DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

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

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

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

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

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

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

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates

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 [117]:

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

In [118]:

```
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_pr
efix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_
3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approve
d']
```

In [119]:

```
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[119]:

	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 [120]:

```
# 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-glr-gallery-pie-and-polar-charts-pie-and-donut-labels-py


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

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

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

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

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

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

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

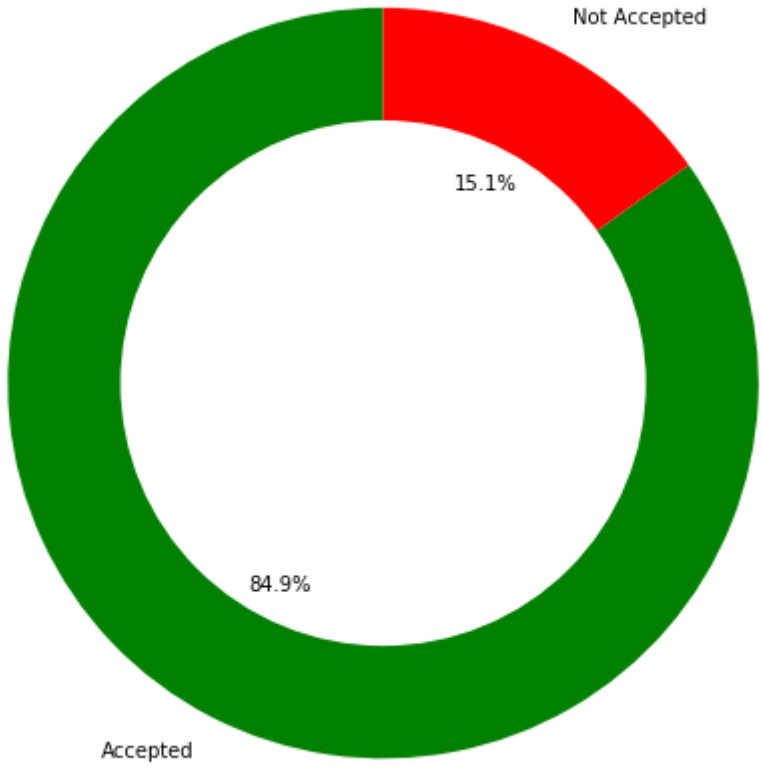
```
Number of projects thar are approved for funding  92706 , ( 84.85830
404217927 %)
Number of projects thar are not approved for funding  16542 , ( 15.1
41695957820739 %)
```

New code added for visualization

- Here I have put in a simplified code snippet which represents a pie chart in donut form and plots the accpeted and not accepted projects along with their labelled percentages.
- This is because I found the given code too cumbersome to understand, the following does a similar job and is readable in my opinion.
- References for the code : **https://medium.com/@kvnamipara/a-better-visualisation-of-pie-charts-by-matplotlib-935b7667d77f (https://medium.com/@kvnamipara/a-better-visualisation-of-pie-charts-by-matplotlib-935b7667d77f)**

In [121]:

```
#Pie chart
labels = ['Accepted','Not Accepted']
sizes = [y_value_counts[1],y_value_counts[0]]
#colors
colors = ['green','red']

fig1, ax1 = plt.subplots(figsize=(6, 6))
ax1.pie(sizes, colors = colors, labels=labels, autopct='%1.1f%%', startangle=90)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
plt.tight_layout()
# Let us increase font size. I find the default to be too small to read :)
plt.rcParams.update({'font.size': 10})
plt.show()
```



1.2.1 Univariate Analysis: School State

In [122]:

```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/408
4039

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

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

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,22
0)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,3
9,143)']]

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

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

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

Out[122]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.c
om/a/9620\n\nscl = [[0.0, \'rgb(242,240,247)\'],[0.2, \'rgb(218,218,
235)\'],[0.4, \'rgb(188,189,220)\'], [0.6, \'rgb(158,154,
200)\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\nda
ta = [ dict(\n type=\'choropleth\',\n colorscale = sc
l,\n autocolorscale = False,\n locations = temp[\'stat
e_code\'],\n z = temp[\'num_proposals\'].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 pro")\n )
]\n\nlayout = dict(\n title = \'Project Proposals % of Accept
ance 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.iplo
t(fig, filename=\'us-map-heat-map\')\n'
```

In [123]:

```
# A peek inside temp
temp.head()
```

Out[123]:

	state_code	num_proposals
0	AK	0.840580
1	AL	0.854711
2	AR	0.831268
3	AZ	0.838379
4	CA	0.858136

In [124]:

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

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

Obervations:

- The state with maximum number of project approvals is **Delaware** (89.79%) which is closely followed by **North Dakota** (88.81%).
- The state with minimum number of project approval is **Vermont** (80%). **District of Columbia,** is only a fraction ahed of Vermont in terms of project approvals with a percentage of 80.23%.

In [125]:

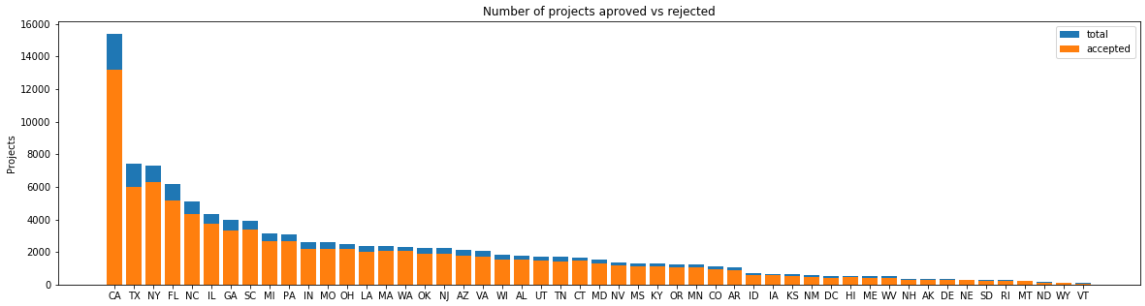
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_mar
kers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

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

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

In [126]:

```
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 [127]:

```
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

==========

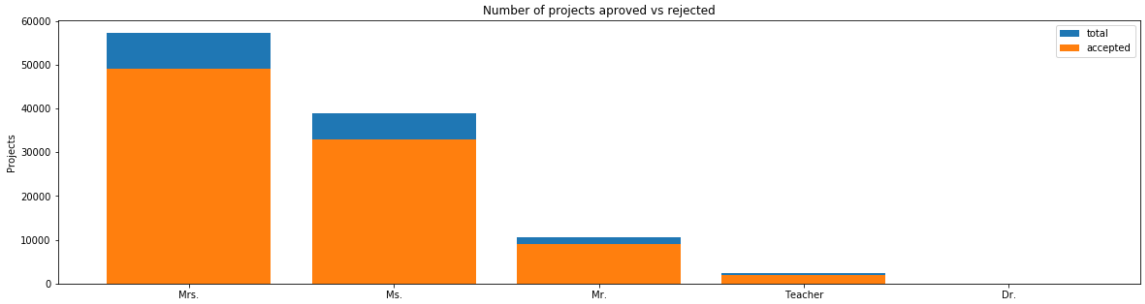
	school_state	project_is_approved	total	Avg
39	RI	243	285	0.852632
26	MT	200	245	0.816327
28	ND	127	143	0.888112
50	WY	82	98	0.836735
46	VT	64	80	0.800000

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

1.2.2 Univariate Analysis: teacher_prefix

In [128]:

```
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
```

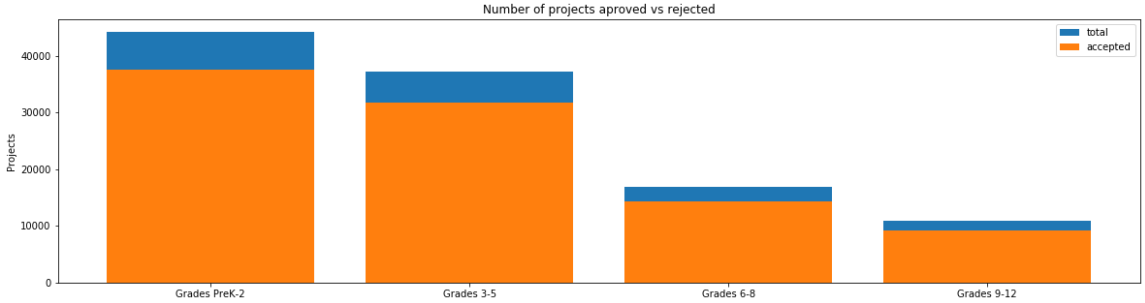
Observation:

- Teachers with maximum numbers of project approved use prefix as **Mrs**. They have average of **85.5%** approvals.
- Teachers with prefix **Ms.** get about **84.4%** of the projects approved.
- Teacher with prefix **Dr** have average approval of only **69.23%**.

1.2.3 Univariate Analysis: project_grade_category

In [129]:

```
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
```

Obervation:

- Maximum number of project submissions have been done for **grade PreK-2 (44225)**. The projects for these grades have an average approval of **84.87% (37536 approved)**.
- Minimum number of project submissions have been done for **grade 9-12**. They have an average approval of **83.76%**.
- **Grades 3-5** and **6-8** have average project approval of **85.43% and 84.25%** respectively.

1.2.4 Univariate Analysis: project_subject_categories

In [130]:

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

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

In [131]:

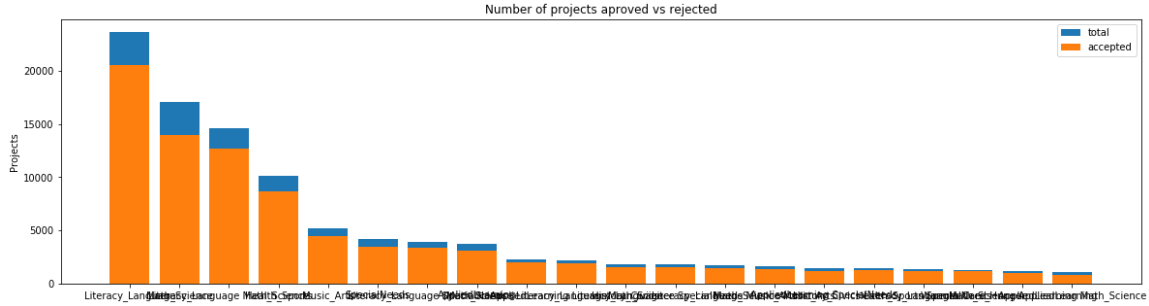
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[131]:

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

In [132]:

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




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

Observation:

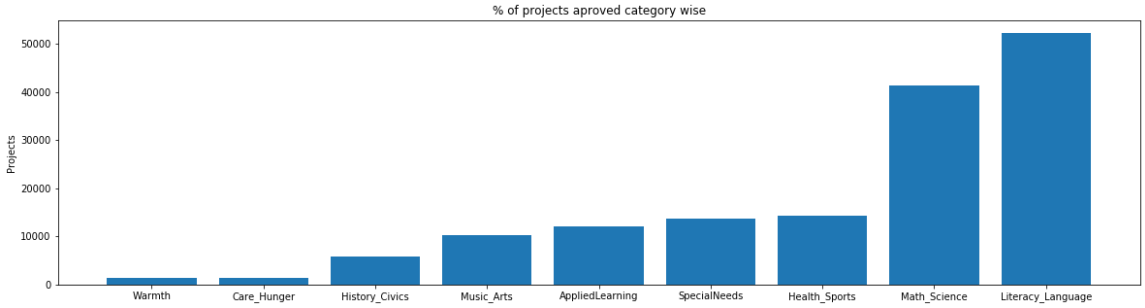
- Projects under the category of **Warmth and Care Hunger** have the maximum approval rate of **92.58%**. This shows that people tend to donate more towards child care and providing for food and shelter.
- Maximum number of project submitted are under the category of **Literacy_Language (23,655)**. They have an approval rate of **86.74%**.
- Least number of projects are submitted under the category of **AppliedLearning Math_Science (1,052)**. They also have the least acceptance rate of about **81.2%**.
- Project in **all categories** get approved **more than 80%** of the time.

In [133]:

```
# 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 [134]:

```
# 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 [135]:

```
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
```

1.2.5 Univariate Analysis: project_subject_subcategories

In [136]:

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

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

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

In [137]:

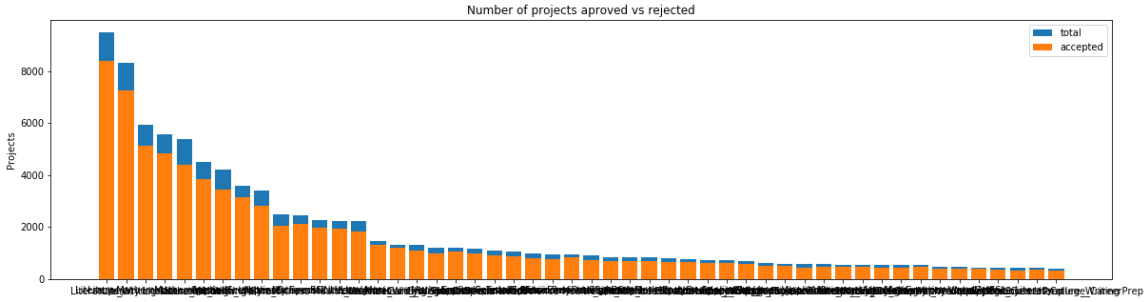
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[137]:

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

In [138]:

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




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

Obervations:

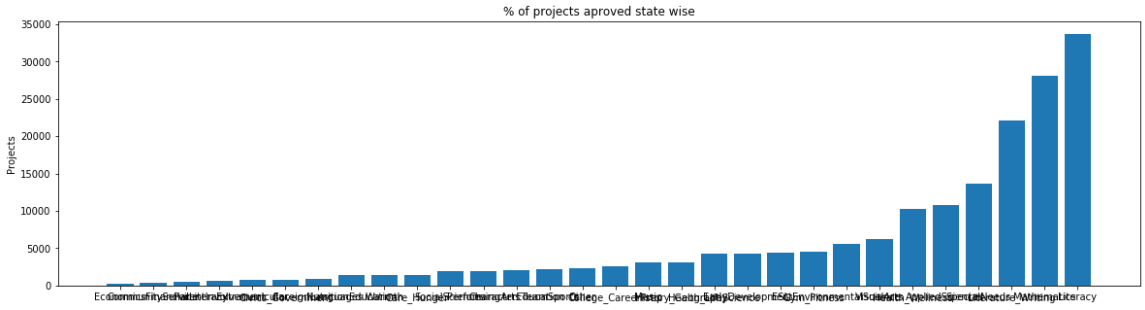
- The projects with subcategory **Literacy** have the highest average approval of **88.24%**.
- This is closely followed by subcategory **EnvironmentalScience and Literacy (87.6%)** and **Literacy Mathematics (87.2%)** .
- This shows that people tend to donate more for environmental causes and literacy.
- Projects in **all the subcategories** have an approval rate of **more than 80%**.

In [139]:

```
# 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 [140]:

```
# 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 [141]:

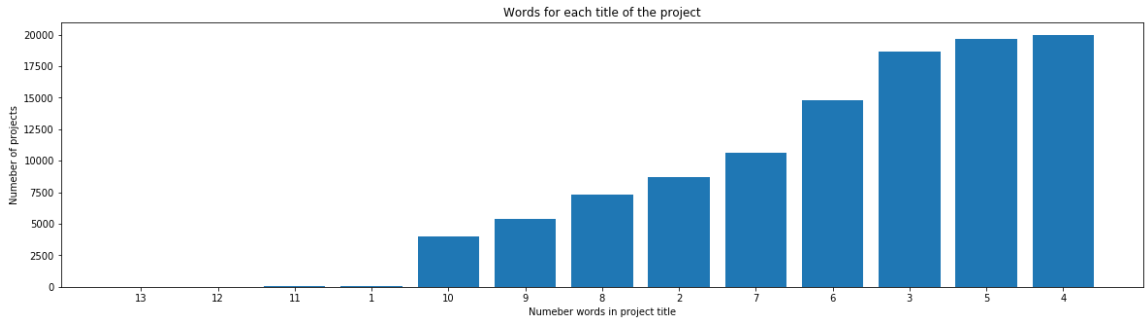
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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
```

1.2.6 Univariate Analysis: Text features (Title)

In [142]:

```
#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()
```



Observations:

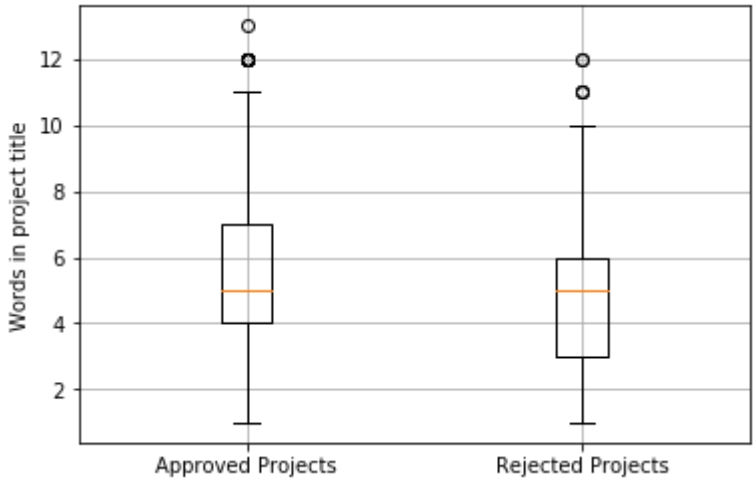
- The number of projects having **4 words in their title is maximum**.
- There are a very **few projects** that have more than **10 words** in their titles.
- **Most of the projects** tend to have titles of **length 2-7**.

In [143]:

```
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
```

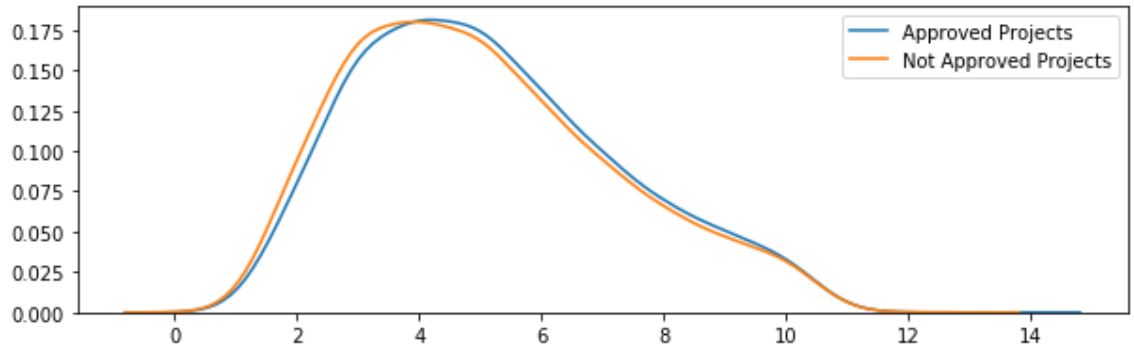
In [144]:

```
# 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 [145]:

```
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()
```



Obervations:

- The **PDFs** of project approved or not against the number of words **mostly overlap**, hence nothing can be said about the effect of number of words in the title and the approval rate.
- However, **projects which get approved have slighly more number of words than the ones which do not get approved**. The number of words lie between 4-6.
- From the box plot it can be observed that the **medians of approved and not approved projects are almost the same** with respect to number of words in the title.

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

In [146]:

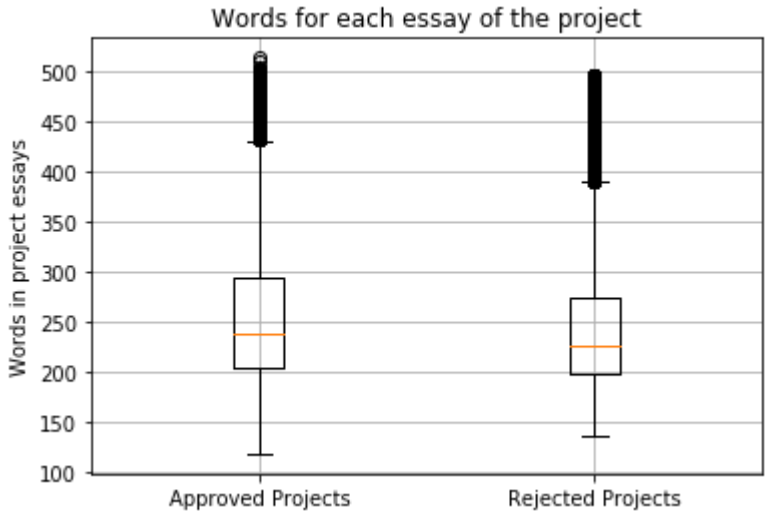
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [147]:

```
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
```

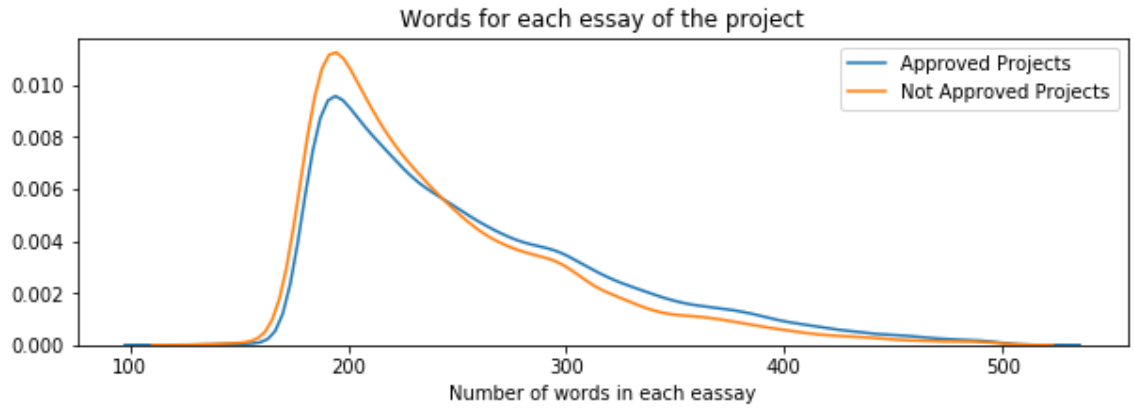
In [148]:

```
# 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 [149]:

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



Observations:

- From the density plot it can be concluded that **projects have essays with greater than 150 words**.
- Project with **essays of length 150-250 words** in length tend to **get rejected more**.
- **As the number of words in the essays increase the number of approved projected increases**.
- This is because longer essays have a better explanation of the issues and challenges for which they are seeking donation.

1.2.8 Univariate Analysis: Cost per project

In [150]:

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

Out[150]:

	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 [151]:

```
# 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[151]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [152]:

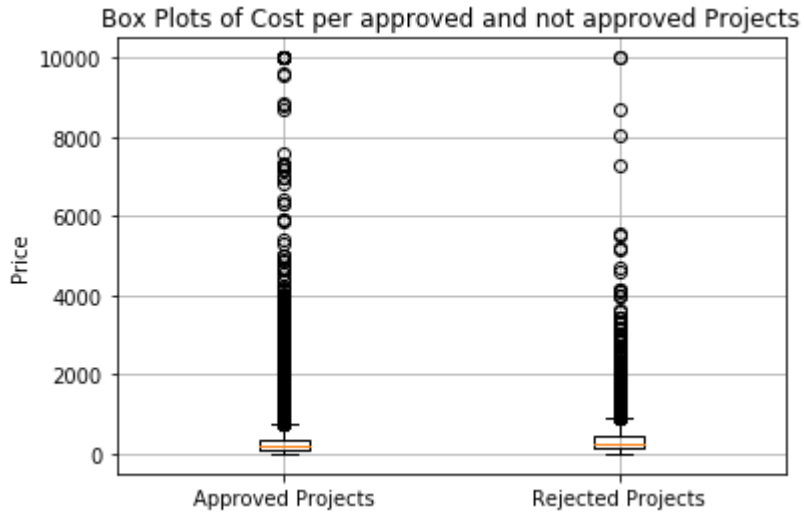
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [153]:

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

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

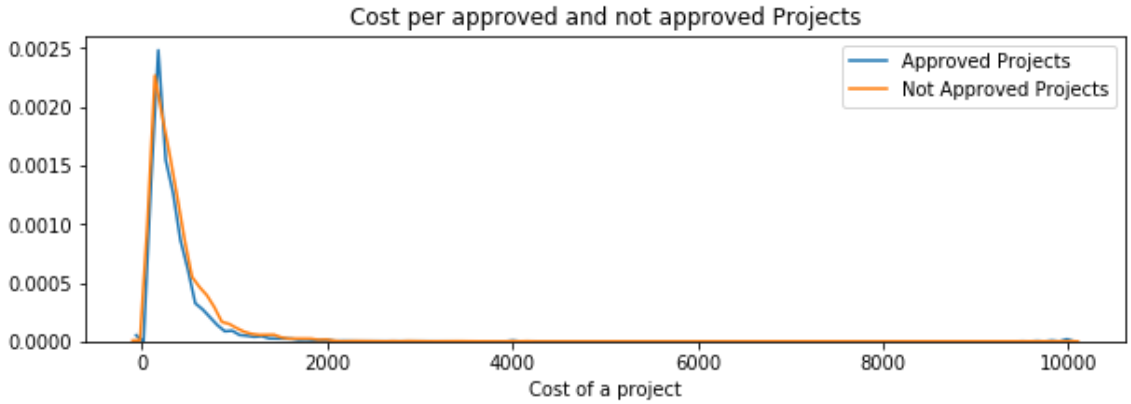
In [154]:

```
# 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 [155]:

```
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 [156]:

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

#If you get a ModuleNotFoundError error , install prettytable using: pip3 instal
l prettytable

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

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

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

Observations:

- The PDFs of approved and not approved projects overlap when plotted against the cost.
- However, it can be observed from the PDFs that as the cost increases the number of project approved decreases. This makes sense because costlier project require more donation and hence a strong representation to be funded for donation if approved.
- **50% of approved** projects cost less than 200 dollars and **50% of not approved projects** cost less than 264 dollars.
- At any point in time a approved project always costs less than a not approved project.

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

Univariate Analysis:Teacher previous posted project analysis

In [157]:

```
teacher_prev_proj_data = project_data[['teacher_id','teacher_number_of_previously_posted_projects']].copy()
teacher_prev_proj_data.head() # we now have teacher id corresponding to the number of project posted previously
```

Out[157]:

	teacher_id	teacher_number_of_previously_posted_projects
0	c90749f5d961ff158d4b4d1e7dc665fc	0
1	897464ce9ddc600bced1151f324dd63a	7
2	3465aaf82da834c0582ebd0ef8040ca0	1
3	f3cb9bffbba169bef1a77b243e620b60	4
4	be1f7507a41f8479dc06f047086a39ec	1

In [158]:

```
teacher_prev_proj_data.describe()
```

Out[158]:

	teacher_number_of_previously_posted_projects
count	109248.000000
mean	11.153165
std	27.777154
min	0.000000
25%	0.000000
50%	2.000000
75%	9.000000
max	451.000000

Observations:

- The mean of the previously posted project is 11.
- 50% of the teachers have posted 2 projects.

Bar graph of the top 50 teachers vs their previously posted projects

In [159]:

```
top_50_teacher = teacher_prev_proj_data.sort_values(['teacher_number_of_previously_posted_projects'], ascending=False)
top_50_teacher = top_50_teacher[:50]
top_50_teacher.head()
```

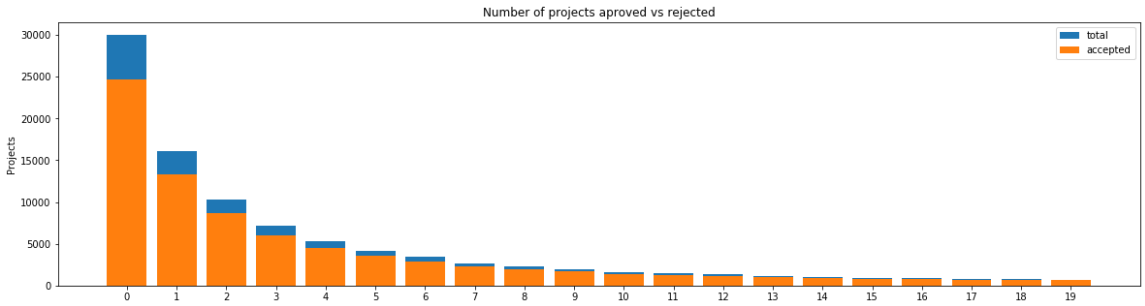
Out[159]:

	teacher_id	teacher_number_of_previously_posted_projects
88015	fa2f220b537e8653fb48878ebb38044d	451
106026	fa2f220b537e8653fb48878ebb38044d	437
50805	fa2f220b537e8653fb48878ebb38044d	433
72233	fa2f220b537e8653fb48878ebb38044d	432
13777	fa2f220b537e8653fb48878ebb38044d	428

- The teacher with id **fa2f220b537e8653fb48878ebb38044d** has posted maximum projects (451).

In [264]:

```
# teacher_id = top_50_teacher['teacher_id'].values

# number_of_prev_projects = top_50_teacher['teacher_number_of_previously_posted_
projects'].values
# plt.figure(figsize=(20,5))
# ind = np.arange(len(teacher_id))
# plt.bar(ind, list(number_of_prev_projects))
# plt.xlabel('Teacher ID', fontsize=12)
# plt.ylabel('No of projects posted previously', fontsize=12)
# plt.xticks(ind, list(teacher_id), fontsize=12, rotation=90)
# plt.title('Previously posted projects of top 50 teachers')
# plt.show()
univariate_barplots(project_data,'teacher_number_of_previously_posted_projects',
'project_is_approved',top=20)
```




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

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
    teacher_number_of_previously_posted_projects  project_is_approve
d  total  \
15                                           15                  81
8    942
16                                           16                  76
9    894
17                                           17                  71
2    803
18                                           18                  66
6    772
19                                           19                  63
2    710

         Avg
15  0.868365
16  0.860179
17  0.886675
18  0.862694
19  0.890141
```

In [161]:

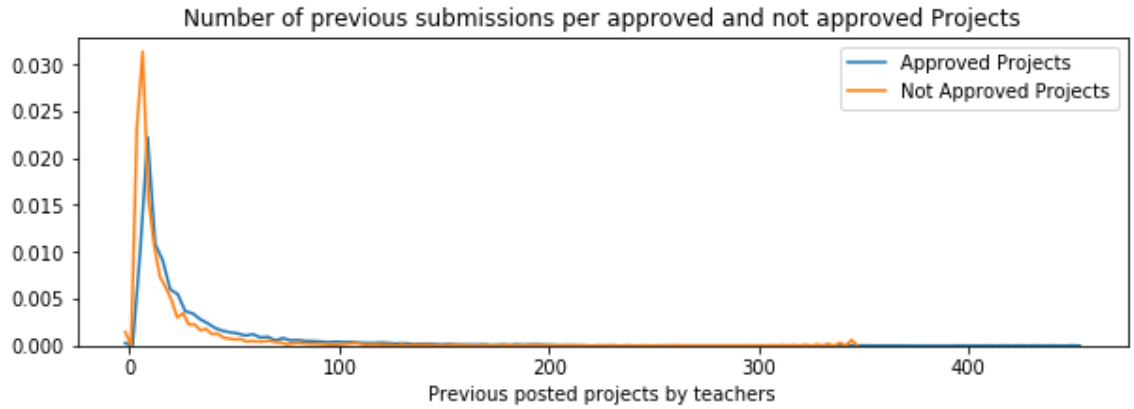
```
previously_posted_projects = project_data['teacher_number_of_previously_posted_p
rojects'].values
```

In [296]:

```
approved_previous = project_data[project_data['project_is_approved']==1]['teache
r_number_of_previously_posted_projects'].values
rejected_previous = project_data[project_data['project_is_approved']==0]['teache
r_number_of_previously_posted_projects'].values
```

In [297]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_previous, hist=False, label="Approved Projects")
sns.distplot(rejected_previous, hist=False, label="Not Approved Projects")
plt.title('Number of previous submissions per approved and not approved Project
s')
plt.xlabel('Previous posted projects by teachers')
plt.legend()
plt.show()
```

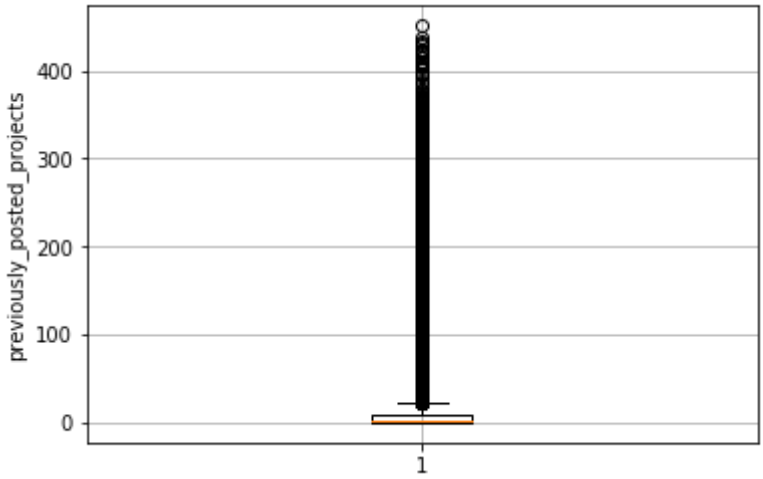


Obervation:

- From the above distribution we can conclude that teachers with less number of previously posted projects usually between 0-1 get rejected more.
- With more number of previously posted projects the accepted projects are more than rejected projects.
- A large number of teachers have posted a very small number of projects previously.

In [163]:

```
plt.boxplot([previously_posted_projects])
#plt.xticks([1],('previously_posted_projects'))
plt.ylabel('previously_posted_projects')
plt.grid()
plt.show()
```



Observations:

- It is observed that the second and third quartile value lie very close to each other.
- However, the maximum value of the previously posted projects is much higher than the third quartile.

In [164]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Percentile", "previously_posted_projects"]

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

```
+------------+-----------------------------+
| Percentile | previously_posted_projects |
+------------+-----------------------------+
|     0      |             0.0             |
|     5      |             0.0             |
|     10     |             0.0             |
|     15     |             0.0             |
|     20     |             0.0             |
|     25     |             0.0             |
|     30     |             1.0             |
|     35     |             1.0             |
|     40     |             1.0             |
|     45     |             2.0             |
|     50     |             2.0             |
|     55     |             3.0             |
|     60     |             4.0             |
|     65     |             5.0             |
|     70     |             7.0             |
|     75     |             9.0             |
|     80     |             12.0            |
|     85     |             18.0            |
|     90     |             28.0            |
|     95     |             53.0            |
|    100     |            451.0            |
+------------+-----------------------------+
```

Observations:

- The percentiles provide us a fairly good understanding of the previously posted projects by teachers.
- About 25th percentile value is **zero** for previously posted projects, this means that 25% of the teachers are new to the platform or have submitted a proposal of the first time.
- **50% of the teachers have posted less than equal to 2 projects previously.**
- **Only 5% of the teachers have posted close to 400 projects previously.** A small portion of the teachers is very active on the platform and is quite sigificantly applying for project approvals.

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

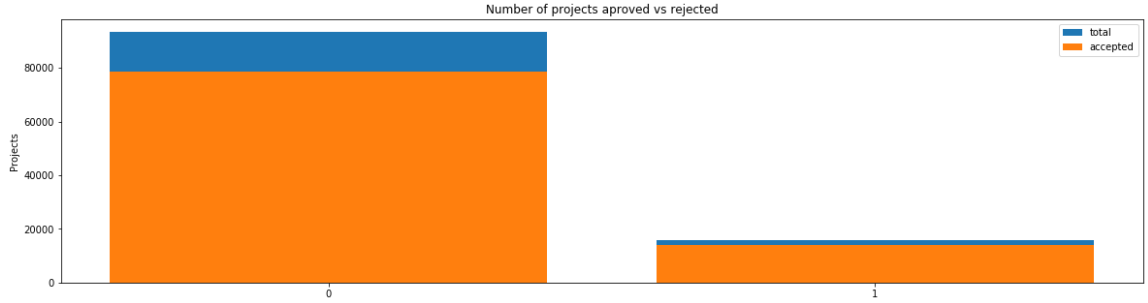
Check if the `presence of the numerical digits` in the `project_resource_summary` effects the acceptance of the project or not. If you observe that `presence of the numerical digits` is helpful in the classification, please include it for further process or you can ignore it.

In [298]:

```
#https://stackoverflow.com/questions/19859282/check-if-a-string-contains-a-numbe
r
def summary_with_digits(summary):
    '''
        This function takes a string as input ans searches for atleast one digi
t.
        If at least one digit is found, return 1
        Else, returns 0.
    '''
    if bool(re.search(r'\d+', summary)) == True:
        return 1
    return 0

#fetch the project resource summary
resource_summary = project_data['project_resource_summary']
#apply the function using map to the entire summary column
project_resource_summary_contains_digits = resource_summary.map(summary_with_dig
its)
# create a new column in the data frame with the results for further analysis
project_data['project_resource_summary_contains_digits'] = project_resource_summ
ary_contains_digits
#project_data.head()
```

In [299]:

```
univariate_barplots(project_data, 'project_resource_summary_contains_digits', 'p
roject_is_approved' , top=False)
```




```
   project_resource_summary_contains_digits  project_is_approved  to
tal  \
0                                         0                78616  93
492
1                                         1                14090  15
756

        Avg
0  0.840885
1  0.894263
==================================================
   project_resource_summary_contains_digits  project_is_approved  to
tal  \
0                                         0                78616  93
492
1                                         1                14090  15
756

        Avg
0  0.840885
1  0.894263
```

In [167]:

```
print("Percentage of projects HAVING digits in summary: ",15756/(15756+93492))
print("Percentage of projects HAVING digits in summary and getting APPROVED: ",1
4090/15756)
print("Percentage of projects NOT HAVING digits in summary and getting APPROVED:
",78616/93492)
```

```
Percentage of projects HAVING digits in summary:  0.1442223198594024
5
Percentage of projects HAVING digits in summary and getting APPROVE
D:  0.8942625031733943
Percentage of projects NOT HAVING digits in summary and getting APPR
OVED:  0.8408847815855902
```

Observation:

- It is observed that only 14% of the total projects have digits in the resource summary.
- If a resource summary contains digit it has 5% more chance of getting approved than if it does not have digits.
- Even though the precentage of acceptance is higher in projects containing digits, the total number of submissions not having digits in the resource summary is much higher.
- Hence, nothing can be concluded from this analysis.
- It might be the case that, instead of writing numbers in the form of digits, teachers choose to write numbers in words. For example, twenty for 20, two for 2.

1.3 Text preprocessing

1.3.1 Essay Text

In [168]:

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

Out[168]:

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

2 rows × 21 columns

In [169]:

```
# 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[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as thei
r second or third languages. We are a melting pot of refugees, immig
rants, and native-born Americans bringing the gift of language to ou
r school. \r\n\r\n We have over 24 languages represented in our Engl
ish Learner program with students at every level of mastery.  We als
o have over 40 countries represented with the families within our sc
hool.  Each student brings a wealth of knowledge and experiences to
us that open our eyes to new cultures, beliefs, and respect.\"The li
mits of your language are the limits of your world.\"-Ludwig Wittgen
stein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to
read and speak English along side of their children.  Sometimes this
creates barriers for parents to be able to help their child learn ph
onetics, letter recognition, and other reading skills.\r\n\r\nBy pro
viding these dvd's and players, students are able to continue their
mastery of the English language even if no one at home is able to as
sist.  All families with students within the Level 1 proficiency sta
tus, will be a offered to be a part of this program.  These educatio
nal videos will be specially chosen by the English Learner Teacher a
nd will be sent home regularly to watch.  The videos are to help the
child develop early reading skills.\r\n\r\nParents that do not have
access to a dvd player will have the opportunity to check out a dvd
player to use for the year.  The plan is to use these videos and edu
cational dvd's for the years to come for other EL students.\r\nnanna
n
==================================================
The 51 fifth grade students that will cycle through my classroom thi
s year all love learning, at least most of the time. At our school,
97.3% of the students receive free or reduced price lunch. Of the 56
0 students, 97.3% are minority students. \r\nThe school has a vibran
t community that loves to get together and celebrate. Around Hallowe
en there is a whole school parade to show off the beautiful costumes
that students wear. On Cinco de Mayo we put on a big festival with c
rafts made by the students, dances, and games. At the end of the yea
r the school hosts a carnival to celebrate the hard work put in duri
ng the school year, with a dunk tank being the most popular activit
y.My students will use these five brightly colored Hokki stools in p
lace of regular, stationary, 4-legged chairs. As I will only have a
total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During
independent reading time they will be used as special chairs student
s will each use on occasion. I will utilize them in place of chairs
at my small group tables during math and reading times. The rest of
the day they will be used by the students who need the highest amoun
t of movement in their life in order to stay focused on school.\r\n
\r\nWhenever asked what the classroom is missing, my students always
say more Hokki Stools. They can't get their fill of the 5 stools we
already have. When the students are sitting in group with me on the
Hokki Stools, they are always moving, but at the same time doing the
ir work. Anytime the students get to pick where they can sit, the Ho
kki Stools are the first to be taken. There are always students who
head over to the kidney table to get one of the stools who are disap
pointed as there are not enough of them. \r\n\r\nWe ask a lot of stu
dents to sit for 7 hours a day. The Hokki stools will be a compromis
e that allow my students to do desk work and move at the same time.
These stools will help students to meet their 60 minutes a day of mo
vement by allowing them to activate their core muscles for balance w
hile they sit. For many of my students, these chairs will take away
the barrier that exists in schools for a child who can't sit still.n
annan
==================================================
```

How do you remember your days of school? Was it in a sterile environ
ment with plain walls, rows of desks, and a teacher in front of the
room? A typical day in our room is nothing like that. I work hard to
create a warm inviting themed room for my students look forward to c
oming to each day.\r\n\r\nMy class is made up of 28 wonderfully uniq
ue boys and girls of mixed races in Arkansas.\r\nThey attend a Title
I school, which means there is a high enough percentage of free and
reduced-price lunch to qualify. Our school is an \"open classroom\"
concept, which is very unique as there are no walls separating the c
lassrooms. These 9 and 10 year-old students are very eager learners;
they are like sponges, absorbing all the information and experiences
and keep on wanting more.With these resources such as the comfy red
throw pillows and the whimsical nautical hanging decor and the blue
fish nets, I will be able to help create the mood in our classroom s
etting to be one of a themed nautical environment. Creating a classr
oom environment is very important in the success in each and every c
hild's education. The nautical photo props will be used with each ch
ild as they step foot into our classroom for the first time on Meet
the Teacher evening. I'll take pictures of each child with them, hav
e them developed, and then hung in our classroom ready for their fir
st day of 4th grade. This kind gesture will set the tone before eve
n the first day of school! The nautical thank you cards will be used
throughout the year by the students as they create thank you cards t
o their team groups.\r\n\r\nYour generous donations will help me to
help make our classroom a fun, inviting, learning environment from d
ay one.\r\n\r\nIt costs lost of money out of my own pocket on resour
ces to get our classroom ready. Please consider helping with this pr
oject to make our new school year a very successful one. Thank you!n
annan

==

My kindergarten students have varied disabilities ranging from speec
h and language delays, cognitive delays, gross/fine motor delays, to
autism. They are eager beavers and always strive to work their harde
st working past their limitations. \r\n\r\nThe materials we have are
the ones I seek out for my students. I teach in a Title I school whe
re most of the students receive free or reduced price lunch. Despit
e their disabilities and limitations, my students love coming to sch
ool and come eager to learn and explore.Have you ever felt like you
had ants in your pants and you needed to groove and move as you were
in a meeting? This is how my kids feel all the time. The want to be
able to move as they learn or so they say.Wobble chairs are the answ
er and I love then because they develop their core, which enhances g
ross motor and in Turn fine motor skills. \r\nThey also want to lear
n through games, my kids don't want to sit and do worksheets. They w
ant to learn to count by jumping and playing. Physical engagement is
the key to our success. The number toss and color and shape mats can
make that happen. My students will forget they are doing work and ju
st have the fun a 6 year old deserves.nannan

==

The mediocre teacher tells. The good teacher explains. The superior
teacher demonstrates. The great teacher inspires. -William A. Ward\r
\n\r\nMy school has 803 students which is makeup is 97.6% African-Am
erican, making up the largest segment of the student body. A typical
school in Dallas is made up of 23.2% African-American students. Most
of the students are on free or reduced lunch. We aren't receiving do
ctors, lawyers, or engineers children from rich backgrounds or neigh
borhoods. As an educator I am inspiring minds of young children and
we focus not only on academics but one smart, effective, efficient,
and disciplined students with good character.In our classroom we can
utilize the Bluetooth for swift transitions during class. I use a sp
eaker which doesn't amplify the sound enough to receive the message.

```
Due to the volume of my speaker my students can't hear videos or boo
ks clearly and it isn't making the lessons as meaningful. But with t
he bluetooth speaker my students will be able to hear and I can sto
p, pause and replay it at any time.\r\nThe cart will allow me to hav
e more room for storage of things that are needed for the day and ha
s an extra part to it I can use.  The table top chart has all of the
letter, words and pictures for students to learn about different let
ters and it is more accessible.nannan
==================================================
```

In [170]:

```
# 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 [171]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speec
h and language delays, cognitive delays, gross/fine motor delays, to
autism. They are eager beavers and always strive to work their harde
st working past their limitations. \r\n\r\nThe materials we have are
the ones I seek out for my students. I teach in a Title I school whe
re most of the students receive free or reduced price lunch.  Despit
e their disabilities and limitations, my students love coming to sch
ool and come eager to learn and explore.Have you ever felt like you
had ants in your pants and you needed to groove and move as you were
in a meeting? This is how my kids feel all the time. The want to be
able to move as they learn or so they say.Wobble chairs are the answ
er and I love then because they develop their core, which enhances g
ross motor and in Turn fine motor skills. \r\nThey also want to lear
n through games, my kids do not want to sit and do worksheets. They
want to learn to count by jumping and playing. Physical engagement i
s the key to our success. The number toss and color and shape mats c
an make that happen. My students will forget they are doing work and
just have the fun a 6 year old deserves.nannan
==================================================
```

In [172]:

```
# \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 kindergarten students have varied disabilities ranging from speec
h and language delays, cognitive delays, gross/fine motor delays, to
autism. They are eager beavers and always strive to work their harde
st working past their limitations.     The materials we have are the
ones I seek out for my students. I teach in a Title I school where m
ost of the students receive free or reduced price lunch.  Despite th
eir disabilities and limitations, my students love coming to school
and come eager to learn and explore.Have you ever felt like you had
ants in your pants and you needed to groove and move as you were in
a meeting? This is how my kids feel all the time. The want to be abl
e to move as they learn or so they say.Wobble chairs are the answer
and I love then because they develop their core, which enhances gros
s motor and in Turn fine motor skills.   They also want to learn thr
ough games, my kids do not want to sit and do worksheets. They want
to learn to count by jumping and playing. Physical engagement is the
key to our success. The number toss and color and shape mats can mak
e that happen. My students will forget they are doing work and just
have the fun a 6 year old deserves.nannan
```

In [173]:

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

```
My kindergarten students have varied disabilities ranging from speec
h and language delays cognitive delays gross fine motor delays to au
tism They are eager beavers and always strive to work their hardest
working past their limitations The materials we have are the ones I
seek out for my students I teach in a Title I school where most of t
he students receive free or reduced price lunch Despite their disabi
lities and limitations my students love coming to school and come ea
ger to learn and explore Have you ever felt like you had ants in you
r pants and you needed to groove and move as you were in a meeting T
his is how my kids feel all the time The want to be able to move as
they learn or so they say Wobble chairs are the answer and I love th
en because they develop their core which enhances gross motor and in
Turn fine motor skills They also want to learn through games my kids
do not want to sit and do worksheets They want to learn to count by
jumping and playing Physical engagement is the key to our success Th
e number toss and color and shape mats can make that happen My stude
nts will forget they are doing work and just have the fun a 6 year o
ld deserves nannan
```

In [174]:

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

In [175]:

```
# 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%|██████████| 109248/109248 [01:17<00:00, 1418.79it/s]

In [176]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[176]:

```
'my kindergarten students varied disabilities ranging speech languag
e delays cognitive delays gross fine motor delays autism they eager
beavers always strive work hardest working past limitations the mate
rials ones i seek students i teach title i school students receive f
ree reduced price lunch despite disabilities limitations students lo
ve coming school come eager learn explore have ever felt like ants p
ants needed groove move meeting this kids feel time the want able mo
ve learn say wobble chairs answer i love develop core enhances gross
motor turn fine motor skills they also want learn games kids not wan
t sit worksheets they want learn count jumping playing physical enga
gement key success the number toss color shape mats make happen my s
tudents forget work fun 6 year old deserves nannan'
```

1.3.2 Project title Text

In [177]:

```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_titles = []
# 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_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:03<00:00, 31658.58it/s]
```

In [178]:

```
# after preprocesing
preprocessed_titles[20001]
```

Out[178]:

```
'the beautiful life butterfly'
```

1. 4 Preparing data for models

In [179]:

```
project_data.columns
```

Out[179]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_s
tate',
       'project_submitted_datetime', 'project_grade_category', 'proj
ect_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_a
pproved',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
'quantity',
       'project_resource_summary_contains_digits'],
      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 [180]:

```
# 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'].value
s)
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLe
arning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_
Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [181]:

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


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolv
ement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages',
'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'Pe
rformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College
_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'E
arlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'Vis
ualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Lit
erature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [182]:

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

#One-hot encoding for state
unique_states = np.unique(project_data['school_state'].values)
vectorizer = CountVectorizer(vocabulary=list(unique_states), lowercase=False, bi
nary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


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

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA',
'HI', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'M
I', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'N
V', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'U
T', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
Shape of matrix after one hot encodig  (109248, 51)
```

In [184]:

```
#One-hot encoding for teacher_prefix
#remove nan from teacher prefix:
#https://stackoverflow.com/questions/21011777/how-can-i-remove-nan-from-list-pyt
hon-numpy
def clean_teacher_prefix(prefix):
    if str(prefix)!='nan':
        return prefix
    return "none"
teacher_prefix = project_data['teacher_prefix']
cleaned_teacher_prefix = teacher_prefix.map(clean_teacher_prefix)
project_data['clean_teacher_prefix'] = cleaned_teacher_prefix
unique_teacher_prefix = np.unique(project_data['clean_teacher_prefix'].values)
vectorizer = CountVectorizer(vocabulary=list(unique_teacher_prefix), lowercase=F
alse, binary=True)
vectorizer.fit(project_data['clean_teacher_prefix'].values)
print(vectorizer.get_feature_names())
teacher_prefix_one_hot = vectorizer.transform(project_data['clean_teacher_prefi
x'].values)
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
project_data.drop(['teacher_prefix'], axis=1, inplace=True)
```

```
['Dr.', 'Mr.', 'Mrs.', 'Ms.', 'Teacher', 'none']
Shape of matrix after one hot encodig  (109248, 6)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [194]:

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

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.4.2.2 Bag of Words on `project_title`

In [195]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("shape of TITLE BOW: ",title_bow.shape)
```

```
shape of TITLE BOW:  (109248, 3329)
```

1.4.2.3 TFIDF vectorizer

In [196]:

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

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [197]:

```
# Similarly you can vectorize for title also
vectorizer = TfidfVectorizer(min_df=10)
title_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of TITLES TFIDF-VECTOR",title_tfidf.shape)
```

```
Shape of TITLES TFIDF-VECTOR (109248, 3329)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [0]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[0]:

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

In [189]:

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

In [198]:

```
# average Word2Vec
# compute average word2vec for each review.
essay_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
    essay_avg_w2v_vectors.append(vector)

print(len(essay_avg_w2v_vectors))
print(len(essay_avg_w2v_vectors[0]))
```

```
100%|██████████| 109248/109248 [00:42<00:00, 2592.16it/s]

109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [199]:

```
# Similarly you can vectorize for title also
title_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in
 this list
for sentence in tqdm(preprocessed_titles): # 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
    title_avg_w2v_vectors.append(vector)

print(len(title_avg_w2v_vectors))
print(len(title_avg_w2v_vectors[0]))
```

```
100%|██████████| 109248/109248 [00:02<00:00, 46994.89it/s]

109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [200]:

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

In [201]:

```
# average Word2Vec
# compute average word2vec for each review.
essay_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored i
n this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    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 essay_tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf val
ue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors.append(vector)

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

```
100%|██████████| 109248/109248 [04:30<00:00, 403.16it/s]

109248
300
```

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

In [202]:

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

In [203]:

```
# Similarly you can vectorize for title also
# average Word2Vec
# compute average word2vec for each review.
title_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored i
n this list
for sentence in tqdm(preprocessed_titles): # 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 title_tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf val
ue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_vectors.append(vector)

print(len(title_tfidf_w2v_vectors))
print(len(title_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 109248/109248 [00:05<00:00, 20093.14it/s]

109248
300
```

1.4.3 Vectorizing Numerical features

In [204]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.
... 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 standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scal
ar.var_[0])}")

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

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

In [205]:

```
price_standardized
```

Out[205]:

```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

1.4.4 Merging all the above features

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

In [207]:

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

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [221]:

```
# 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, essay_bow, price_standar
dized))
print(X.shape)
print(type(X))
```

```
(109248, 16663)
<class 'scipy.sparse.coo.coo_matrix'>
```

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 [253]:

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

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

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

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

# for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
# for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_
y','Score'])
# colors = {0:'red', 1:'blue', 2:'green'}
# plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne
_df['Score'].apply(lambda x: colors[x]))
# plt.show()
```

Pre-processing data before we apply T-SNE

One hot encoding of project_grade_category

In [222]:

```
unique_project_grade_category = np.unique(project_data['project_grade_category'
])
vectorizer = CountVectorizer(vocabulary=list(unique_project_grade_category), low
ercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())
project_grade_category_one_hot = vectorizer.transform(project_data['project_grad
e_category'].values)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot.sh
ape)
```

```
['Grades 3-5', 'Grades 6-8', 'Grades 9-12', 'Grades PreK-2']
Shape of matrix after one hot encodig  (109248, 4)
```

Standardizing : teacher_number_of_previously_posted_project

In [223]:

```
prev_posted_project_scalar = StandardScaler()
prev_posted_project_scalar.fit(project_data['teacher_number_of_previously_posted
_projects'].values.reshape(-1,1)) # finding the mean and standard deviation of t
his data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scal
ar.var_[0])}")
# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized = price_scalar.transfo
rm(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-
1, 1))
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496

/home/neeraj/anaconda3/lib/python3.6/site-packages/sklearn/utils/val
idation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScal
er.

/home/neeraj/anaconda3/lib/python3.6/site-packages/sklearn/utils/val
idation.py:595: DataConversionWarning:

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

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

In [224]:

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

- Data has been pre-processed before.
- The features being used in t-sne vizualization are as follows:
  - school_state_one_hot
  - categories_one_hot
  - sub_categories_one_hot
  - teacher_prefix_one_hot
  - project_grade_category one hot
  - title_bow
  - price_standardized
  - teacher_number_of_previously_posted_projects_standardized

Merging all the above features

In [331]:

```
print("Print shape of the data before merging the features.")
print(school_state_one_hot.shape)
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(title_bow.shape)
print(price_standardized.shape)
print(teacher_number_of_previously_posted_projects_standardized.shape)
```

```
Print shape of the data before merging the features.
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 6)
(109248, 4)
(109248, 3329)
(109248, 1)
(109248, 1)
```

In [354]:

```
# taking the first 3000 points from each of the features

f1 = school_state_one_hot[:5000]
f2 = categories_one_hot[:5000]
f3 = sub_categories_one_hot[:5000]
f4 = teacher_prefix_one_hot[:5000]
f5 = project_grade_category_one_hot[:5000]
f6 = title_bow[:5000]
f7 = price_standardized[:5000]
f8 = teacher_number_of_previously_posted_projects_standardized[:5000]
```

In [355]:

```
X = hstack((f1,f2,f3,f4,f5,f6,f7,f8))
X.shape
```

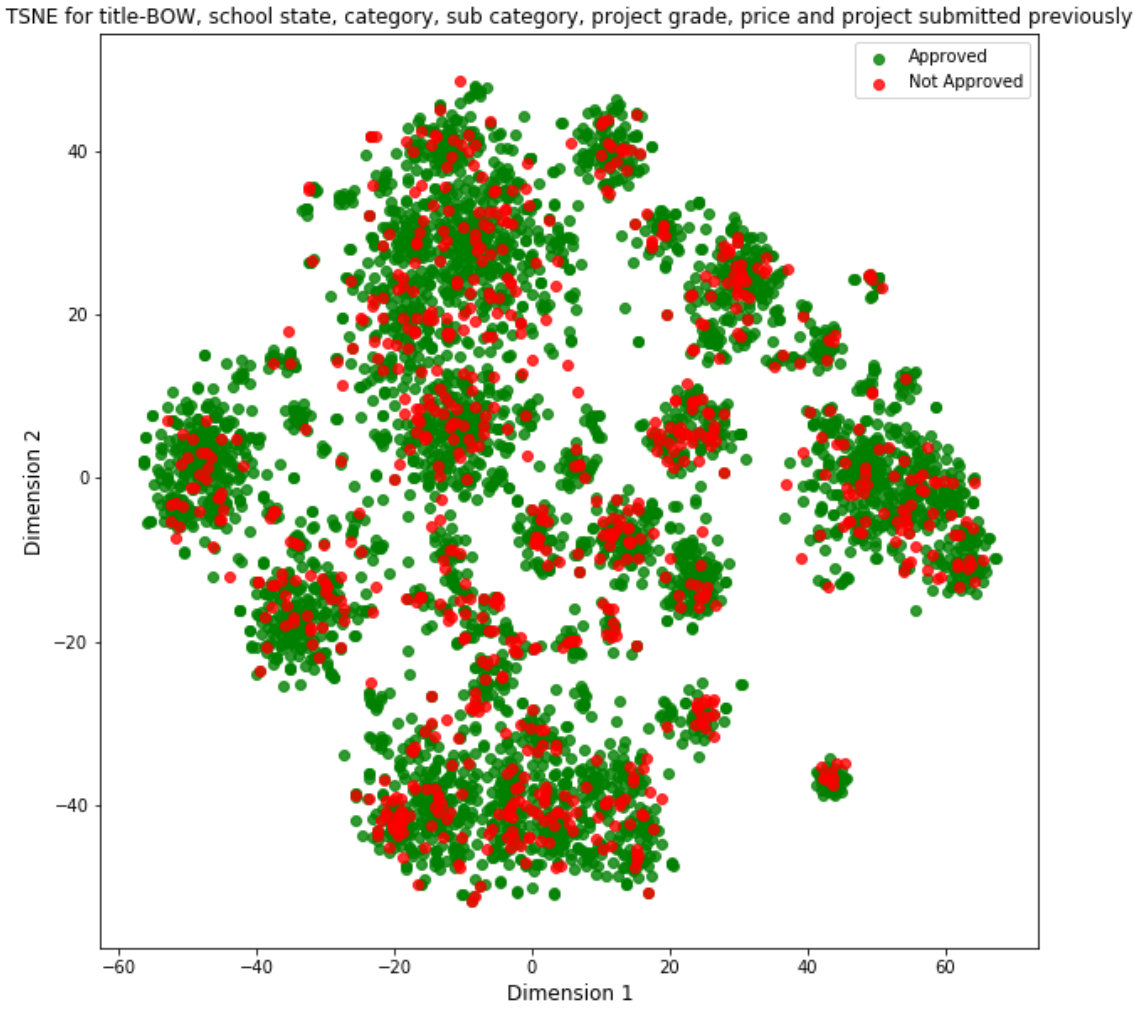
Out[355]:

```
(5000, 3431)
```

Applying T-SNE

In [356]:

```
# 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


y = project_data['project_is_approved'].values
y = y[:5000]
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

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

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

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'Score'])

#Plotting the tsne-reduced data
tsne_approved = for_tsne_df[for_tsne_df['Score']==1]
tsne_not_approved = for_tsne_df[for_tsne_df['Score']==0]
fig = plt.figure(figsize=(10,10))
fig.suptitle('', fontsize=12)
plt.scatter(tsne_approved['Dimension_x'], tsne_approved['Dimension_y'], alpha=0.
8, c="green",label=("Approved"))
plt.scatter(tsne_not_approved['Dimension_x'], tsne_not_approved['Dimension_y'],
alpha=0.8, c="red",label=("Not Approved"))
plt.title('TSNE for title-BOW, school state, category, sub category, project gra
de, price and project submitted previously')
plt.xlabel('Dimension 1', fontsize=12)
plt.ylabel('Dimension 2', fontsize=12)
plt.legend()
plt.show()
```



Observations:

- The above plot produced by the BOW representation of project titles along with other attributes produces overlapping data points.
- Presence of separate clusters can be seen but the datapoints overlap there as well and hence there is no clear distinction between approved and not approved projects.
- Nothing conclusive can be said from the BOW representation of the project titles.

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

In [339]:

```
title_tfidf = np.array(title_tfidf_w2v_vectors)
title_tfidf = title_tfidf[:5000]
title_tfidf.shape
```

Out[339]:

```
(5000, 300)
```

In [340]:

```
f1 = school_state_one_hot[:5000]
f2 = categories_one_hot[:5000]
f3 = sub_categories_one_hot[:5000]
f4 = teacher_prefix_one_hot[:5000]
f5 = project_grade_category_one_hot[:5000]
f6 = title_tfidf[:5000]
f7 = price_standardized[:5000]
f8 = teacher_number_of_previously_posted_projects_standardized[:5000]
```

In [341]:

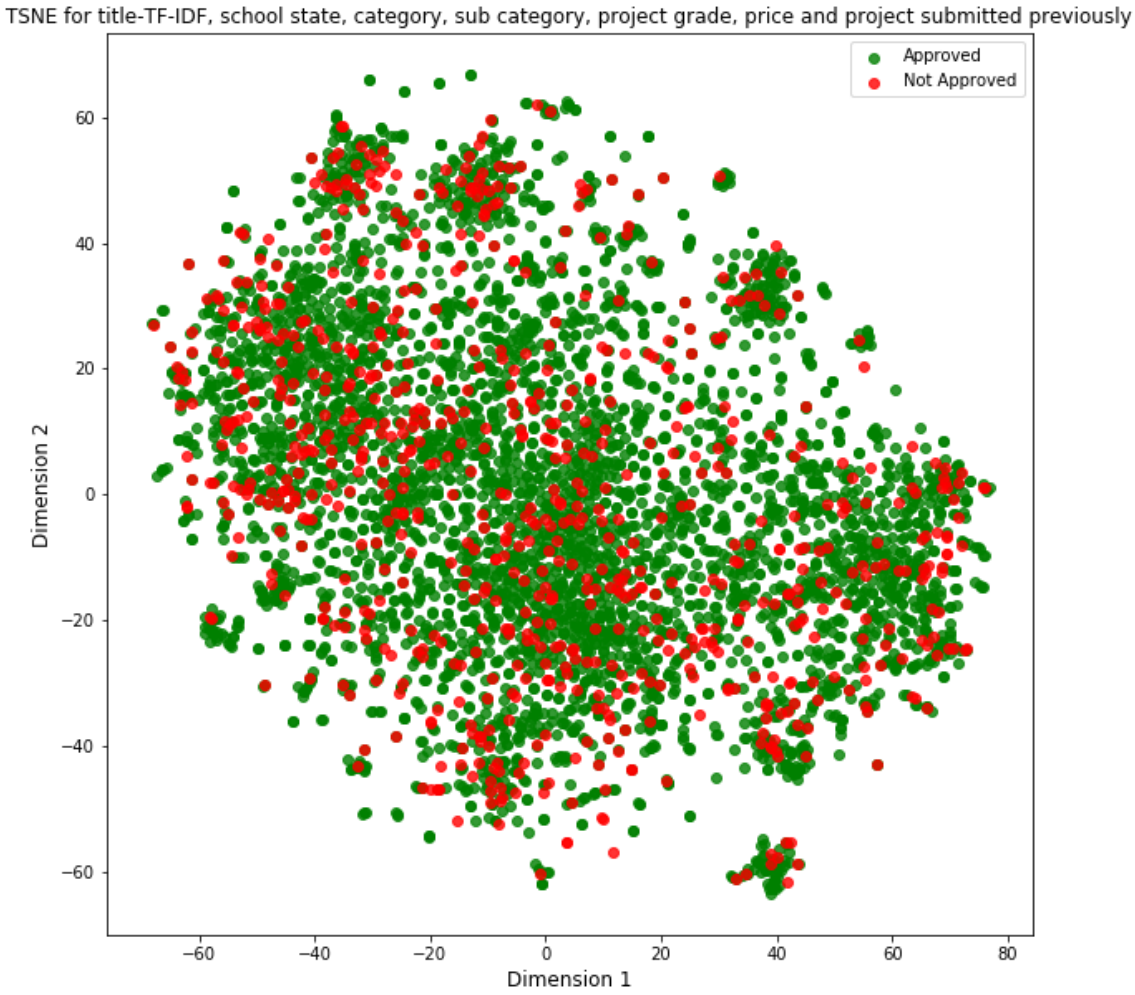
```
X = hstack((f1,f2,f3,f4,f5,f6,f7,f8))
X.shape
```

Out[341]:

```
(5000, 402)
```

In [342]:

```
y = project_data['project_is_approved'].values
y = y[:5000]
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

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

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'Score'])

#Plotting the tsne-reduced data
tsne_approved = for_tsne_df[for_tsne_df['Score']==1]
tsne_not_approved = for_tsne_df[for_tsne_df['Score']==0]
fig = plt.figure(figsize=(10, 10))
fig.suptitle('', fontsize=12)
plt.scatter(tsne_approved['Dimension_x'], tsne_approved['Dimension_y'], alpha=0.
8, c="green",label=("Approved"))
plt.scatter(tsne_not_approved['Dimension_x'], tsne_not_approved['Dimension_y'],
alpha=0.8, c="red",label=("Not Approved"))
plt.title('TSNE for title-TF-IDF, school state, category, sub category, project
 grade, price and project submitted previously')
plt.xlabel('Dimension 1', fontsize=12)
plt.ylabel('Dimension 2', fontsize=12)
plt.legend()
plt.show()
# colors = {0:'red', 1:'green'}
# fig = plt.figure()
# fig.suptitle('TSNE for title-TFIDF, school state, category, sub category, proj
ect grade, price and project submitted previously', fontsize=12)
# plt.xlabel('Dimension 1', fontsize=12)
# plt.ylabel('Dimension 2', fontsize=12)
# plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne
_df['Score'].apply(lambda x: colors[x]))
# plt.show()
```



Observations:

- The above scatter plot from t-sne does not separate the data well for vizualization.
- The approved and not approved data points overlap.
- Small separate clusters are visible but no clear conclusion can be drawn about the words in the project title from its TF-IDF representation.

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

In [343]:

```
title_avg_w2v = np.array(title_avg_w2v_vectors)
title_avg_w2v.shape
```

Out[343]:

```
(109248, 300)
```

In [344]:

```
f1 = school_state_one_hot[:5000]
f2 = categories_one_hot[:5000]
f3 = sub_categories_one_hot[:5000]
f4 = teacher_prefix_one_hot[:5000]
f5 = project_grade_category_one_hot[:5000]
f6 = title_avg_w2v[:5000]
f7 = price_standardized[:5000]
f8 = teacher_number_of_previously_posted_projects_standardized[:5000]
```

In [345]:

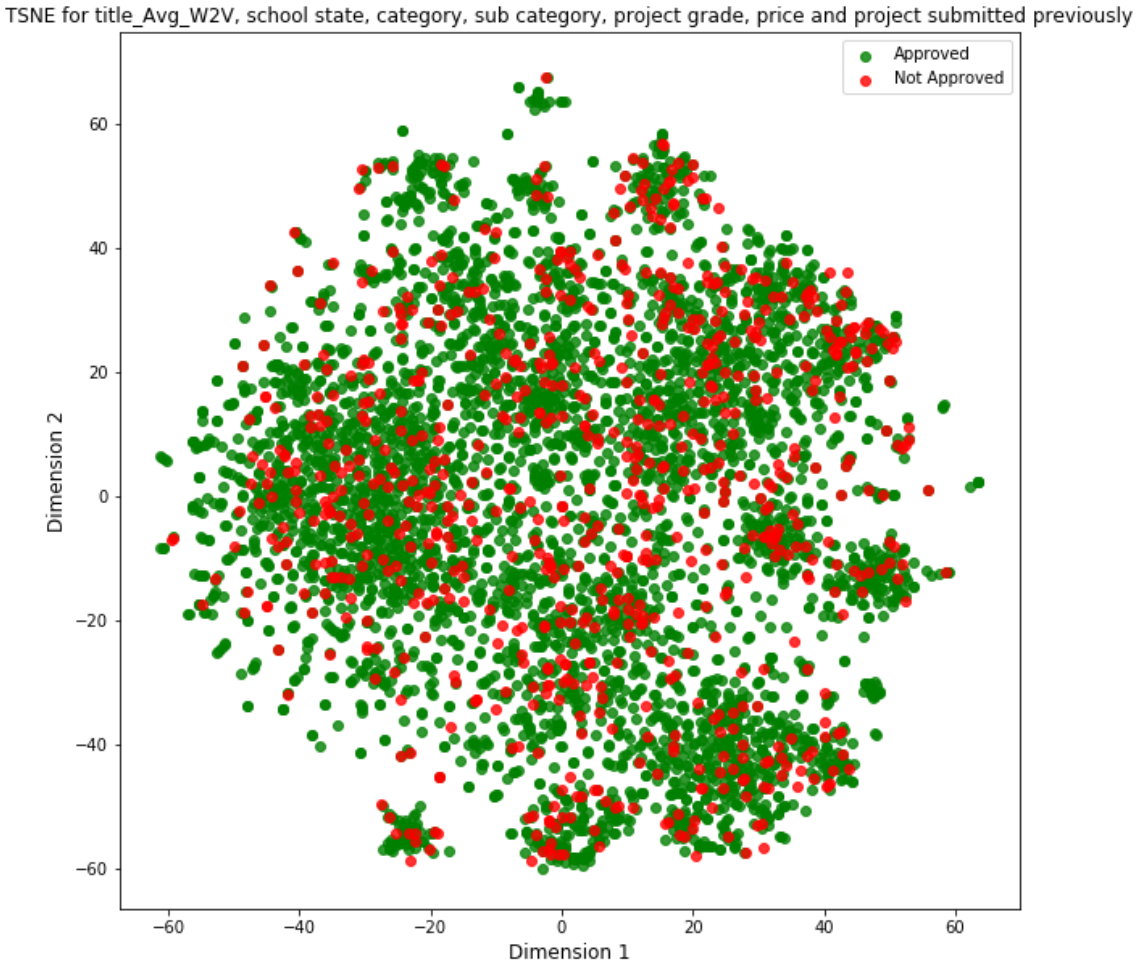
```
X = hstack((f1,f2,f3,f4,f5,f6,f7,f8))
X.shape
```

Out[345]:

```
(5000, 402)
```

In [346]:

```
y = project_data['project_is_approved'].values
y = y[:5000]
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

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

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'Score'])

#Plotting the tsne-reduced data
tsne_approved = for_tsne_df[for_tsne_df['Score']==1]
tsne_not_approved = for_tsne_df[for_tsne_df['Score']==0]
fig = plt.figure(figsize=(10, 10))
fig.suptitle('', fontsize=12)
plt.scatter(tsne_approved['Dimension_x'], tsne_approved['Dimension_y'], alpha=0.
8, c="green",label=("Approved"))
plt.scatter(tsne_not_approved['Dimension_x'], tsne_not_approved['Dimension_y'],
alpha=0.8, c="red",label=("Not Approved"))
plt.title('TSNE for title_Avg_W2V, school state, category, sub category, project
grade, price and project submitted previously')
plt.xlabel('Dimension 1', fontsize=12)
plt.ylabel('Dimension 2', fontsize=12)
plt.legend()
plt.show()
# colors = {0:'red', 1:'green'}
# fig = plt.figure()
# fig.suptitle('TSNE for title-AvgW2V, school state, category, sub category, pro
ject grade, price and project submitted previously', fontsize=12)
# plt.xlabel('Dimension 1', fontsize=12)
# plt.ylabel('Dimension 2', fontsize=12)
# plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne
_df['Score'].apply(lambda x: colors[x]))
# plt.show()
```



Observation:

- The scatter plot from average word to vec repreentation of the project titles does not separate the data into two clusters.
- Nothing can be said about the type of words present in approved and rejected projects from this plot.

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

In [347]:

```
title_tfidf_w2v = np.array(title_tfidf_w2v_vectors)
title_tfidf_w2v.shape
```

Out[347]:

```
(109248, 300)
```

In [348]:

```
f1 = school_state_one_hot[:5000]
f2 = categories_one_hot[:5000]
f3 = sub_categories_one_hot[:5000]
f4 = teacher_prefix_one_hot[:5000]
f5 = project_grade_category_one_hot[:5000]
f6 = title_tfidf_w2v[:5000]
f7 = price_standardized[:5000]
f8 = teacher_number_of_previously_posted_projects_standardized[:5000]
```

In [349]:

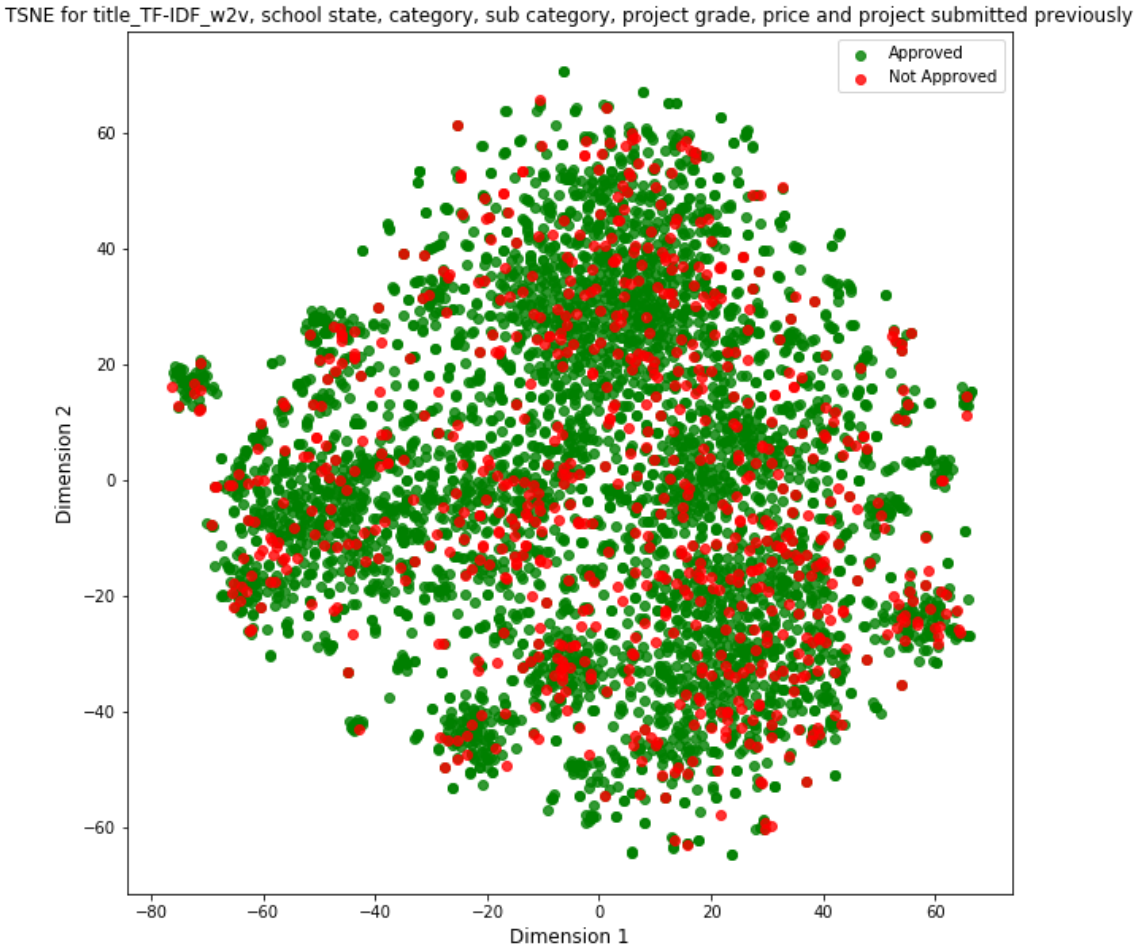
```
X = hstack((f1,f2,f3,f4,f5,f6,f7,f8))
X.shape
```

Out[349]:

```
(5000, 402)
```

In [350]:

```
y = project_data['project_is_approved'].values
y = y[:5000]
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

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

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'Score'])

#Plotting the tsne-reduced data
tsne_approved = for_tsne_df[for_tsne_df['Score']==1]
tsne_not_approved = for_tsne_df[for_tsne_df['Score']==0]
fig = plt.figure(figsize=(10, 10))
fig.suptitle('', fontsize=12)
plt.scatter(tsne_approved['Dimension_x'], tsne_approved['Dimension_y'], alpha=0.
8, c="green",label=("Approved"))
plt.scatter(tsne_not_approved['Dimension_x'], tsne_not_approved['Dimension_y'],
alpha=0.8, c="red",label=("Not Approved"))
plt.title('TSNE for title_TF-IDF_w2v, school state, category, sub category, proj
ect grade, price and project submitted previously')
plt.xlabel('Dimension 1', fontsize=12)
plt.ylabel('Dimension 2', fontsize=12)
plt.legend()
plt.show()
# colors = {0:'red', 1:'green'}
# fig = plt.figure()
# fig.suptitle('TSNE for title-TFIDF-W2V, school state, category, sub category,
 project grade, price and project submitted previously', fontsize=12)
# plt.xlabel('Dimension 1', fontsize=12)
# plt.ylabel('Dimension 2', fontsize=12)
# plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne
_df['Score'].apply(lambda x: colors[x]))
# plt.show()
```



Observations:

- The plot from TF-IDF weighted word2vec representation of the title along with other attributes produces a well scattered plot which is overlapping.
- Tiny separate clusters are visible in the plot but the data points for approved and not approved overlap there as well.
- Nothing can be concluded from the plot

2.5 TSNE with `BOW`, 'TF-IDF', `AvgW2V` and `TFIDF Weighted W2V` encoding of `project_title` feature

In [351]:

```
f1 = school_state_one_hot[:5000]
f2 = categories_one_hot[:5000]
f3 = sub_categories_one_hot[:5000]
f4 = teacher_prefix_one_hot[:5000]
f5 = project_grade_category_one_hot[:5000]
f6 = title_bow[:5000]
f7 = title_tfidf[:5000]
f8 = title_avg_w2v[:5000]
f9 = title_tfidf_w2v[:5000]
f10 = price_standardized[:5000]
f11 = teacher_number_of_previously_posted_projects_standardized[:5000]
```

In [352]:

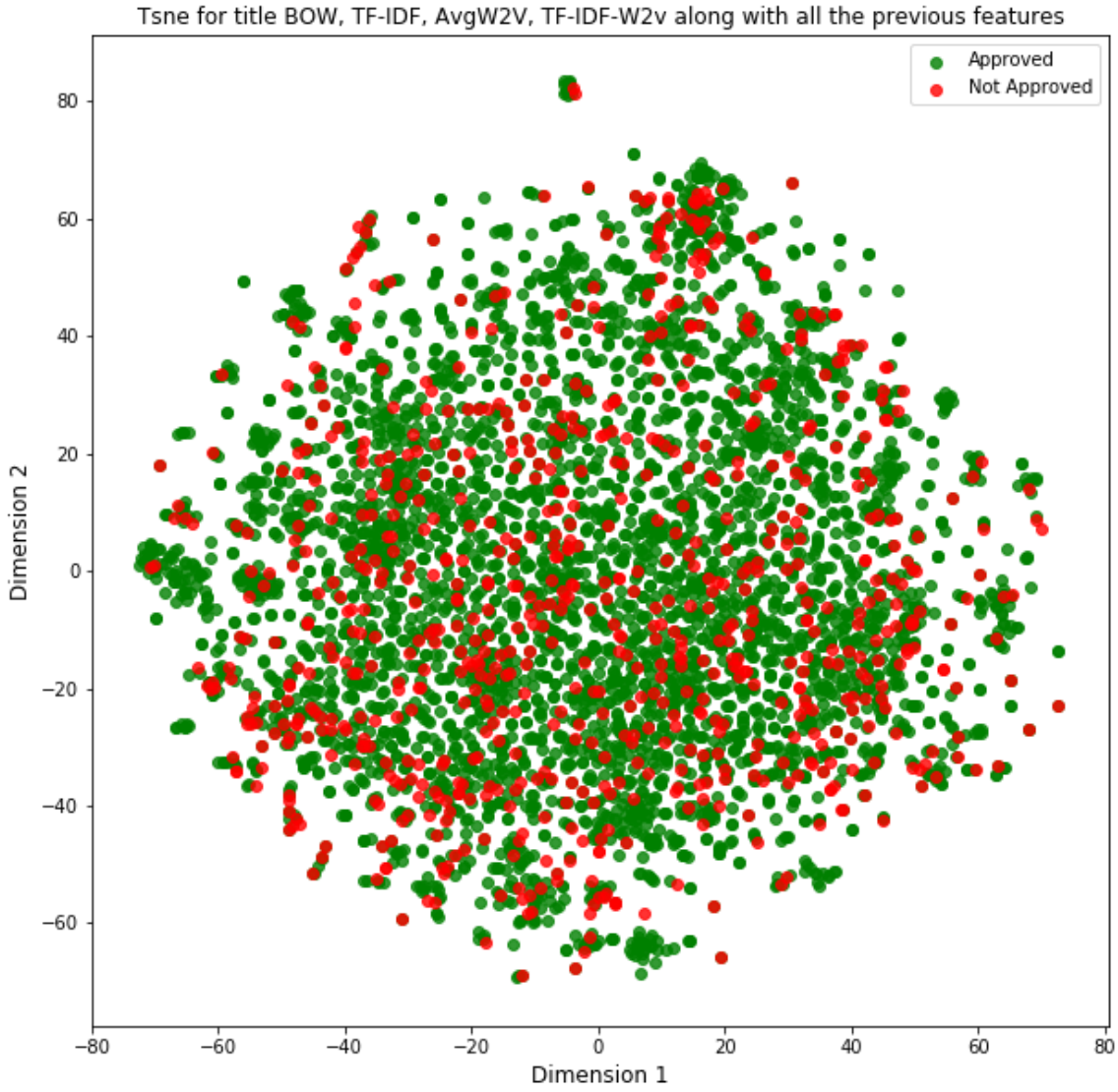
```
X = hstack((f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11))
X.shape
```

Out[352]:

```
(5000, 4331)
```

In [353]:

```
y = project_data['project_is_approved'].values
y = y[:5000]
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

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

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'Score'])

#Plotting the tsne-reduced data
tsne_approved = for_tsne_df[for_tsne_df['Score']==1]
tsne_not_approved = for_tsne_df[for_tsne_df['Score']==0]
fig = plt.figure(figsize=(10, 10))
fig.suptitle('', fontsize=12)
plt.scatter(tsne_approved['Dimension_x'], tsne_approved['Dimension_y'], alpha=0.
8, c="green",label=("Approved"))
plt.scatter(tsne_not_approved['Dimension_x'], tsne_not_approved['Dimension_y'],
alpha=0.8, c="red",label=("Not Approved"))
plt.title('Tsne for title BOW, TF-IDF, AvgW2V, TF-IDF-W2v along with all the pre
vious features')
plt.xlabel('Dimension 1', fontsize=12)
plt.ylabel('Dimension 2', fontsize=12)
plt.legend()
plt.show()
# colors = {0:'red', 1:'green'}
# fig = plt.figure()
# fig.suptitle('TSNE for title-TFIDF-W2V, school state, category, sub category,
 project grade, price and project submitted previously', fontsize=12)
# plt.xlabel('Dimension 1', fontsize=12)
# plt.ylabel('Dimension 2', fontsize=12)
# plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne
_df['Score'].apply(lambda x: colors[x]))
# plt.show()
```


Tsne for title BOW, TF-IDF, AvgW2V, TF-IDF-W2v along with all the previous features
Approved
Not Approved
Dimension 2
Dimension 1
80
60
40
20
0
−20
−40
−60
−80
−60
−40
−20
0
20
40
60
80

Obervations:

- The above plot has all the types of vector representation of title text along with other attributes, yet it produces a plot which is more or less similar to other techniques.
- Here as well we see that most of the data points are overlapping with no clear distinction between approved and not approved projects.
- Like TF_IDF Word2Vec, this plot also shows tiny separated clusters with overlap.
- Nothing can be concluded from the plot.

2.5 Summary

- The above visualization through T-SNE of the Donors Choose Application approval dataset we find that most of the data points overlap for BOW, TF-IDF, Avg W2V and TF-IDF weighted W2V.
- Only a very tiny clusters of approved and not approved projects exist scattered all over the plot.
- As T-SNE is not able to separate the data into indistinct clusters, we cannot conclude anything from the above visualization apart from the existence of tiny clusters of data scattered all over.
- **Conclusion:** Since even all the text to vector characterization fail to separate data point. We can say that **most of the words are same in the title field of approved and not approved projects**.