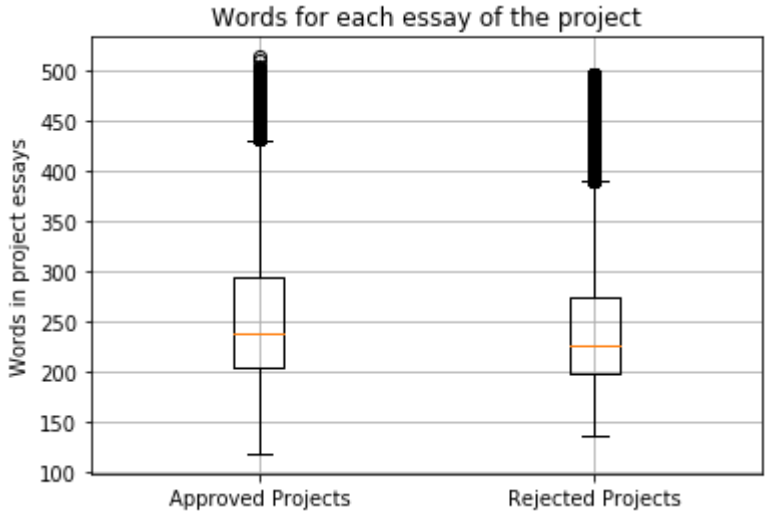
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [41]:

```
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 [42]:

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

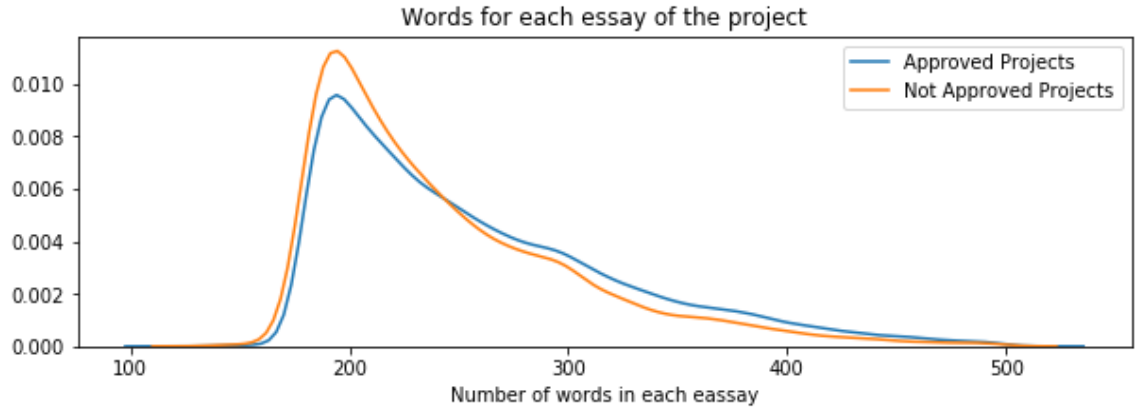


In [43]:

```
#observation:nothing much effective can be concluded from this plot for approval or rej
ection of projects on the basis of number of words in the easy of the project
```

In [44]:

```
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()
```



In [45]:

```
#observation:
#1.pdf for approved projects is slightly ahead of pdf of rejected projects for number o
f words between 250 to 500
#2.pdf for rejected projects is slightly ahead of pdf of approved projects for number o
f words between 100 to 250
```

1.2.8 Univariate Analysis: Cost per project

In [46]:

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

Out[46]:

	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 [47]:

```
# 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[47]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [48]:

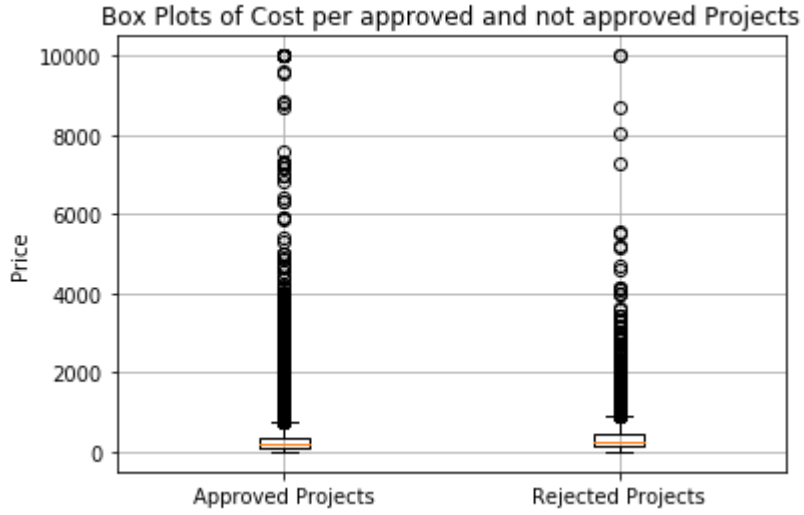
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [49]:

```
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 [50]:

```
# 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()
```

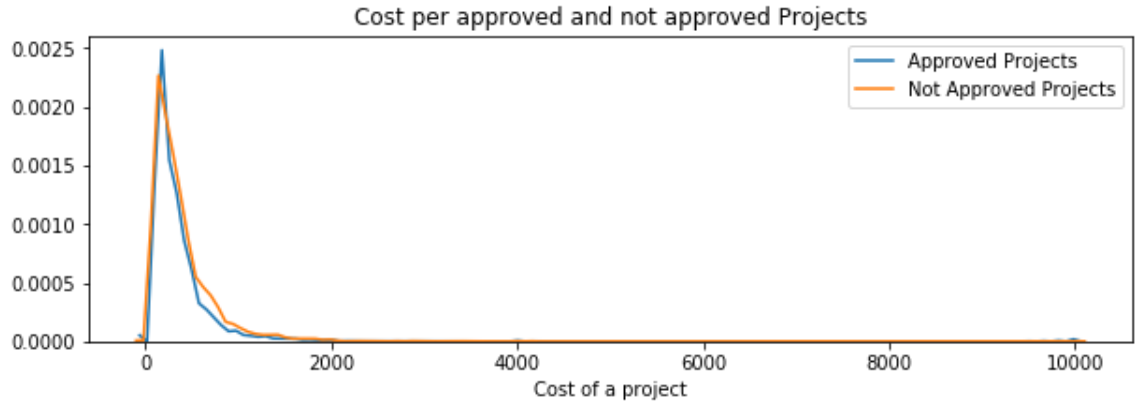


In [51]:

```
#observation:nothing much effective can be said from this plot about approval or rejection of projects on the basis of cost per projects
```

In [52]:

```
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 [53]:

```
#observation:Both pdfs are overlapping greatly.So nothing can be said about approval or rejection of projects on the basis of cost of a projects
```

In [54]:

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

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prett
ytable

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

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

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

In [55]:

```
#observation:
#1.If we talk about 5th percentile then approved projects have lower cost than not appr
oved projects
#2.similary for 50 th percentile the approved project have lower cost than not approved
projects
#3.one thing can be concluded that for each of the percentile approved projects have lo
wer cost than 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 [56]:

```
temp = pd.DataFrame(project_data.groupby("teacher_number_of_previously_posted_projects"
)["project_is_approved"].apply(np.size)).reset_index()
# if you have data which contain only 0 and 1, then the mean = percentage (think about
 it)
temp.columns = ['previously_posted_projects', 'by_no_of_teachers']
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.p
df
print("number of teachers with highest previously posted project")
print(temp.head(5))
print('='*50)
print("number of teachers with lowest previously posted project")
print(temp.tail(5))
```

```
number of teachers with highest previously posted project
   previously_posted_projects  by_no_of_teachers
0                           0              30014
1                           1              16058
2                           2              10350
3                           3               7110
4                           4               5266
==================================================
number of teachers with lowest previously posted project
     previously_posted_projects  by_no_of_teachers
369                         428                  1
370                         432                  1
371                         433                  1
372                         437                  1
373                         451                  1
```

In [57]:

```
#observation:
#     1.we can see there are 1363 teachers who did not previously posted any projects.S
imilarly there is only one teachers who previously posted 354 projects
#     2.likewise we can see there are 243 teachers who posted 4 projects previously
```

In [58]:

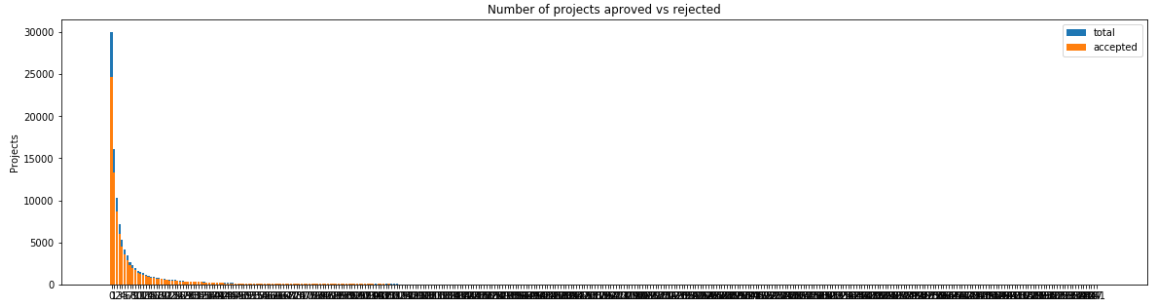
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/ba
r_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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 [59]:

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

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

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

    if top:
        temp = temp[0:top]

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

In [60]:

```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects', 'proj
ect_is_approved', False)
```




```
   teacher_number_of_previously_posted_projects  project_is_approved  tota
l  \
0                                             0                24652  3001
4
1                                             1                13329  1605
8
2                                             2                 8705  1035
0
3                                             3                 5997   711
0
4                                             4                 4452   526
6

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
     teacher_number_of_previously_posted_projects  project_is_approved  to
tal  \
242                                           242                   1
1
268                                           270                   1
1
234                                           234                   1
1
335                                           347                   1
1
373                                           451                   1
1

     Avg
242  1.0
268  1.0
234  1.0
335  1.0
373  1.0
```

In [ ]:

```
#observation:
#1.we can see there are 30014 teachers are posting the projects for the first time and
 24652 teachers get approval for their project
#2.there are very few teachers who have posted many projects previously
#3.Teachers who have posted 2 or 3 or 4 projects previously are having the approval rat
e 84%
```

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

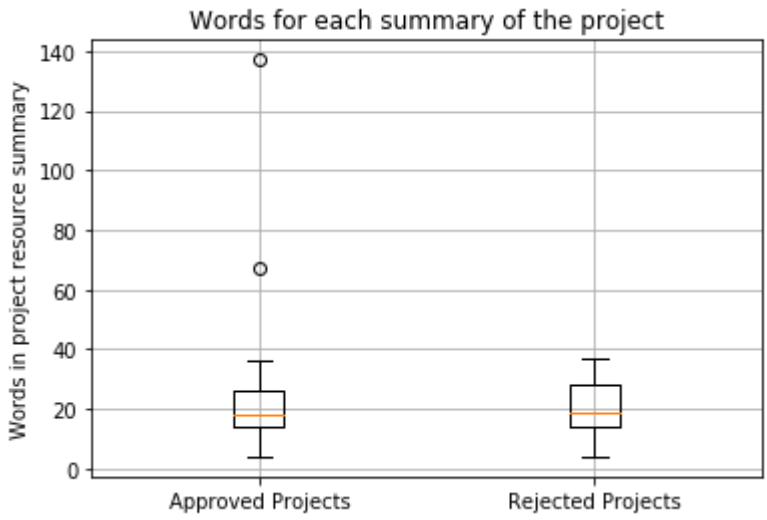
Check if the `presence of the numerical digits` in the `project_resource_summary` effects the acceptance of the project or not. If you observe that `presence of the numerical digits` is helpful in the classification, please include it for further process or you can ignore it.

In [62]:

```
approved_summary_count = project_data[project_data['project_is_approved']==1]['project_
resource_summary'].str.split().apply(len)
approved_summary_count = approved_summary_count.values

rejected_summary_count = project_data[project_data['project_is_approved']==0]['project_
resource_summary'].str.split().apply(len)
rejected_summary_count = rejected_summary_count.values
```

In [63]:

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

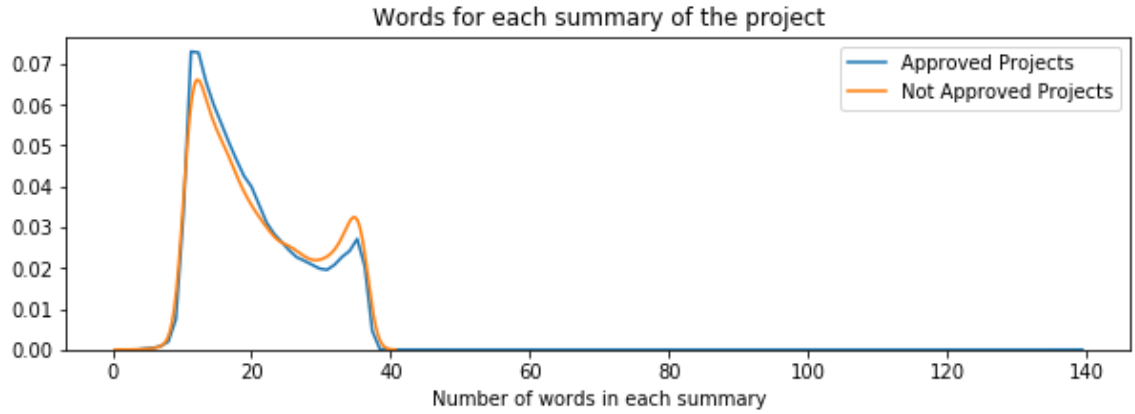


In [64]:

```
#observation:Rejected projects are having slightly more number of words in project reso
urce summary column.
```

In [65]:

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



In [ ]:

```
#observation:1.pdf of approved projects are slightly ahead than pdf of not approved pro
jects for the number of words less than or
#equal to 20 in each summary
#2.pdf of Not approved projects is slightly ahead than pdf of approved projects for the
number of words more than 20 in summary
```

In [66]:

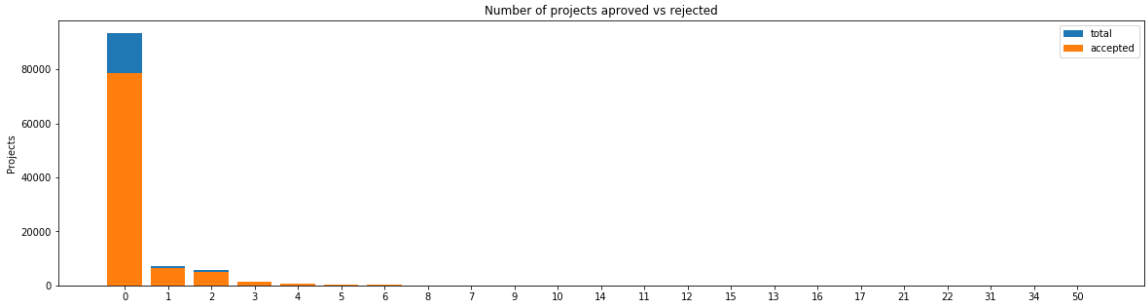
```
#univariate analysis for presence of the numerical digits` in the `project_resource_sum
mary`
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/ba
r_stacked.html

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 [67]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of non-zeros in dataframe python: https://stackoverflow.com/a/515405
21/4084039
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.ne(0).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['Percentage'] = (temp['project_is_approved']/temp['total'])*100


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

    if top:
        temp = temp[0:top]

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

In [68]:

```
#count digits in column of pandas dataframe https://stackoverflow.com/questions/4607918
5/
project_data['digits in summary']=project_data['project_resource_summary'].str.count(r
'\d')
univariate_barplots(project_data, 'digits in summary', 'project_is_approved', False)
```



	digits in summary	project_is_approved	total	Percentage
0	0	78616	93492	84.088478
1	1	6518	7229	90.164615
2	2	5107	5704	89.533661
3	3	1212	1378	87.953556
4	4	686	779	88.061617
5	5	187	216	86.574074
6	6	154	185	83.243243
8	8	73	89	82.022472
7	7	43	55	78.181818
9	9	35	37	94.594595
10	10	18	22	81.818182
14	14	19	19	100.000000
11	11	12	16	75.000000
12	12	11	11	100.000000
15	15	4	4	100.000000
13	13	2	2	100.000000
16	16	2	2	100.000000
17	17	2	2	100.000000
18	21	2	2	100.000000
19	22	0	1	0.000000
20	31	1	1	100.000000
21	34	1	1	100.000000
22	50	1	1	100.000000

In [ ]:

```
#observation:
#1.there are 93492 projects which has "no" digit in summary and there approval rate is
 84%
#2.for the number of digits in summary from 1 to 11 we can see averagely 85 % projects
 are approved
#3.there are few projects which has number of digits in summary more than 12 with 100%
 approval
#4.Digits in summary is not important factor for the approval of projects so i  am not
 going to consider it in further process
```

1.3 Text preprocessing

I am considering sample of 20k data points for further process.

1.3.1 Essay Text

In [70]:

```
project_data=project_data.sample(20000)
project_data.head(2)
```

Out[70]:

	Unnamed: 0	id	teacher_id	teacher_prefix	scho
47176	59089	p159010	b77f624e56532929ac021ab48e1db85f	Mrs.	IL
94387	11238	p239494	132c8fd6bd9af1ab5ccf69092c208744	Teacher	MI

2 rows × 21 columns

In [71]:

```
# 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)
```

1st Grade ROCKS!\r\nWe have amazing students, supportive administrators, d
edicated teachers and devoted support staff! The school environment is bot
h warm and inviting for staff, students, parents, and visitors. I couldn't
ask for a better place to work! \r\nI have 25 students in my class. My st
udents' eyes grow with wonder as they learn something new. It is an honor
to be able to grow and learn beside each of my students daily. My ultimat
e goal is to provide my students with an authentic learning environment, w
here they are excited to explore, grow, learn, and utilize technology dail
y.We are asking for two new Chromebooks to provide students with the oppor
tunity to practice skills in reading, writing, and math. The headphones,
mice, powerstrip, and licenses are all needed to get the most use out of t
hese Chromebooks. \r\n\r\nGiving students access to technology is an inves
tment in their future. \r\nThe use of this technology will allow students
to work independently at their own level and pace. Reading books, math f
acts, and practicing keyboarding are just a few examples of the many ways
students will use this technology. Thank you for your consideration to hel
p make the best use of technology in our classroom.nannan
==========
These 3-5 year olds know how to play hard and work hard! Most of them have
some sort of learning difference and need specially designed instruction.
We are located in the most diverse district in Washington and we have at l
east 75% of our children qualify for free or reduced lunches. In my class
I have 10 different cultures represented. Unfortunately, these students do
n't have access to technology or STEM activities in our school because we
do not have the funding to provide the materials or technology.I want to e
ngage my preschoolers in fun, engaging activities that make them think and
solve problems. I can just see them working together to create ramps for t
he balls to travel down or building a house that looks like where they liv
e. They can do mighty things, but they need just the right materials and v
isuals to meet their needs. \r\n\r\nThese materials will be shared through
out our building so that even more children will have the opportunity to i
ncrease their problem solving skills. It is important to increase our stud
ents skills in science, engineering, math and I am convinced that these ma
terials would help us prepare our students for a bright and successful fut
ure!nannan
==========
I work with students at both the elementary and junior high level in McLea
n County Unit District #5. The elementary school I serve is a Title One s
chool with approximately 65% of the students that are low income and we ha
ve a 21% mobility rate. It is unique because it also houses an English La
nguage Learner program for the district, creating a very diverse populatio
n for the school. The junior high that I serve has a lower percentage of
low income students at approximately 21%, but houses the junior high age e
motional disability program for the district. \r\n\r\nThese students don't
just come to school for academic instruction, they need love, encouragemen
t, positive supports, and someone to be their advocate. These students WA
NT to learn, but sometimes need their basic needs met before that can happ
en. We are working to meet those needs and to push them to be the student
s we know they can be!The students that I work with have difficulties regu
lating emotions and engaging in the classroom due to a variety of reasons.
They don't always respond to typical interventions that are implemented an
d we need to think outside of the box to come up with ideas to help suppor
t their behavioral progress at school.\r\n\r\nThe MotivAider is a device t
hat can assist students in making behavioral changes. The device can be w
orn by the student or teacher and it is set at different intervals to vibr
ate accordingly. The vibrations are intended to help students with self-m
onitoring of behaviors as well as with the teacher in reinforcing certain
behaviors. The MotivAider can also be used to collect behavioral data, wh
ich can be used to develop behavior intervention plans. \r\n\r\nStudents
who have difficulty maintaining appropriate behaviors can be challenging t
o work with. It is important that we continue to look for interventions a

nd supports that will help improve these behaviors so that these students can get the most out of their educational experience. We can't give up on these students!\r\n\r\nnannan

==

My students help me make music all over the building! I lost my classroom due to an increase in class sizes, so I teach music from a cart. This does n't stop my students from singing, playing, moving, and grooving to the mu sic every day.\r\n\r\nI teach at a Title I school where more than 75% of m y students are receiving free or reduced-price lunch and come from single family homes. We are a low performing school according to the State and a re listed in Priority Status.\r\n\r\nMy students enjoy music and will love the chance at being able to do something new and exciting during their mus ical center rotations.My students love rhythm! I always try to find a way for them to express themselves through music. This is an ideal outlet for my students who may not have the means to purchase their own drum. They c an see that they don't need an expensive instrument to get themselves star ted moving to the beat! My plan is to not only teach rhythm with drummin g, but to incorporate rhythm with dance and movement. Students in a high- risk school have unneeded stress on them at home and school daily. Music is an essential part of every person's life and this moving and drumming c an only alleviate more stress!\r\nnannan

==

When you enter our 21st century kindergarten classroom, not only will you see Play-doh, building blocks, and paints, you will also see tablets, inte ractive whiteboards, and laptops! We are a busy group of learners, eager to grow our mindsets and challenge ourselves to do better and be better ev eryday!\r\n\r\nOur class is a melting pot of cultures!\r\nMany of our stud ents have just moved to our country and are just learning English, while o thers have lived here all their lives, but are experiencing school for the first time! Our school is a Title I school with over twenty-seven language s spoken. We are a community committed to providing the best education po ssible to all of our amazing students.Having these materials will enable m y students to express their learning through art. So many of my students are not at the point in their development where they are able to write usi ng words and instead express themselves more confidently through their dra wings. Having an easel available to them will offer up another way for th em to express their thinking in language arts, math, science, social studi es, etc. For example, my students can use the easel during literacy cente rs to illustrate the setting of a story, describe a character, or retell i mportant parts of a story. In math, they can paint to decompose and compo se numbers, create art using different shapes, solve math story problems. In science, my students can use the easel to sketch what they observe happ ening when we study various animal lifcycles. More importantly, this give s access to my special needs students to express themselves in a way that is more engaging and often easier than verbally or in traditional written form. The possibilities are endless with these materials!nannan

==

In [72]:

```
# 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 [73]:

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

```
My students help me make music all over the building! I lost my classroom
due to an increase in class sizes, so I teach music from a cart. This does
not stop my students from singing, playing, moving, and grooving to the mu
sic every day.\r\n\r\nI teach at a Title I school where more than 75% of m
y students are receiving free or reduced-price lunch and come from single
family homes.  We are a low performing school according to the State and a
re listed in Priority Status.\r\n\r\nMy students enjoy music and will love
the chance at being able to do something new and exciting during their mus
ical center rotations.My students love rhythm!  I always try to find a way
for them to express themselves through music.  This is an ideal outlet for
my students who may not have the means to purchase their own drum.  They c
an see that they do not need an expensive instrument to get themselves sta
rted moving to the beat!  My plan is to not only teach rhythm with drummin
g, but to incorporate rhythm with dance and movement.  Students in a high-
risk school have unneeded stress on them at home and school daily.  Music
is an essential part of every person is life and this moving and drumming
can only alleviate more stress!\r\nnannan
==================================================
```

In [74]:

```
# \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)
```

```
My students help me make music all over the building! I lost my classroom
due to an increase in class sizes, so I teach music from a cart. This does
not stop my students from singing, playing, moving, and grooving to the mu
sic every day.    I teach at a Title I school where more than 75% of my st
udents are receiving free or reduced-price lunch and come from single fami
ly homes.  We are a low performing school according to the State and are l
isted in Priority Status.    My students enjoy music and will love the cha
nce at being able to do something new and exciting during their musical ce
nter rotations.My students love rhythm!  I always try to find a way for th
em to express themselves through music.  This is an ideal outlet for my st
udents who may not have the means to purchase their own drum.  They can se
e that they do not need an expensive instrument to get themselves started
moving to the beat!  My plan is to not only teach rhythm with drumming, bu
t to incorporate rhythm with dance and movement.  Students in a high-risk
school have unneeded stress on them at home and school daily.  Music is an
essential part of every person is life and this moving and drumming can on
ly alleviate more stress!  nannan
```

In [75]:

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

```
My students help me make music all over the building I lost my classroom d
ue to an increase in class sizes so I teach music from a cart This does no
t stop my students from singing playing moving and grooving to the music e
very day I teach at a Title I school where more than 75 of my students are
receiving free or reduced price lunch and come from single family homes We
are a low performing school according to the State and are listed in Prior
ity Status My students enjoy music and will love the chance at being able
to do something new and exciting during their musical center rotations My
students love rhythm I always try to find a way for them to express themse
lves through music This is an ideal outlet for my students who may not hav
e the means to purchase their own drum They can see that they do not need
an expensive instrument to get themselves started moving to the beat My pl
an is to not only teach rhythm with drumming but to incorporate rhythm wit
h dance and movement Students in a high risk school have unneeded stress o
n them at home and school daily Music is an essential part of every person
is life and this moving and drumming can only alleviate more stress nannan
```

In [76]:

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

In [77]:

```
# 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%|██████████████████████████████████████████████████████████████████████████████
█████| 20000/20000 [00:13<00:00, 1457.81it/s]
```

In [78]:

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

Out[78]:

```
'my students help make music building i lost classroom due increase class
sizes i teach music cart this not stop students singing playing moving gro
oving music every day i teach title i school 75 students receiving free re
duced price lunch come single family homes we low performing school accord
ing state listed priority status my students enjoy music love chance able
something new exciting musical center rotations my students love rhythm i
always try find way express music this ideal outlet students may not means
purchase drum they see not need expensive instrument get started moving be
at my plan not teach rhythm drumming incorporate rhythm dance movement stu
dents high risk school unneeded stress home school daily music essential p
art every person life moving drumming alleviate stress nannan'
```

1.3.2 Project title Text

In [80]:

```
project_data.head(2)
```

Out[80]:

	Unnamed: 0	id	teacher_id	teacher_prefix	scho
47176	59089	p159010	b77f624e56532929ac021ab48e1db85f	Mrs.	IL
94387	11238	p239494	132c8fd6bd9af1ab5ccf69092c208744	Teacher	MI

2 rows × 21 columns

In [81]:

```
# printing some random title.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[50])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
print(project_data['project_title'].values[500])
print("="*50)
print(project_data['project_title'].values[2000])
print("="*50)
print(project_data['project_title'].values[5000])
print("="*50)
print(project_data['project_title'].values[7000])
print("="*50)
print(project_data['project_title'].values[10000])
print("="*50)
```

```
Tiny Titans Need Tech Too!
==================================================
Mobile Teaching!
==================================================
Our Very Own Chromebooks!
==================================================
Exploring Science With Technology
==================================================
Filling a Bucket One Beat at a Time
==================================================
Feeling Squeaky Clean
==================================================
MagnaTiles for TK
==================================================
Different Perspectives
==================================================
```

In [82]:

```
# 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 [83]:

```
sent = decontracted(project_data['project_title'].values[2000])
print(sent)
print("="*50)
sent1 = decontracted(project_data['project_title'].values[3000])
print(sent1)
print("="*50)
sent2 = decontracted(project_data['project_title'].values[5000])
print(sent2)
print("="*50)
```

```
Filling a Bucket One Beat at a Time
==================================================
Expand our knowledge with books!
==================================================
Feeling Squeaky Clean
==================================================
```

In [84]:

```
# \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)
```

```
Filling a Bucket One Beat at a Time
```

In [85]:

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

```
Filling a Bucket One Beat at a Time
```

In [86]:

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

In [87]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].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_title.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████
████| 20000/20000 [00:00<00:00, 30907.68it/s]
```

In [88]:

```
# after preprocesing
preprocessed_title[1000]
```

Out[88]:

```
'motivaider a behavioral support tool challenging students'
```

1. 4 Preparing data for models

In [89]:

```
project_data.columns
```

Out[89]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_ti
tle',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quant
ity',
       '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 [90]:

```
# 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', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (20000, 9)
```

In [91]:

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


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutrition
Education', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'A
ppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Lit
eracy']
Shape of matrix after one hot encodig  (20000, 30)
```

In [94]:

```
#Vectorizing Categorical data:State

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


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

# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer1 = CountVectorizer(vocabulary=list(sorted_state_dict.keys()), lowercase=Fals
e, binary=True)
vectorizer1.fit(project_data['school_state'].values)
print(vectorizer1.get_feature_names())


state_one_hot = vectorizer1.transform(project_data['school_state'].values)

print("Shape of matrix after one hot encodig ",state_one_hot.shape)
```

```
['WY', 'VT', 'ND', 'MT', 'NE', 'AK', 'NH', 'RI', 'SD', 'DE', 'DC', 'WV',
'KS', 'HI', 'ME', 'NM', 'IA', 'ID', 'CO', 'AR', 'MN', 'OR', 'MS', 'NV', 'K
Y', 'MD', 'CT', 'TN', 'UT', 'WI', 'AL', 'VA', 'AZ', 'NJ', 'OK', 'LA', 'W
A', 'MA', 'OH', 'MO', 'IN', 'PA', 'MI', 'GA', 'SC', 'IL', 'NC', 'FL', 'T
X', 'NY', 'CA']
Shape of matrix after one hot encodig  (20000, 51)
```

In [95]:

```
#Vectorizing Categorical data:teacher_prefix

project_data1=project_data

#to drop a row having nan https://stackoverflow.com/questions/13413590
project_data1=project_data1.dropna(subset=['teacher_prefix'])
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
my_counter.update(project_data1['teacher_prefix'])

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

#we use count vectorizer to convert the values into one hot encoded features

vectorizer1 = CountVectorizer(vocabulary=list(sorted_teacher_dict.keys()), lowercase=Fa
lse, binary=True)
vectorizer1.fit(project_data1['teacher_prefix'].values)
print(vectorizer1.get_feature_names())


prefix_one_hot = vectorizer1.transform(project_data1['teacher_prefix'].values)

print("Shape of matrix after one hot encodig ",prefix_one_hot.shape)
```

```
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encodig  (19999, 5)
```

In [96]:

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


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

#https://thispointer.com/different-ways-to-remove-a-key-from-dictionary-in-python/
if "Grades" in sorted_grade_dict:
    del sorted_grade_dict["Grades"]


#Vectorizing Categorical data:project_grade_category

# we use count vectorizer to convert the values into one hot encoded features
vectorizer3 = CountVectorizer(vocabulary=list(sorted_grade_dict.keys()), lowercase=Fals
e, binary=True)
vectorizer3.fit(project_data['project_grade_category'].values)
print(vectorizer3.get_feature_names())


grade_one_hot = vectorizer3.transform(project_data['project_grade_category'].values)
print("Shape of matrix after one hot encodig ",grade_one_hot.shape)
```

```
['9-12', '6-8', '3-5', 'PreK-2']
Shape of matrix after one hot encodig  (20000, 4)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [97]:

```
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects).
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  (20000, 8463)
```

1.4.2.2 Bag of Words on `project_title`

In [100]:

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

```
Shape of matrix after one hot encodig  (20000, 1134)
```

1.4.2.3 TFIDF vectorizer

In [101]:

```
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  (20000, 8463)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [103]:

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

```
Shape of matrix after one hot encodig  (20000, 1134)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [104]:

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

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

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

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

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

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

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our 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.com/how-to-use-p
ickle-to-save-and-load-variables-in-python/

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


'''
```

Out[104]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Mode
l")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    f
or line in tqdm(f):\n        splitLine = line.split()\n        word = spli
tLine[0]\n        embedding = np.array([float(val) for val in splitLine
[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," w
ords loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.
txt\')\n\n# ============================\nOutput:\n    \nLoading Glove Mod
el\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ====
========================\n\nwords = []\nfor i in preproced_texts:\n    wor
ds.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend
(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords
= set(words)\nprint("the unique words in the coupus", len(words))\n\ninter
_words = set(model.keys()).intersection(words)\nprint("The number of words
that are present in both glove vectors and our coupus",       len(inter_wo
rds),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpu
s = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in word
s_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length",
len(words_courpus))\n\n\n# stronging variables into pickle files python: h
ttp://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-
python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n
pickle.dump(words_courpus, f)\n\n\n'
```

In [105]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-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 [106]:

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

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

```
100%|██████████████████████████████████████████████████████████████████
█████| 20000/20000 [00:07<00:00, 2761.50it/s]

20000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [108]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-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 [109]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(preprocessed_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title.append(vector)

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

```
100%|████████████████████████████████████████████████████████████████████
████| 20000/20000 [00:00<00:00, 44741.89it/s]

20000
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [110]:

```
# 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 [111]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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%|████████████████████████████████████████████████████████████████
██████| 20000/20000 [00:48<00:00, 408.77it/s]

20000
300
```

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

In [113]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_title)
# 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_title = set(tfidf_model.get_feature_names())
```

In [114]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this
 list
for sentence in tqdm(preprocessed_title): # 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_title):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_title.append(vector)

print(len(tfidf_w2v_vectors_title))
print(len(tfidf_w2v_vectors_title[0]))
```

```
100%|████████████████████████████████████████████████████████████████
████| 20000/20000 [00:01<00:00, 16199.06it/s]

20000
300
```

1.4.3 Vectorizing Numerical features

In [115]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.pr
eprocessing.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=[725.05 213.03 329.   ...
 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 the mean and sta
ndard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_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 : 298.6617355, Standard deviation : 380.77479530992207
```

In [116]:

```
price_standardized
```

Out[116]:

```
array([[-0.20681972],
       [ 0.18317458],
       [-0.31565045],
       ...,
       [-0.15151143],
       [ 0.21353374],
       [ 0.5190424 ]])
```

In [117]:

```
# Vectorizing Numerical features:teacher_number_of_previously_posted_projects

# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


teacherNo_scalar = StandardScaler()
teacherNo_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
print(f"Mean : {teacherNo_scalar.mean_[0]}, Standard deviation : {np.sqrt(teacherNo_scalar.var_[0])}")

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

```
C:\Users\shind\Anaconda3\lib\site-packages\sklearn\utils\validation.py:47
5: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.


Mean : 11.1071, Standard deviation : 27.4375623842571

C:\Users\shind\Anaconda3\lib\site-packages\sklearn\utils\validation.py:47
5: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

In [118]:

```
teacherNo_standardized
```

Out[118]:

```
array([[-0.40481366],
       [-0.40481366],
       [-0.2954745 ],
       ...,
       [ 1.70907675],
       [-0.33192089],
       [-0.40481366]])
```

1.4.4 Merging all the above features

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

In [119]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(20000, 9)
(20000, 30)
(20000, 8463)
(20000, 1)
```

In [120]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[120]:

```
(20000, 8503)
```

In [121]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(state_one_hot.shape)
print(prefix_one_hot.shape)
print(grade_one_hot.shape)
print(title_bow.shape)
print(price_standardized.shape)
print(teacherNo_standardized.shape)
```

```
(20000, 9)
(20000, 30)
(20000, 51)
(19999, 5)
(20000, 4)
(20000, 1134)
(20000, 1)
(20000, 1)
```

prefix_one_hot has not merged due to it's dimension problem.Since there is 'nan' for teacher_prefix in one row,so we have dropped that row from data that's why i has 19999 row instead of 20k.

In [123]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
# 1.Merging all the above features i.e.categorical, numerical features + project_title (BOW)

X_bow = hstack((categories_one_hot, sub_categories_one_hot, state_one_hot, grade_one_hot, title_bow, price_standardized, teacherNo_standardized))
X_bow.shape
```

Out[123]:

```
(20000, 1230)
```

In [124]:

```
# 2.Merging all the above features i.e.categorical, numerical features + project_title (TFIDF)
X_tfidf= hstack((categories_one_hot, sub_categories_one_hot, state_one_hot, grade_one_hot, title_tfidf, price_standardized, teacherNo_standardized))
X_tfidf.shape
```

Out[124]:

```
(20000, 1230)
```

In [125]:

```
# 3.Merging all the above features i.e.categorical, numerical features + project_title (AVG W2V)
X_avg_w2v= hstack((categories_one_hot, sub_categories_one_hot, state_one_hot, grade_one_hot, avg_w2v_vectors_title, price_standardized, teacherNo_standardized))
X_avg_w2v.shape
```

Out[125]:

```
(20000, 396)
```

In [126]:

```
# 4.Merging all the above features i.e.categorical, numerical features + project_title (TFIDF WEIGHTED W2V)
X_tfidf_w2v= hstack((categories_one_hot, sub_categories_one_hot, state_one_hot, grade_one_hot, tfidf_w2v_vectors_title, price_standardized, teacherNo_standardized))
X_tfidf_w2v.shape
```

Out[126]:

```
(20000, 396)
```

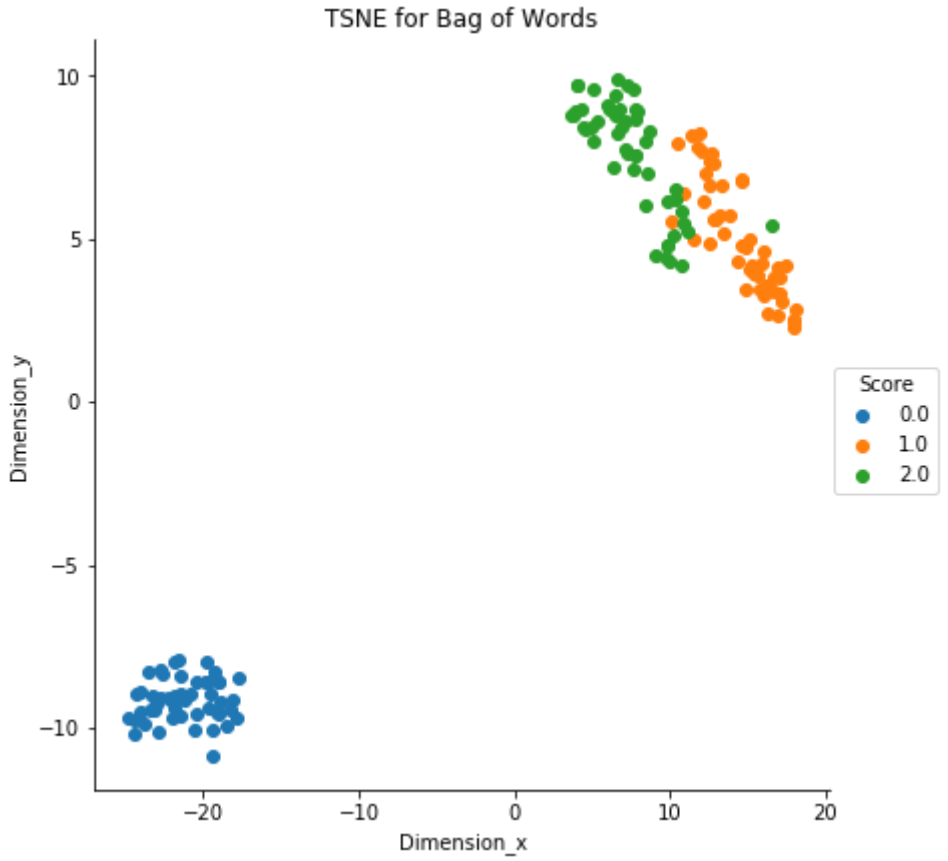
Assignment 2: Apply TSNE

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 [127]:

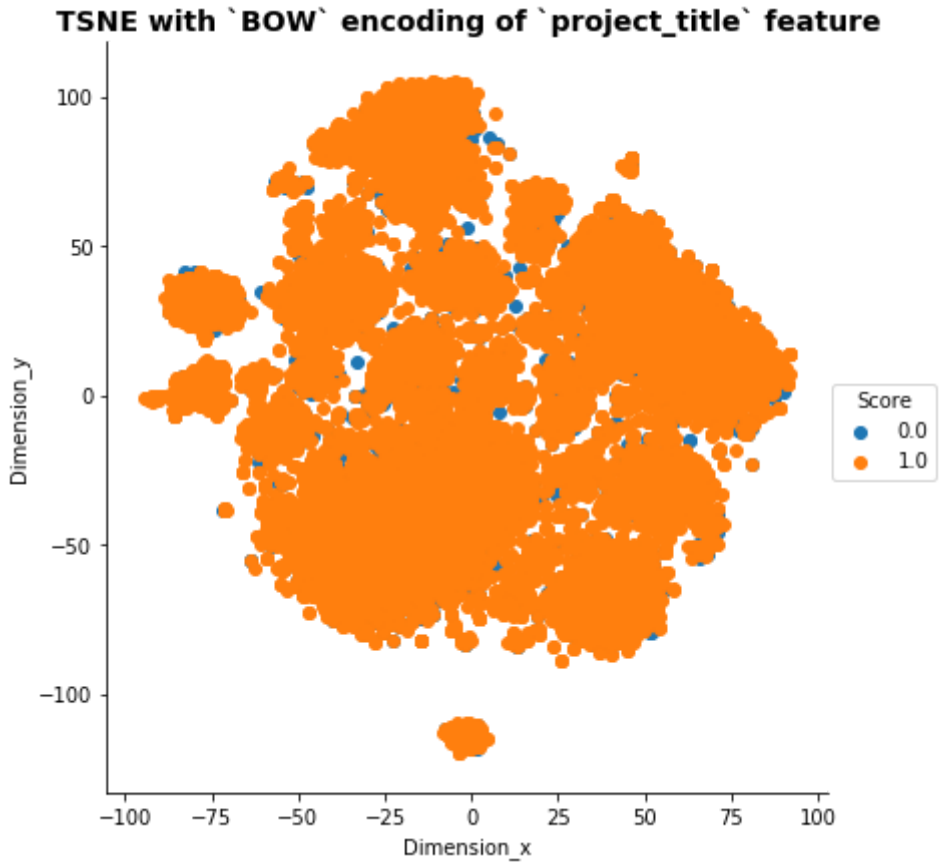
```
# this is the example code for TSNE
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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.toa
rray()) , .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'
])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Score", size=6).map(plt.scatter, 'Dimension_x', 'Dimens
ion_y').add_legend()
plt.title("TSNE for Bag of Words")
plt.show()
```



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

In [128]:

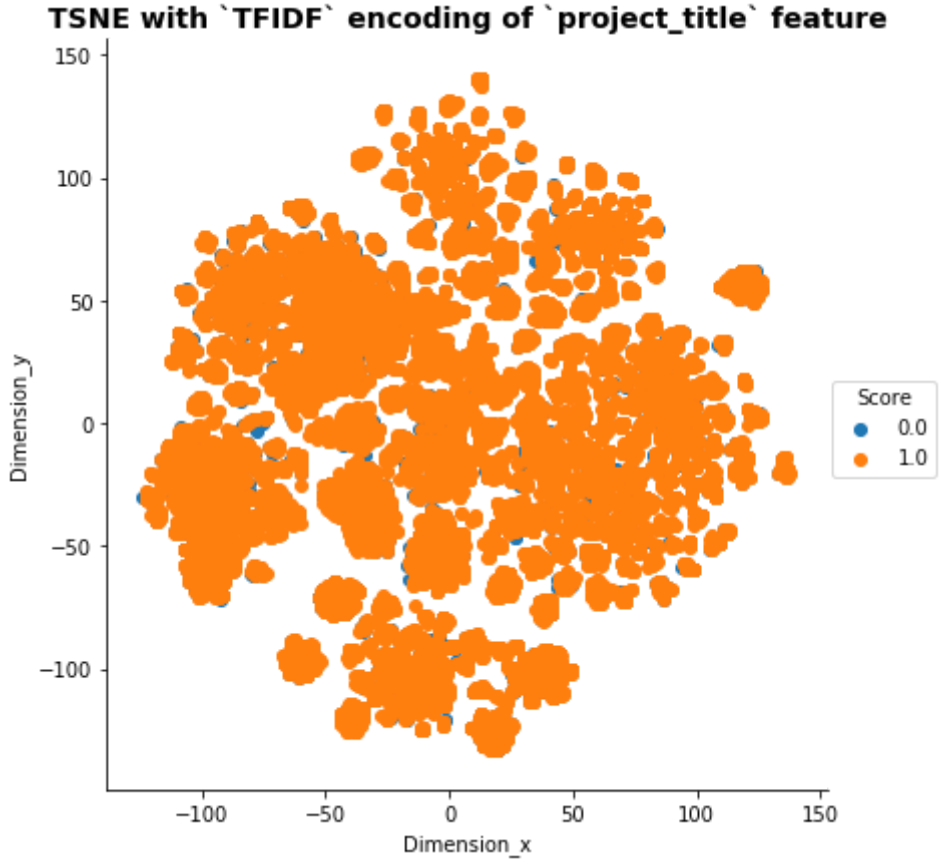
```
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

x = X_bow
y = project_data['project_is_approved']
y = np.array(y)
y = np.reshape(y,(-1,1))
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200, n_iter=2000)

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

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

for_tsne = np.hstack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne,columns=['Dimension_x','Dimension_y','Score'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Score", size=6).map(plt.scatter, 'Dimension_x', 'Dimens
ion_y').add_legend()
plt.title('TSNE with `BOW` encoding of `project_title` feature', weight='bold').set_fon
tsize('14')
plt.show()
```



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

In [130]:

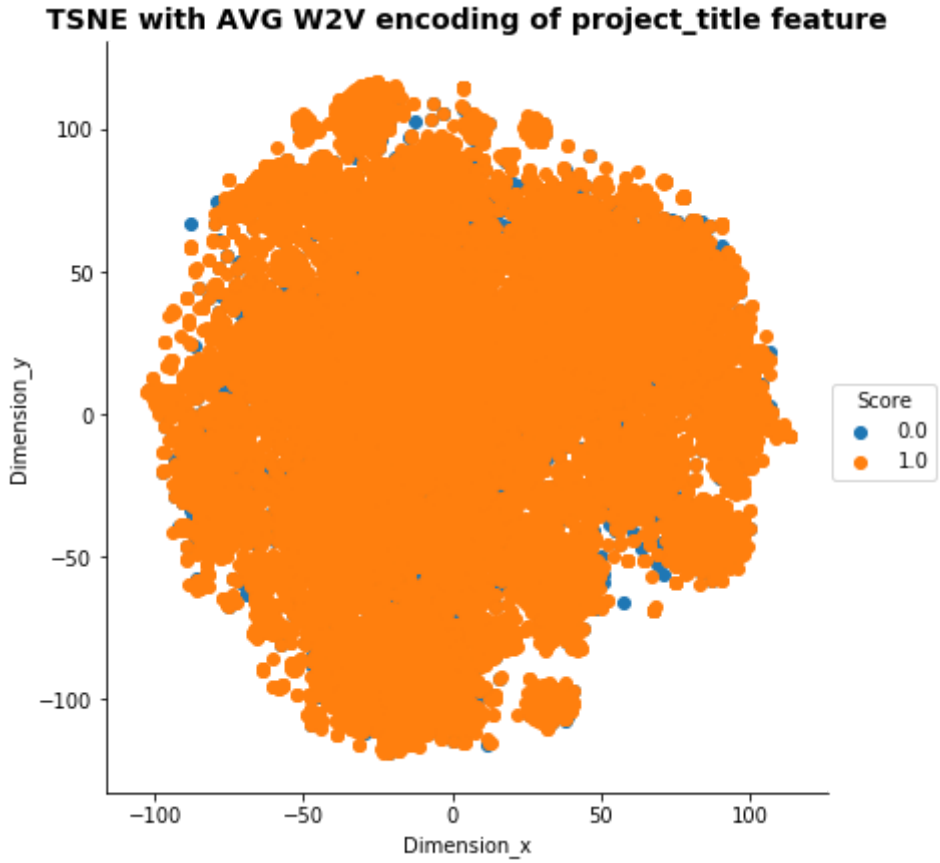
```
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

x = X_tfidf
y = project_data['project_is_approved']
y = np.array(y)
y = np.reshape(y,(-1,1))
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200, n_iter=2000)

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

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

for_tsne = np.hstack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne,columns=['Dimension_x','Dimension_y','Score'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Score", size=6).map(plt.scatter, 'Dimension_x', 'Dimens
ion_y').add_legend()
plt.title('TSNE with `TFIDF` encoding of `project_title` feature', weight='bold').set_f
ontsize('14')
plt.show()
```



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

In [134]:

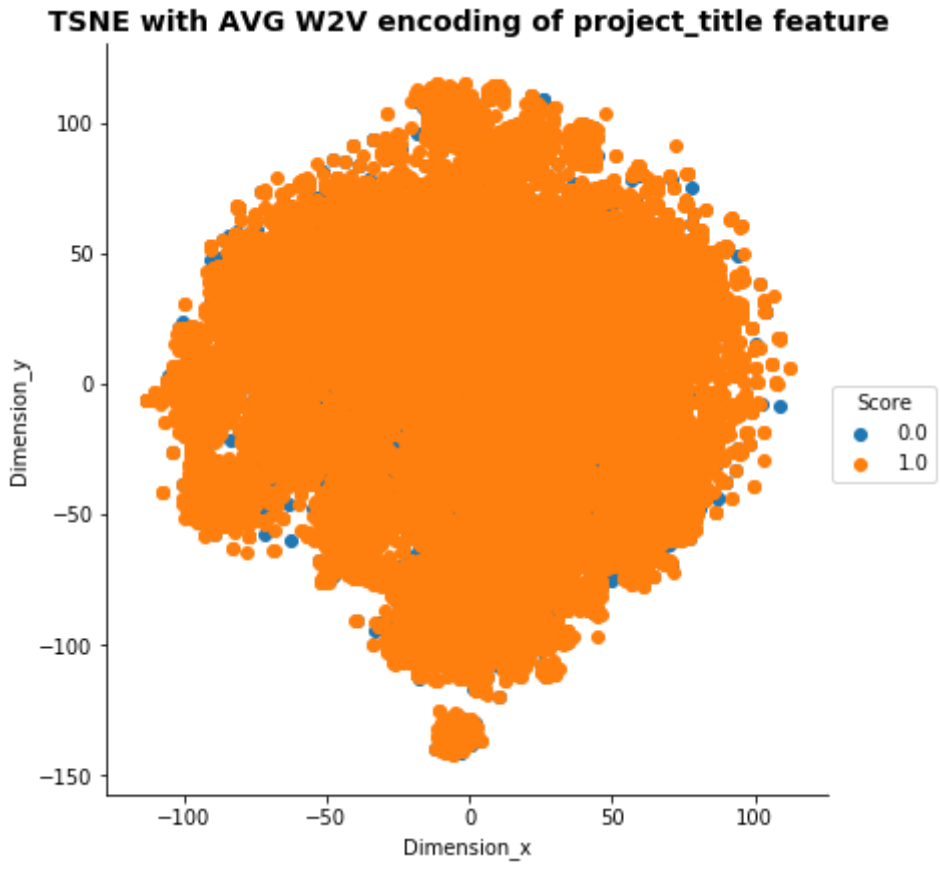
```
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

x = X_avg_w2v
y = project_data['project_is_approved']
y = np.array(y)
y = np.reshape(y,(-1,1))
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200, n_iter=2000)

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

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

for_tsne = np.hstack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne,columns=['Dimension_x','Dimension_y','Score'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Score", size=6).map(plt.scatter, 'Dimension_x', 'Dimens
ion_y').add_legend()
plt.title('TSNE with AVG W2V encoding of project_title feature', weight='bold').set_fon
tsize('14')
plt.show()
```



2.4 TSNE with X_tfidf_w2v encoding of project_title feature

In [135]:

```
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

x = X_avg_w2v
y = project_data['project_is_approved']
y = np.array(y)
y = np.reshape(y,(-1,1))
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200, n_iter=2000)

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

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

for_tsne = np.hstack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne,columns=['Dimension_x','Dimension_y','Score'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Score", size=6).map(plt.scatter, 'Dimension_x', 'Dimens
ion_y').add_legend()
plt.title('TSNE with AVG W2V encoding of project_title feature', weight='bold').set_fon
tsize('14')
plt.show()
```



In [136]:

```
#5.Merging all the above features i.e.categorical, numerical features + project_title(B
OW) + project_title(TFIDF) + project_title(AVG W2V) + project_title(TFIDF WEIGHTED W2V)

X_final= hstack((categories_one_hot, sub_categories_one_hot, state_one_hot, grade_one_h
ot, title_bow, title_tfidf, avg_w2v_vectors_title, tfidf_w2v_vectors_title, price_stand
ardized, teacherNo_standardized))
X_final.shape
```

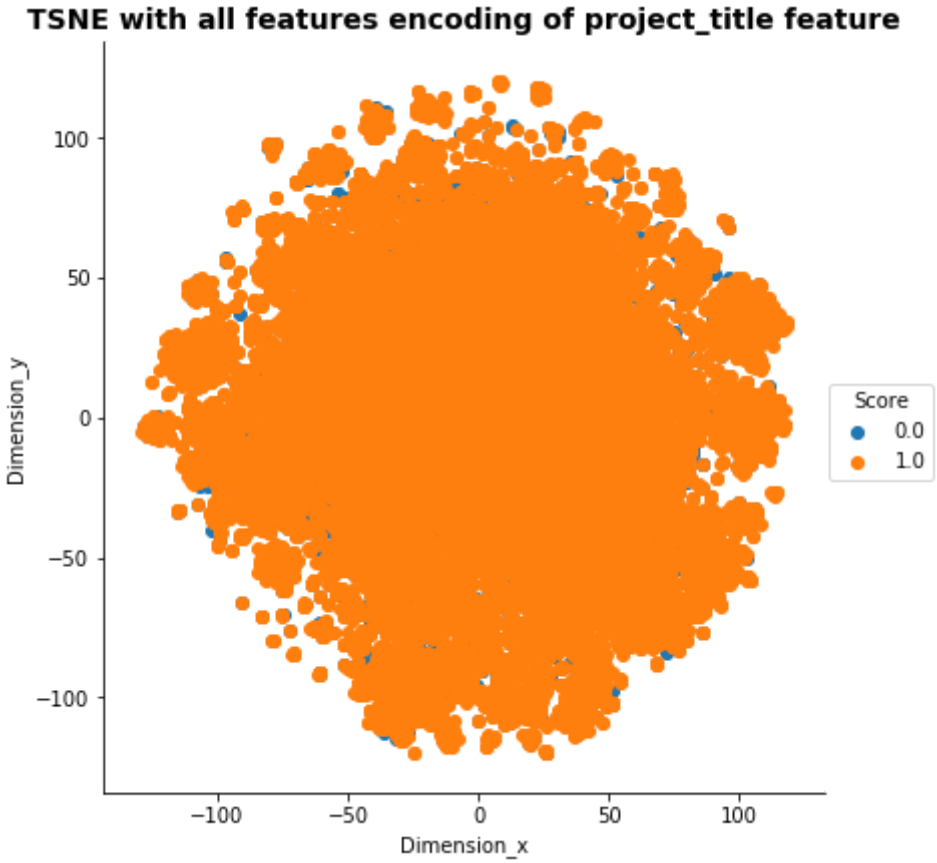
Out[136]:

```
(20000, 2964)
```

TSNE with all the above features i.e.categorical, numerical features + project_title(BOW) + project_title(TFIDF) + project_title(AVG W2V) + project_title(TFIDF WEIGHTED W2V)

In [137]:

```
#TSNE with 'all' features encoding of project_title feature

import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

x = X_final
y = project_data['project_is_approved']
y = np.array(y)
y = np.reshape(y,(-1,1))
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200, n_iter=2000)

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

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

for_tsne = np.hstack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne,columns=['Dimension_x','Dimension_y','Score'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Score", size=6).map(plt.scatter, 'Dimension_x', 'Dimens
ion_y').add_legend()
plt.title('TSNE with all features encoding of project_title feature', weight='bold').se
t_fontsize('14')
plt.show()
```



SUMMARY:

tsne reduces very high dimensional data points into 2-Dimensional data points.using seabron tsne can be plotted very beautifully

In tsne plot we can observe there are high number of projects which are approved.

In tsne plot we can observe there is overlapping in class label of datapoints.

In project summary column digits in the summary are not playing very important role for the acceptance of projects.Most of the projects have no digit in summary.

Box-cox plot and pdfs of costs per project are not providing much information about the acceptance or rejection of projects. In this case percentile gives clear information that less costly projects are approved while more costly projects are rejected

In the project subject categories-warmth and hunger,most of the projects are approved