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.
`project_title`	Title of the • Art Will •
`project_grade_category`	Grade level of students for which the project is targeted. C e • • • •

Feature	
project_subject_categories	One or more (comma-separated) subject categories for t following enume • A • • • • H • Lite • • N • • • • N • Literacy & Language,
school_state	State where school is located (Two-lett (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviatio
project_subject_subcategories	One or more (comma-separated) subject subcatego • • Literature & Writing,
project_resource_summary	An explanation of the resources needed for the • My students need hands on literacy mater
project_essay_1	Firs
project_essay_2	Second
project_essay_3	Third
project_essay_4	Fourth
project_submitted_datetime	Datetime when project application was submitted. **Exan**
teacher_id	A unique identifier for the teacher of the proposed bdf8baa8fedef6bfe
teacher_prefix	Teacher's title. One of the following e • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by

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

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

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

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

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

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

Notes on the Essay Data

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

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_4:** "Close by sharing why your project will make a difference"

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

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [2]:
```
%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
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

1.1 Reading Data

In [3]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [4]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'scho
ol_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [5]:
```
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']
```

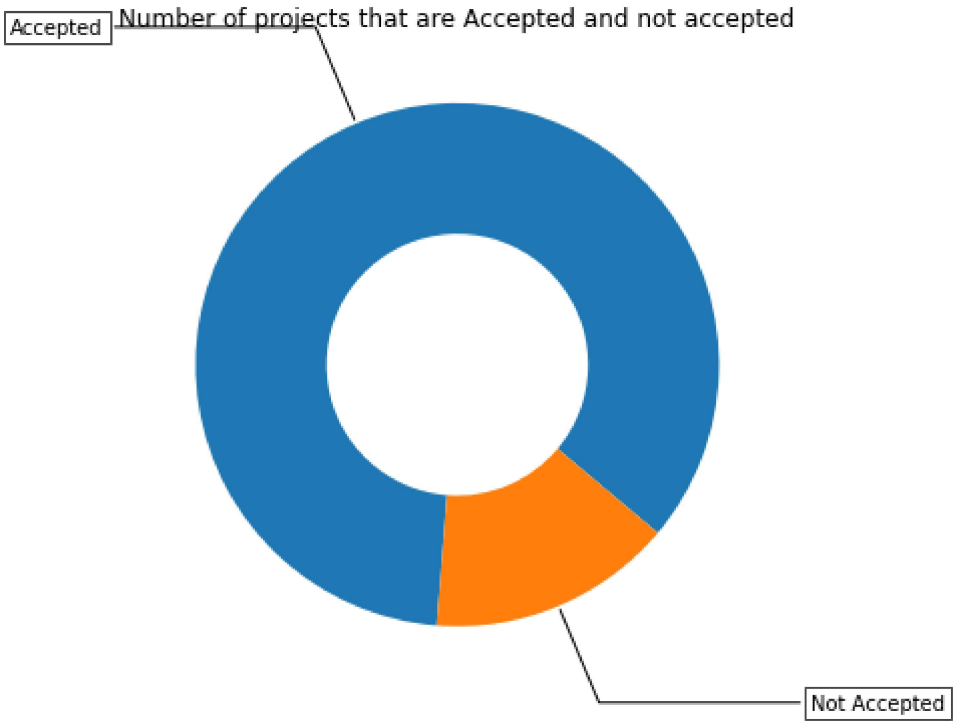
Out[5]:

	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 [6]:
```
# 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#s


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

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("Number of projects that are Accepted and not accepted")

plt.show()
```

```
Number of projects thar are approved for funding  92706 , ( 84.8583040422 %)
Number of projects thar are not approved for funding  16542 , ( 15.1416959578
%)
```



Conclusion

Number of projects thar are approved for funding 92706 , (84.8583040422 %)
Number of projects thar are not approved for funding 16542 , (15.1416959578 %)

1.2.1 Univariate Analysis: School State

In [7]:
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084

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

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

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

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

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

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

Out[7]:
```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n
\nscl (https://datascience.stackexchange.com/a/9620\n\nscl) = [[0.0, \'rgb(242,
240,247)\'],[0.2, \'rgb(218,218,235)\'],[0.4, \'rgb(188,189,220)\'],
[0.6, \'rgb(158,154,200)\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)
\']]\n\ndata = [ dict(\n        type=\'choropleth\',\n        colorscale = sc
l,\n        autocolorscale = False,\n        locations = temp[\'state_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 = dic
t(title = "% of pro")\n    ) ]\n\nlayout = dict(\n        title = \'Project Pro
posals % of Acceptance Rate by US States\',\n        geo = dict(\n            s
cope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n
showlakes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n
 ),\n    )\n\nfig = go.Figure(data=data, layout=layout)\noffline.iplot(fig, fil
ename=\'us-map-heat-map\')\n'
```

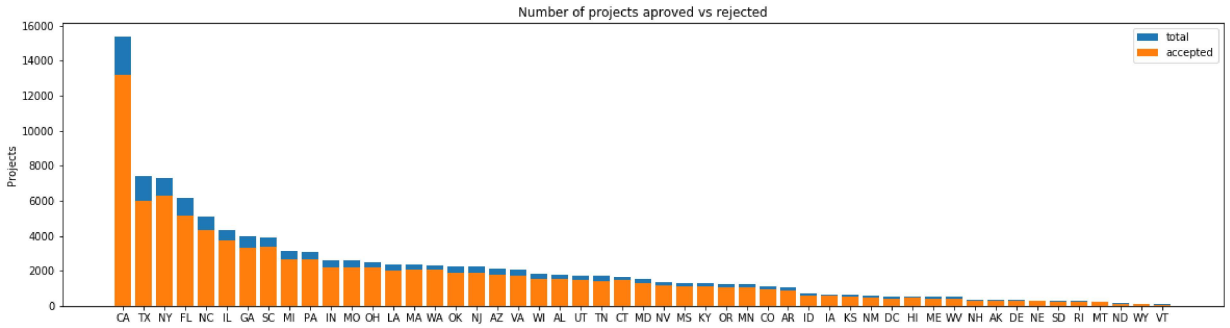
In [8]:
```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstab
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
print('='*50)
print("States with highest % approvals")
print(temp.tail(5))
```

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

In [9]:
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_mark
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

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

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

In [10]:
```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/5154
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).su

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

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

    if top:
        temp = temp[0:top]

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

In [11]:
```
univariate_barplots(project_data, 'school_state', 'project_is_approved', False)
```



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

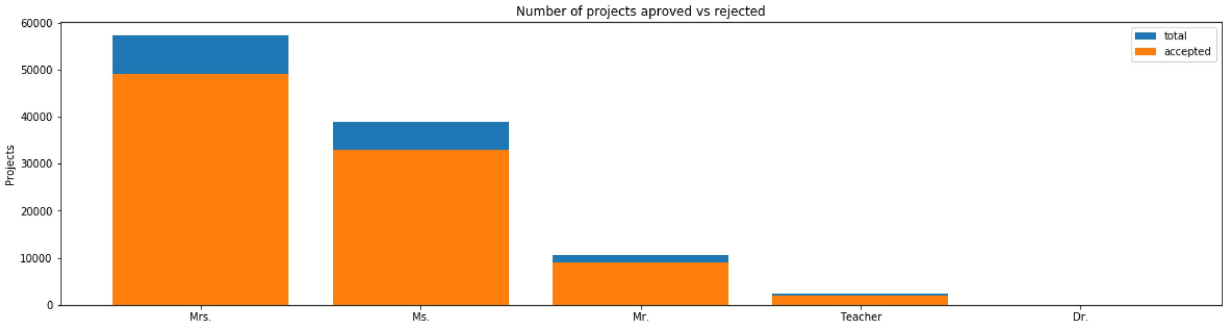
==========

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

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

1.2.2 Univariate Analysis: teacher_prefix

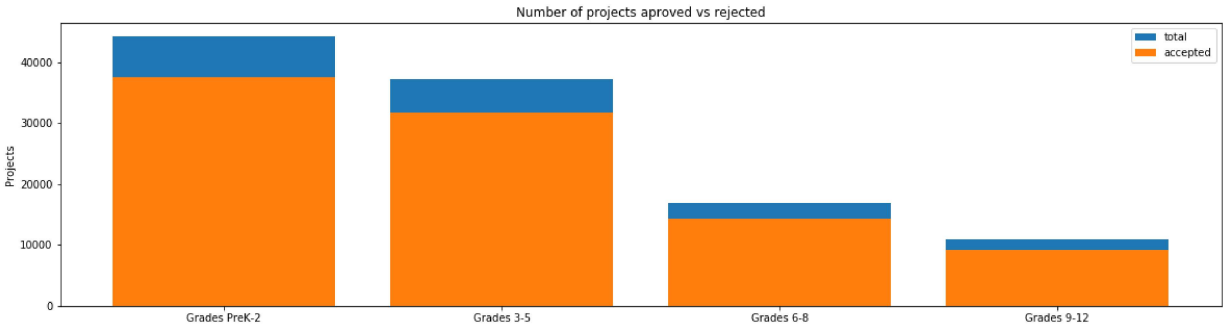
In [12]:
```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=F
```




```
  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
```

1.2.3 Univariate Analysis: project_grade_category

In [13]:
```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved'
```




```
  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
```

1.2.4 Univariate Analysis: project_subject_categories

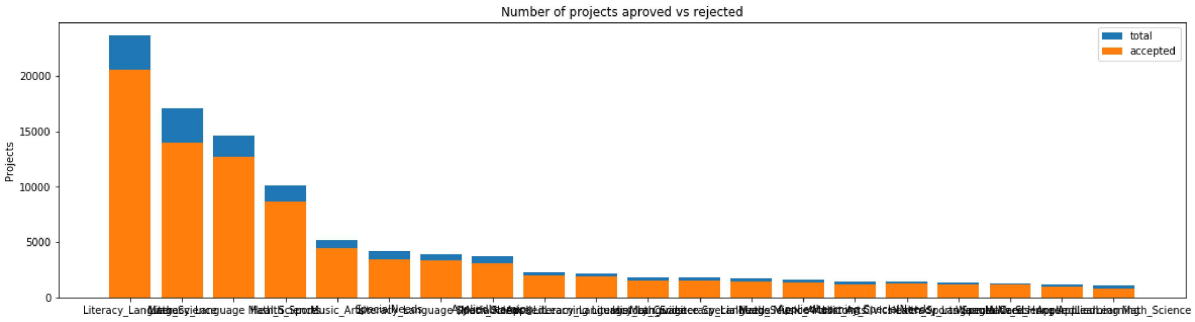
In [14]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-i
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", "
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the traili
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [15]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[15]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [16]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=
```



	clean_categories	project_is_approved	total	Avg
24	Literacy_Language	20520	23655	0.867470
32	Math_Science	13991	17072	0.819529
28	Literacy_Language Math_Science	12725	14636	0.869432
8	Health_Sports	8640	10177	0.848973
40	Music_Arts	4429	5180	0.855019

==========

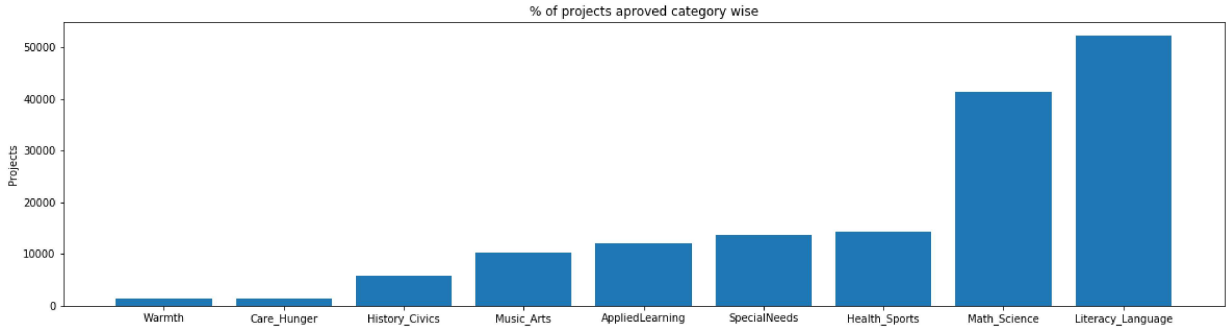
	clean_categories	project_is_approved	total	Avg
19	History_Civics Literacy_Language	1271	1421	0.894441

Conclusion

1. clean category with Maximum project approval : Warmth Care_Hunger
2. clean category with Minimum project approval : AppliedLearning Math_Science

In [17]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

In [18]:
```
# 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 [19]:
```
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
```

Conclusion

Project Approval Counts:

1. Warmth : 1388
2. Care_Hunger : 1388
3. History_Civics : 5914
4. Music_Arts : 10293
5. AppliedLearning : 12135
6. SpecialNeeds : 13642
7. Health_Sports : 14223
8. Math_Science : 41421
9. Literacy_Language : 52239

1.2.5 Univariate Analysis: project_subject_subcategories

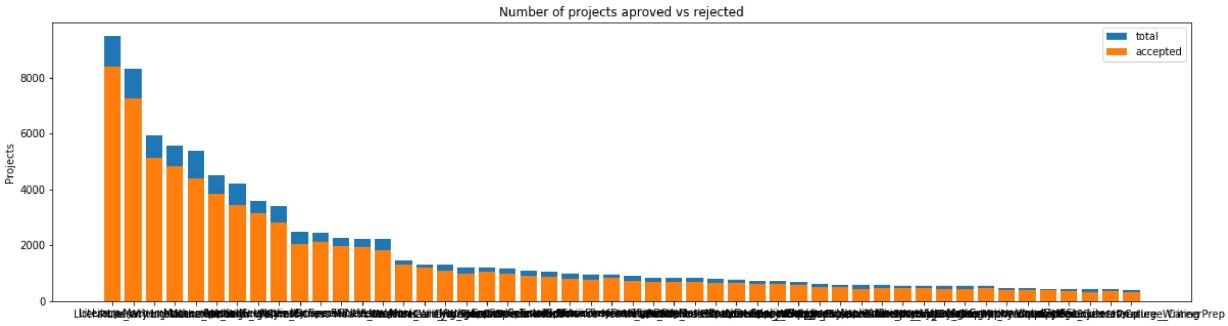
In [20]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

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

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", "
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the traili
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [21]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [22]:
```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', t
```



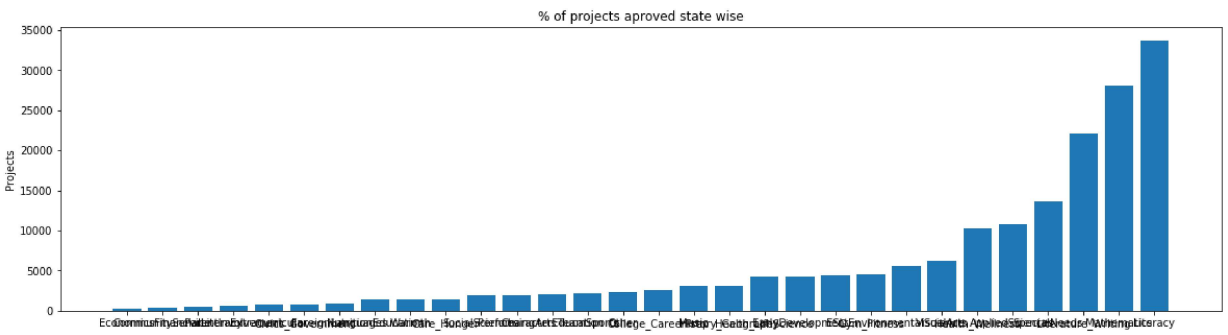
	clean_subcategories	project_is_approved	total	Avg
317	Literacy	8371	9486	0.882458
319	Literacy Mathematics	7260	8325	0.872072
331	Literature_Writing Mathematics	5140	5923	0.867803
318	Literacy Literature_Writing	4823	5571	0.865733
342	Mathematics	4385	5379	0.815207

==========

	clean_subcategories	project_is_approved	total	Avg
196	EnvironmentalScience Literacy	389	444	0.876126
127	ESL	349	421	0.828979
79	College_CareerPrep	343	421	0.814727
17	AppliedSciences Literature_Writing	361	420	0.859524
3	AppliedSciences College_CareerPrep	330	405	0.814815

In [23]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [24]:
```
# 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()
```

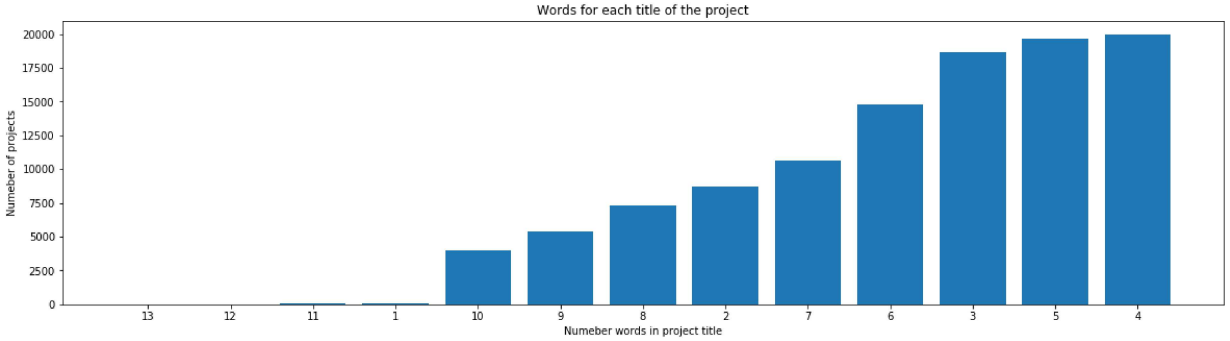


In [25]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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
```

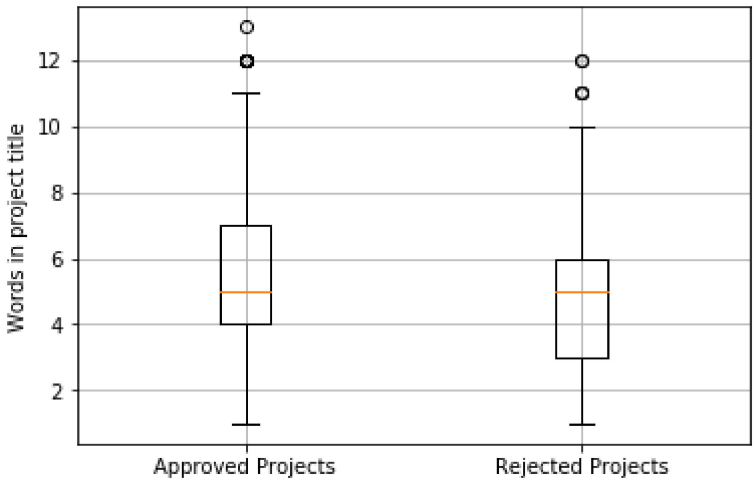
1.2.6 Univariate Analysis: Text features (Title)

In [26]:
```
#How to calculate number of words in a string in DataFrame: https://stackoverflow
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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

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

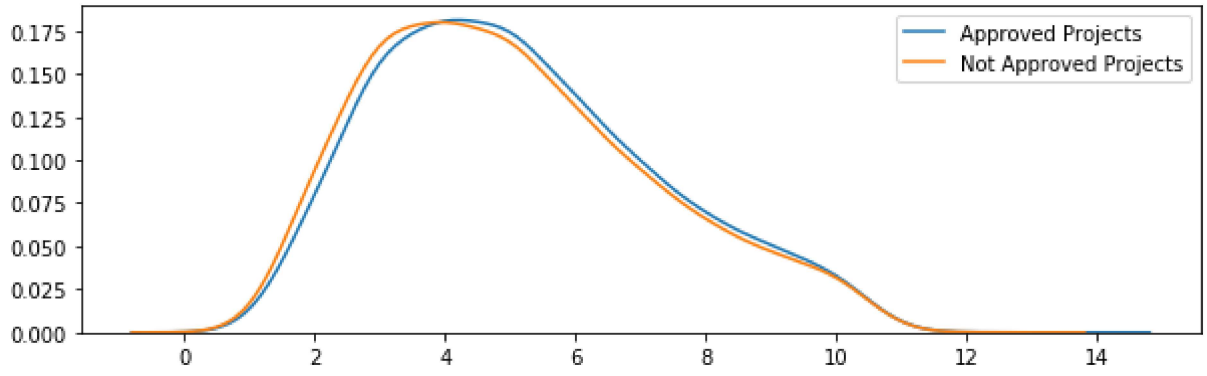


In [27]:
```
approved_title_word_count = project_data[project_data['project_is_approved']==1][
approved_title_word_count = approved_title_word_count.values

rejected_title_word_count = project_data[project_data['project_is_approved']==0][
rejected_title_word_count = rejected_title_word_count.values
```

In [28]:
```
# 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 [29]:
```
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()
```



Conclusion

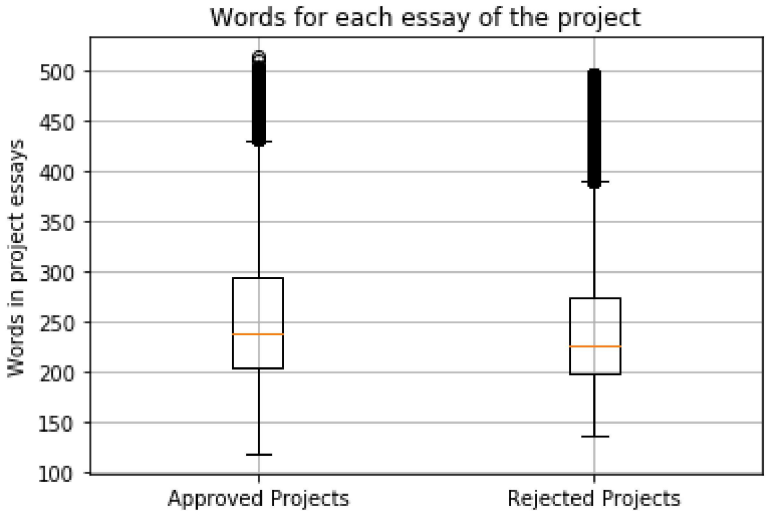
Approval or disapproval do not depends upon the length of the words in the project title

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

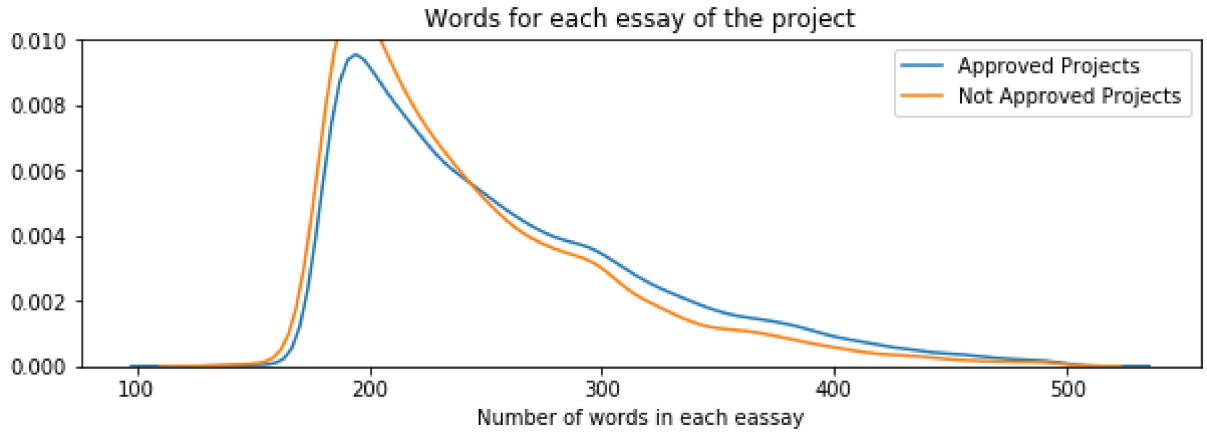
In [30]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [31]:
```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0]['essay
rejected_word_count = rejected_word_count.values
```

In [32]:
```
# 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 [33]:
```
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()
```



Conclusion

Approval or disapproval do not depends upon the length of the words in each essay

1.2.8 Univariate Analysis: Cost per project

In [34]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[34]:

	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 [35]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).r
price_data.head(2)
```

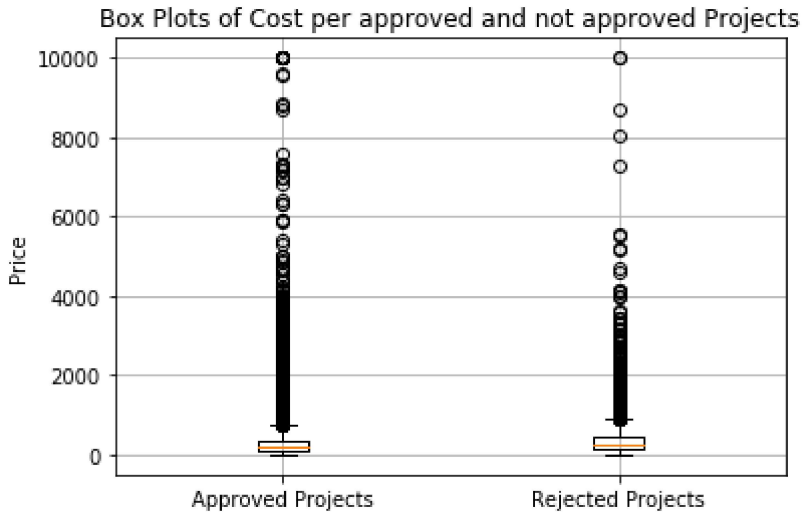
Out[35]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

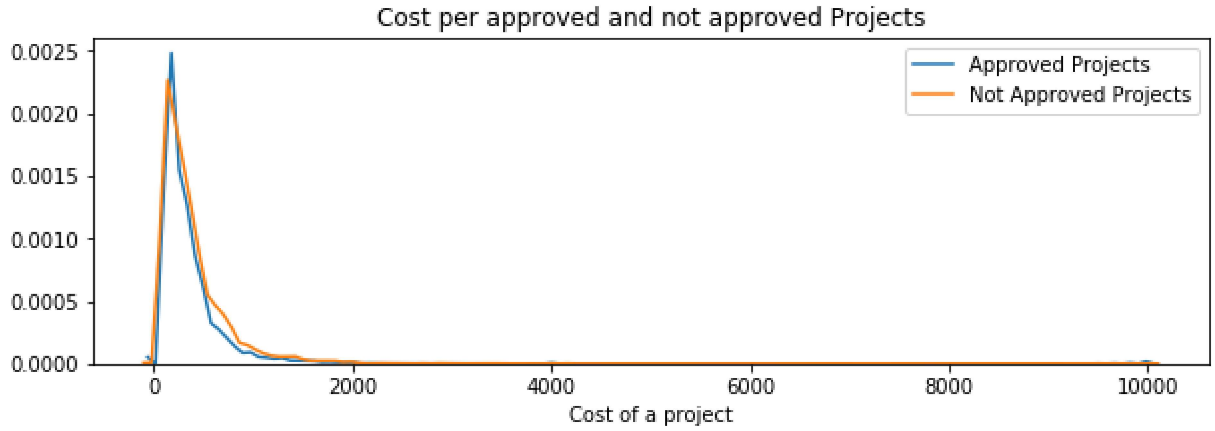
In [36]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [37]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].va

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

In [38]:
```
# 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 [39]:
```
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 [40]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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.percen
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        |
+------------+------------------+-----------------------+
```

Conclusion

Approval or disapproval do not depends upon the Price of the project

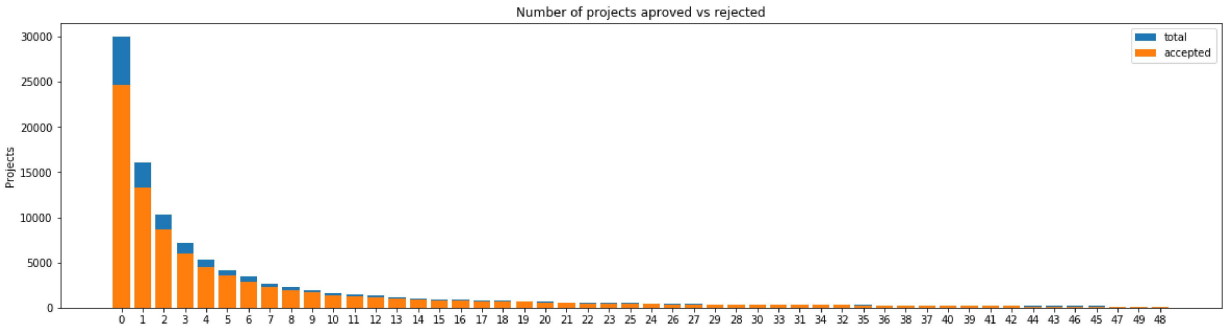
1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

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

In [41]:
```
project_data['teacher_number_of_previously_posted_projects'].head(5)
```

Out[41]:
```
0    0
1    7
2    1
3    4
4    1
Name: teacher_number_of_previously_posted_projects, dtype: int64
```

In [42]:
```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects',
```




```
   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_approved  total  \
46                                            46                 149    164
45                                            45                 141    153
47                                            47                 129    144
49                                            49                 128    143
48                                            48                 135    140

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

Observation

1. Project Approval rate is increasing as the teacher number of previously posted projects increases.

2. Number of projects decline exponentially as the number of previously posted projects increases.

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 [43]:
```
project_data['project_resource_summary'].head(5)
```

Out[43]:
```
0    My students need opportunities to practice beg...
1    My students need a projector to help with view...
2    My students need shine guards, athletic socks,...
3    My students need to engage in Reading and Math...
4    My students need hands on practice in mathemat...
Name: project_resource_summary, dtype: object
```

In [44]:
```
df = pd.DataFrame()
ssm = pd.DataFrame()
def hasNumbers(inputString):
    return any(char.isdigit() for char in inputString)



df['project_is_approved'] = project_data['project_is_approved']
df['has_numerical'] = project_data['project_resource_summary'].apply(hasNumbers)
```

In [45]:
```
has_numerical = df[df['has_numerical']==True]
has_numerical_approved = has_numerical[df['project_is_approved']==1].count()
has_numerical_not_approved = has_numerical[df['project_is_approved']==0].count()

has_no_numerical = df[df['has_numerical']==False]
has_no_numerical_approved = has_no_numerical[df['project_is_approved']==1].count(
has_no_numerical_not_approved = has_no_numerical[df['project_is_approved']==0].co

has_numerical_approved = has_numerical_approved.has_numerical
has_no_numerical_approved = has_no_numerical_approved.has_numerical

print(' ',has_numerical_approved,'\n ' ,has_no_numerical_approved)
```

```
 14090
 78616
```

Observation

The number of approvals in case of project resource summary that contains numerical digits is less and Rejection Rates are Higher

1.3 Text preprocessing

1.3.1 Essay Text

In [46]:
```
project_data.head(2)
```

Out[46]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [47]:
```
# 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 their second or
third languages. We are a melting pot of refugees, immigrants, and native-born
Americans bringing the gift of language to our school. \r\n\r\n We have over 24
languages represented in our English Learner program with students at every lev
el of mastery.  We also have over 40 countries represented with the families wi
thin our school.  Each student brings a wealth of knowledge and experiences to
us that open our eyes to new cultures, beliefs, and respect.\"The limits of you
r language are the limits of your world.\"-Ludwig Wittgenstein  Our English lea
rner's have a strong support system at home that begs for more resources.  Many
times our parents are learning to read and speak English along side of their ch
ildren.  Sometimes this creates barriers for parents to be able to help their c
hild learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy p
roviding these dvd's and players, students are able to continue their mastery o
f the English language even if no one at home is able to assist.  All families
with students within the Level 1 proficiency status, will be a offered to be a
part of this program.  These educational videos will be specially chosen by the
English Learner Teacher and will be sent home regularly to watch.  The videos a
re to help the child develop early reading skills.\r\n\r\nParents that do not h
ave 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 educational dvd's for
the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this 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 560 students, 97.3% are minority st
udents. \r\nThe school has a vibrant community that loves to get together and c
elebrate. Around Halloween there is a whole school parade to show off the beaut
iful costumes that students wear. On Cinco de Mayo we put on a big festival wit
h crafts made by the students, dances, and games. At the end of the year the sc
hool hosts a carnival to celebrate the hard work put in during the school year,
with a dunk tank being the most popular activity.My students will use these fiv
e brightly colored Hokki stools in place of regular, stationary, 4-legged chair
s. As I will only have a total of ten in the classroom and not enough for each
student to have an individual one, they will be used in a variety of ways. Duri
ng independent reading time they will be used as special chairs students will e
ach use on occasion. I will utilize them in place of chairs at my small group t
ables during math and reading times. The rest of the day they will be used by t
he students who need the highest amount of movement in their life in order to s
tay 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 stool
s we already have. When the students are sitting in group with me on the Hokki
Stools, they are always moving, but at the same time doing their work. Anytime
the students get to pick where they can sit, the Hokki Stools are the first to
be taken. There are always students who head over to the kidney table to get on
```

e of the stools who are disappointed as there are not enough of them. \r\n\r\nW
e ask a lot of students to sit for 7 hours a day. The Hokki stools will be a co
mpromise that allow my students to do desk work and move at the same time. Thes
e stools will help students to meet their 60 minutes a day of movement by allow
ing them to activate their core muscles for balance while they sit. For many of
my students, these chairs will take away the barrier that exists in schools for
a child who can't sit still.nannan
==========
How do you remember your days of school? Was it in a sterile environment with p
lain 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 roo
m for my students look forward to coming to each day.\r\n\r\nMy class is made u
p of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey a
ttend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept,
which is very unique as there are no walls separating the classrooms. These 9 a
nd 10 year-old students are very eager learners; they are like sponges, absorbi
ng all the information and experiences and keep on wanting more.With these reso
urces such as the comfy red throw pillows and the whimsical nautical hanging de
cor and the blue fish nets, I will be able to help create the mood in our class
room setting to be one of a themed nautical environment. Creating a classroom e
nvironment is very important in the success in each and every child's educatio
n. The nautical photo props will be used with each child as they step foot into
our classroom for the first time on Meet the Teacher evening. I'll take picture
s of each child with them, have them developed, and then hung in our classroom
ready for their first day of 4th grade. This kind gesture will set the tone be
fore even the first day of school! The nautical thank you cards will be used th
roughout the year by the students as they create thank you cards to their team
groups.\r\n\r\nYour generous donations will help me to help make our classroom
a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of mon
ey out of my own pocket on resources to get our classroom ready. Please conside
r helping with this project to make our new school year a very successful one.
Thank you!nannan
==========
My kindergarten students have varied disabilities ranging from speech and langu
age delays, cognitive delays, gross/fine motor delays, to autism. They are eage
r beavers and always strive to work their hardest working past their limitation
s. \r\n\r\nThe materials we have are the ones I seek out for my students. I tea
ch in a Title I school where most of the students receive free or reduced price
lunch. Despite their 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 able to move as they learn or
so they say.Wobble chairs are the answer and I love then because they develop t
heir core, which enhances gross motor and in Turn fine motor skills. \r\nThey a
lso want to learn through games, my kids don't want to sit and do worksheets. T
hey want to learn to count by jumping and playing. Physical engagement is the k
ey to our success. The number toss and color and shape mats can make that happe
n. My students will forget they are doing work and just have the fun a 6 year o
ld deserves.nannan
==========
The mediocre teacher tells. The good teacher explains. The superior teacher dem
onstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school has 80
3 students which is makeup is 97.6% African-American, making up the largest seg
ment of the student body. A typical school in Dallas is made up of 23.2% Africa
n-American students. Most of the students are on free or reduced lunch. We are
n't receiving doctors, lawyers, or engineers children from rich backgrounds or

```
neighborhoods. As an educator I am inspiring minds of young children and we foc
us not only on academics but one smart, effective, efficient, and disciplined s
tudents with good character.In our classroom we can utilize the Bluetooth for s
wift transitions during class. I use a speaker which doesn't amplify the sound
enough to receive the message. Due to the volume of my speaker my students ca
n't hear videos or books clearly and it isn't making the lessons as meaningful.
But with the bluetooth speaker my students will be able to hear and I can stop,
pause and replay it at any time.\r\nThe cart will allow me to have more room fo
r storage of things that are needed for the day and has an extra part to it I c
an use.  The table top chart has all of the letter, words and pictures for stud
ents to learn about different letters and it is more accessible.nannan
==================================================
```

In [48]:
```
# 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 [49]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and langu
age delays, cognitive delays, gross/fine motor delays, to autism. They are eage
r beavers and always strive to work their hardest working past their limitation
s. \r\n\r\nThe materials we have are the ones I seek out for my students. I tea
ch in a Title I school where most of the students receive free or reduced price
lunch.  Despite their 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 able to move as they learn or
so they say.Wobble chairs are the answer and I love then because they develop t
heir core, which enhances gross motor and in Turn fine motor skills. \r\nThey a
lso 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. The number toss and color and shape mats can make that happ
en. My students will forget they are doing work and just have the fun a 6 year
old deserves.nannan
==================================================
```

In [50]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-bre
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and langu
age delays, cognitive delays, gross/fine motor delays, to autism. They are eage
r beavers and always strive to work their hardest working past their limitation
s.     The materials we have are the ones I seek out for my students. I teach i
n a Title I school where most of the students receive free or reduced price lun
ch.  Despite 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 answer and I love then because they develop thei
r core, which enhances gross motor and in Turn fine motor skills.   They also w
ant 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 t
o our success. The number toss and color and shape mats can make that happen. M
y students will forget they are doing work and just have the fun a 6 year old d
eserves.nannan
```

In [51]:
```
#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 speech and langu
age delays cognitive delays gross fine motor delays to autism They are eager be
avers and always strive to work their hardest working past their limitations Th
e materials we have are the ones I seek out for my students I teach in a Title
I school where most of the students receive free or reduced price lunch Despite
their disabilities and limitations my students love coming to school and come e
ager to learn and explore Have you ever felt like you had ants in your pants an
d you needed to groove and move as you were in a meeting This is how my kids fe
el 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 then because they develop their core which enh
ances 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 The number
toss and color and shape mats can make that happen My students will forget they
are doing work and just have the fun a 6 year old deserves nannan
```

In [52]:
```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', '
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itsel
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because'
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'th
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all'
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than',
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "di
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn'
            'won', "won't", 'wouldn', "wouldn't"]
```

In [53]:
```
# 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 [02:52<00:00, 632.23it/s]
```

In [54]:
```
# after preprocesing
project_data.drop(['project_essay_1','project_essay_2', 'project_essay_3','projec
preprocessed_essays = pd.DataFrame(preprocessed_essays)
project_data['essay'] = preprocessed_essays
```

1.3.2 Project title Text

In [55]:
```
# similarly you can preprocess the titles also
project_data['project_title'].head(2)
```

Out[55]:
```
0    Educational Support for English Learners at Home
1                Wanted: Projector for Hungry Learners
Name: project_title, dtype: object
```

In [56]:
```
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
print(project_data['project_title'].values[10])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
print(project_data['project_title'].values[1500])
print("="*50)
print(project_data['project_title'].values[25000])
print("="*50)
print(project_data['project_title'].values[90999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
Reading Changes Lives
==================================================
21st Century learners, 21st century technology!
==================================================
Listening Center
==================================================
Moving To Learn and Creative Expression
==================================================
Wobble While We Learn
==================================================
```

In [57]:
```
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 [00:07<00:00, 14171.80it/s]
```

In [58]:
```
preprocessed_titles = pd.DataFrame(preprocessed_titles)
project_data['project_title'] = preprocessed_titles
project_data['project_title'].head(10)
```

Out[58]:
```
0            educational support english learners home
1                     wanted projector hungry learners
2      soccer equipment awesome middle school students
3                                techie kindergarteners
4                                interactive math tools
5    flexible seating mrs jarvis terrific third gra...
6        chromebooks special education reading program
7                                     it 21st century
8                          targeting more success class
9                  just for love reading pure pleasure
Name: project_title, dtype: object
```

In [59]:
```
prt = project_data[project_data['teacher_prefix']=='Dr.']
len(prt)
```

Out[59]:
```
13
```

1. 4 Preparing data for models

In [ ]:
```
print(project_data.columns)
#randomly sampling 10000 points
X_p = project_data[project_data['project_is_approved']==1].sample(n=5000)
X_n = project_data[project_data['project_is_approved']==0].sample(n=5000)
project_data = pd.concat([X_p,X_n])
```

we are going to consider

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

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

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

1.4.1 Vectorizing Categorical data

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

In [61]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=F
vectorizer.fit(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'S
pecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding  (10000, 9)
```

In [62]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowerca
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Ex
tracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation',
'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducatio
n', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography',
'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalS
cience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'L
iterature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodimng  (10000, 30)
```

In [63]:
```
# Please do the similar feature encoding with state, teacher_prefix and project_gr
# https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-pytho
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
values = np.array(project_data['school_state'])
print(values[1:10])
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
print(integer_encoded[1:10])

state_school_onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
state_school_onehot_encoded = state_school_onehot_encoder.fit_transform(integer_e

print("Shape of matrix after one hot encoding ",state_school_onehot_encoded.shape
```

```
['MD' 'NC' 'VA' 'CO' 'NC' 'FL' 'CA' 'TX' 'NM']
[20 27 45  5 27  9  4 43 32]
Shape of matrix after one hot encoding  (10000, 51)
```

In [64]:
```
# Please do the similar feature encoding with state, teacher_prefix and project_gr
# https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-pytho
#https://stackoverflow.com/questions/46406720/labelencoder-typeerror-not-supporte
values = (np.array(project_data['teacher_prefix']))
print(values[1:100])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:100])

# binary encode

teacher_prefix_onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
teacher_prefix_onehot_encoded = teacher_prefix_onehot_encoder.fit_transform(integ


print("Shape of matrix after one hot encoding ",teacher_prefix_onehot_encoded.sha
```

```
['Teacher' 'Mrs.' 'Ms.' 'Ms.' 'Ms.' 'Ms.' 'Mrs.' 'Mrs.' 'Mrs.' 'Ms.' 'Mrs.'
 'Mrs.' 'Ms.' 'Ms.' 'Ms.' 'Mrs.' 'Mrs.' 'Ms.' 'Mrs.' 'Mrs.' 'Mrs.' 'Ms.'
 'Mrs.' 'Mrs.' 'Mrs.' 'Mrs.' 'Mrs.' 'Mr.' 'Teacher' 'Ms.' 'Ms.' 'Ms.' 'Ms.'
 'Mrs.' 'Ms.' 'Mrs.' 'Mrs.' 'Ms.' 'Mrs.' 'Ms.' 'Mrs.' 'Mrs.' 'Mrs.' 'Mr.'
 'Mrs.' 'Ms.' 'Mrs.' 'Ms.' 'Mrs.' 'Mrs.' 'Ms.' 'Mrs.' 'Mrs.' 'Ms.' 'Mr.'
 'Mr.' 'Mrs.' 'Ms.' 'Ms.' 'Mrs.' 'Ms.' 'Mrs.' 'Mrs.' 'Ms.' 'Ms.' 'Mrs.'
 'Ms.' 'Mr.' 'Ms.' 'Mr.' 'Mrs.' 'Ms.' 'Mrs.' 'Mrs.' 'Ms.' 'Ms.' 'Mrs.'
 'Ms.' 'Mrs.' 'Mr.' 'Ms.' 'Mrs.' 'Ms.' 'Mr.' 'Mrs.' 'Mrs.' 'Ms.' 'Mrs.'
 'Mrs.' 'Mrs.' 'Mrs.' 'Mrs.' 'Mrs.' 'Ms.' 'Ms.' 'Mrs.' 'Mrs.' 'Mrs.' 'Ms.']
[4 2 3 3 3 3 2 2 2 3 2 2 3 3 3 2 2 3 2 2 2 3 2 2 2 2 2 1 4 3 3 3 3 2 3 2 2
 3 2 3 2 2 2 1 2 3 2 3 2 2 3 2 2 3 1 1 2 3 3 2 3 2 2 3 3 2 3 1 3 1 2 3 2 2
 3 3 2 3 2 1 3 2 3 1 2 2 3 2 2 2 2 2 2 3 3 2 2 2 3]
Shape of matrix after one hot encoding  (10000, 5)
```

In [65]:
```
values = (np.array(project_data['project_grade_category']))
print(values[1:10])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:10])

# binary encode

project_grade_category_onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
project_grade_category_onehot_encoder = project_grade_category_onehot_encoder.fit


print("Shape of matrix after one hot encoding ",project_grade_category_onehot_enc
```

```
['Grades PreK-2' 'Grades PreK-2' 'Grades PreK-2' 'Grades PreK-2'
 'Grades 6-8' 'Grades 3-5' 'Grades 6-8' 'Grades 6-8' 'Grades PreK-2']
[3 3 3 3 1 0 1 1 3]
Shape of matrix after one hot encoding  (10000, 4)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [66]:
```
# We are considering only the words which appeared in at least 10 documents(rows 
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(project_data['essay'])
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (10000, 6087)
```

1.4.2.2 Bag of Words on `project_title`

In [67]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer = CountVectorizer(min_df=1)
title_bow = vectorizer.fit_transform(project_data['project_title'])
print("Shape of matrix after one hot encodig ",title_bow.shape,type(title_bow))
```

```
Shape of matrix after one hot encodig  (10000, 5426) <class 'scipy.sparse.csr.c
sr_matrix'>
```

1.4.2.3 TFIDF vectorizer

In [68]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(project_data['essay'])
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (10000, 6087)
```

1.4.2.4 TFIDF Vectorizer on project_title

In [69]:
```
# Similarly you can vectorize for title also
vectorizer = TfidfVectorizer(min_df=1)
title_tfidf = vectorizer.fit_transform(project_data['project_title'])
print("Shape of matrix after one hot encodig ",title_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (10000, 5426)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [70]:
```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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

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

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

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


# stronging variables into pickle files python: http://www.jessicayung.com/how-to

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


'''
```

Out[70]: '\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084

```
039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGloveModel(glo
veFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', enco
ding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = lin
e.split()\n        word = splitLine[0]\n        embedding = np.array([float(va
l) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Don
e.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel(\'gl
ove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# =
============================\n\nwords = []\nfor i in preproced_texts:\n    word
s.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend(i.spli
t(\' \'))\nprint("all the words in the coupus", len(words))\nwords = set(words)
\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(mode
l.keys()).intersection(words)\nprint("The number of words that are present in b
oth glove vectors and our coupus",       len(inter_words),"(",np.round(len(inte
r_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(mode
l.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i]
 = model[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n# stronging va
riables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-
to-save-and-load-variables-in-python/\n\nimport (http://www.jessicayung.com/how
-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith op
en(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [71]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to
# 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 [72]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this li
for sentence in tqdm(project_data['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

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

```
100%|██████████| 10000/10000 [00:07<00:00, 1334.83it/s]

10000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [73]:
```
avg_w2v_vectors_titles = []
for sentence in tqdm(project_data['project_title']):
    vector_title = np.zeros(300)
    cnt_words =0;
    for word in sentence.split():
        if word in glove_words:
            vector_title += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector_title /= cnt_words
    avg_w2v_vectors_titles.append(vector_title)

print(type(avg_w2v_vectors_titles))
print(len(avg_w2v_vectors_titles[0]))
```

```
100%|
| 10000/10000 [00:00<00:00, 22606.87it/s]

<class 'list'>
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [74]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [75]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(project_data['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf valu
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

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

```
100%|
| 10000/10000 [00:55<00:00, 217.08it/s]

10000
300
```

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

In [76]:
```
# Similarly you can vectorize for title also
tfidf_w2v_vectors_title = [];
for sentence in tqdm(project_data['project_title']):
    vector = np.zeros(300)
    tf_idf_weight_title =0;
    for word in sentence.split():
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title.append(vector)

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

```
100%|██████████| 10000/10000 [00:00<00:00, 13511.96it/s]

10000
300
```

1.4.3 Vectorizing Numerical features

In [77]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklea
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.
# 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 a
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scala

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

```
Mean : 328.47879399999994, Standard deviation : 386.9349042096688
```

In [78]:
```
price_standardized
```

Out[78]:
```
array([[-0.01545168],
       [-0.79628069],
       [-0.52928488],
       ...,
       [ 0.07122957],
       [ 0.1509329 ],
       [ 0.26402686]])
```

1.4.4 Merging all the above features

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

In [79]:
```
print(type(categories_one_hot))
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
<class 'scipy.sparse.csr.csr_matrix'>
(10000, 30)
(10000, 6087)
(10000, 1)
```

In [80]:
```

approval = project_data['project_is_approved']
approval.shape
```

Out[80]: (10000,)

In [84]:
```
# 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 m
X = hstack((state_school_onehot_encoded, categories_one_hot, sub_categories_one_h

X1 = hstack((state_school_onehot_encoded, categories_one_hot, sub_categories_one_
X2 = hstack((state_school_onehot_encoded, categories_one_hot, sub_categories_one_
X3 = hstack((state_school_onehot_encoded, categories_one_hot, sub_categories_one_
X4 = hstack((state_school_onehot_encoded, categories_one_hot, sub_categories_one_
```

In [85]:
```
type(X1)
```

Out[85]: 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

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

In [86]:
```
# please write all of the code with proper documentation and proper titles for ea
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
#https://www.kaggle.com/premvardhan/amazon-fine-food-review-tsne-visualization

type(X1)
```

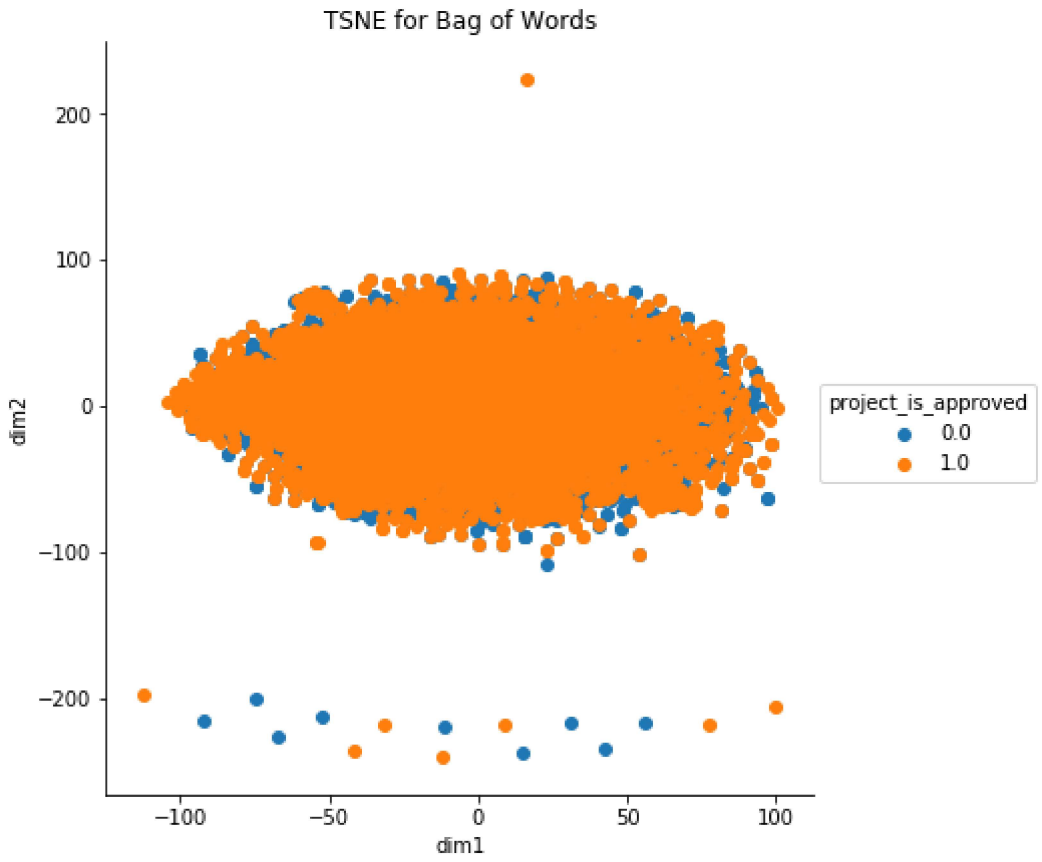
Out[86]:
```
scipy.sparse.coo.coo_matrix
```

In [87]:
```
X1 = X1.todense()
X1.shape
```

Out[87]:
```
(10000, 11613)
```

In [88]:
```
#https://www.kaggle.com/premvardhan/amazon-fine-food-review-tsne-visualization
# 4000 points are used in this tsne plot.
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 30, n_iter = 5000)

# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(X1)
print(tsne_data.shape)

# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, approval)).T
tsne_df = pd.DataFrame(data = tsne_data, columns=("dim1", "dim2", "project_is_app

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'dim1'
plt.title("TSNE for Bag of Words")
plt.show()
```

```
(10000, 2)
```



Observations

Points are not separated well and therefore it is hard to find does a particular project get approval or not.

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

In [89]:
```
# please write all the code with proper documentation, and proper titles for each
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

type(X2)
```

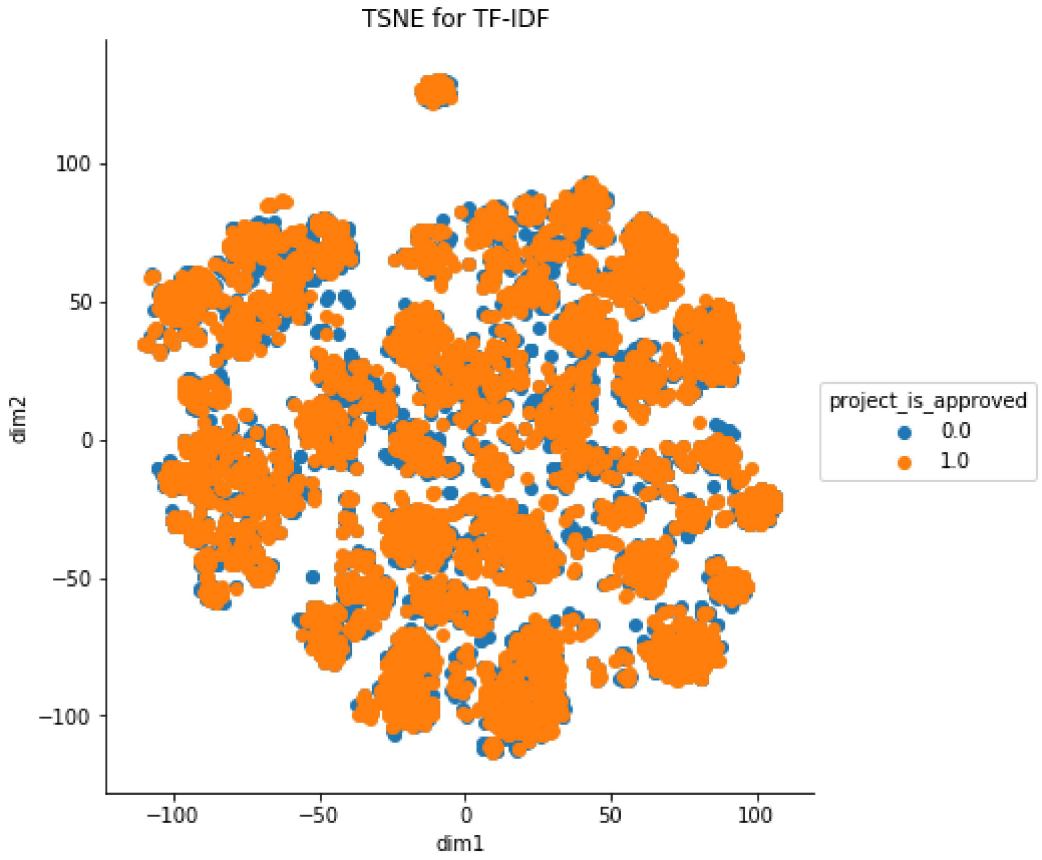
Out[89]: scipy.sparse.coo.coo_matrix

In [90]:
```
X2 = X2.todense()
X2.shape
```

Out[90]: (10000, 11613)

In [91]:
```
#https://www.kaggle.com/premvardhan/amazon-fine-food-review-tsne-visualization
# 4000 points are used in this tsne plot.
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 50, n_iter = 5000)

# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(X2)
print(tsne_data.shape)

# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, approval)).T
tsne_df = pd.DataFrame(data = tsne_data, columns=("dim1", "dim2", "project_is_app

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'dim1'
plt.title("TSNE for TF-IDF ")
plt.show()
```

```
(10000, 2)
```



Observations

Points are not separated well and therefore it is hard to find does a particular project get approval or not.

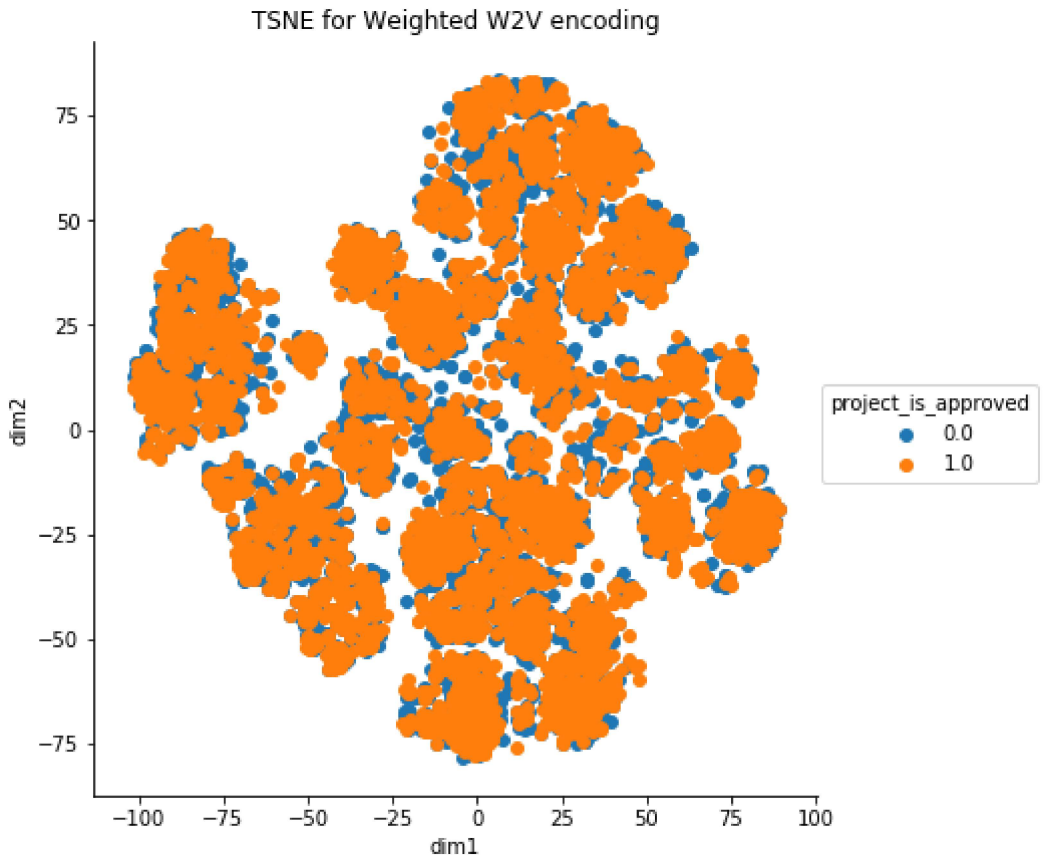
2.3 TSNE with `Weighted W2V` encoding of `project_title` feature

In [92]:
```
X3 = X3.todense()
X3.shape
```

Out[92]: (10000, 700)

In [93]:
```
# please write all the code with proper documentation, and proper titles for each
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

    #https://www.kaggle.com/premvardhan/amazon-fine-food-review-tsne-visualizatio
# 4000 points are used in this tsne plot.
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 75, n_iter = 5000)

# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(X3)
print(tsne_data.shape)

# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, approval)).T
tsne_df = pd.DataFrame(data = tsne_data, columns=("dim1", "dim2", "project_is_app

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'dim1'
plt.title("TSNE for Weighted W2V encoding")
plt.show()
```

```
(10000, 2)
```



Observations

Points are not well separated well and therefore by using Weighted word to vec it is hard to find does a particular project get approval or no

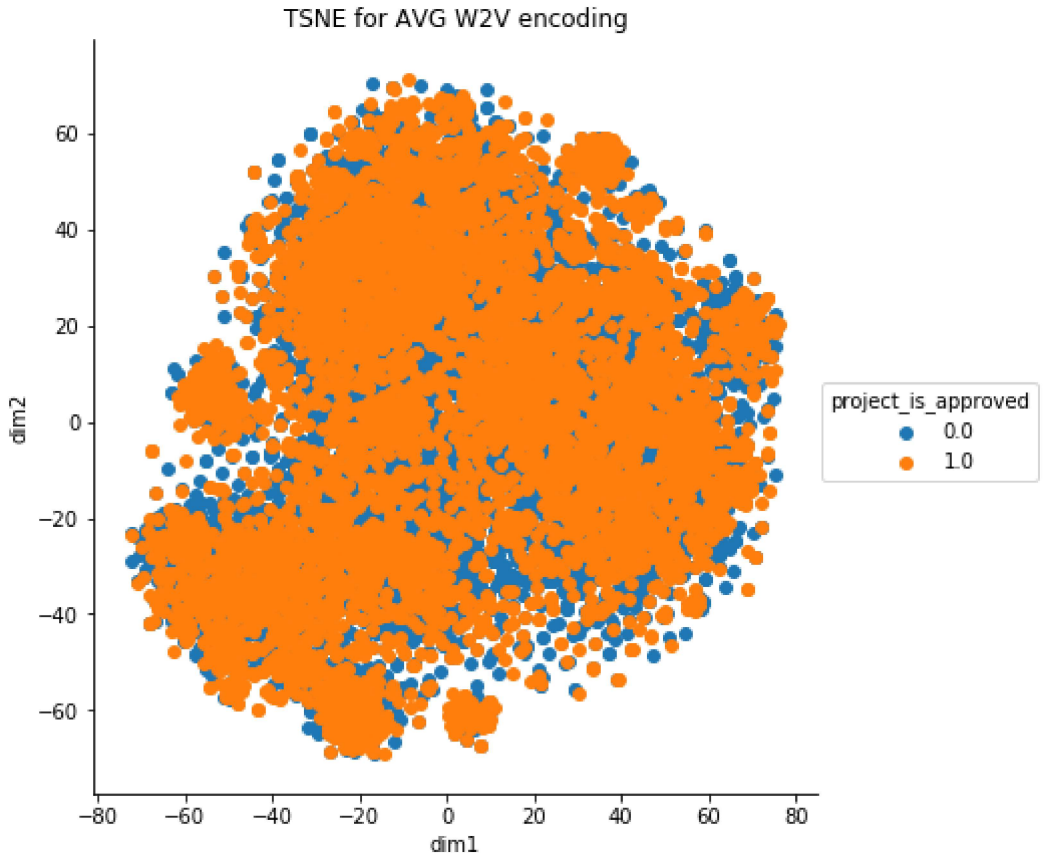
2.4 TSNE with `TFIDF Average W2V` encoding of `project_title` feature

In [94]:
```
X4 = X4.todense()
X4.shape
```

Out[94]:
```
(10000, 700)
```

In [95]:
```
# please write all the code with proper documentation, and proper titles for each
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

    #https://www.kaggle.com/premvardhan/amazon-fine-food-review-tsne-visualizatio
# 4000 points are used in this tsne plot.
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 70, n_iter = 5000)

# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(X4)
print(tsne_data.shape)

# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, approval)).T
tsne_df = pd.DataFrame(data = tsne_data, columns=("dim1", "dim2", "project_is_app

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'dim1'
plt.title("TSNE for AVG W2V encoding")
plt.show()
```

```
(10000, 2)
```



Observations

Points are not separated well and therefore it is hard to find does a particular project get approval or not.

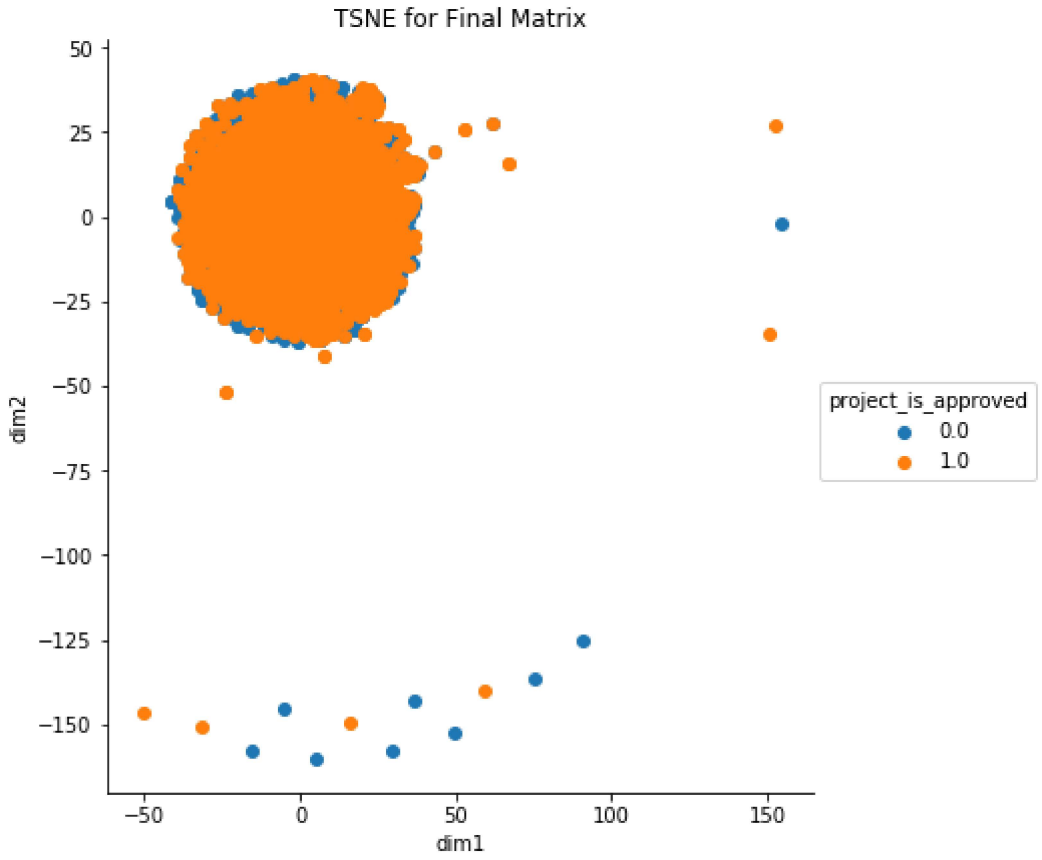
2.4 TSNE with Final Matrix

In [96]:
```
X = X.todense()
X.shape
```

Out[96]:
```
(10000, 24326)
```

In [97]:
```
#https://www.kaggle.com/premvardhan/amazon-fine-food-review-tsne-visualization
# 4000 points are used in this tsne plot.
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 50, n_iter = 5000)

# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(X)
print(tsne_data.shape)

# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, approval)).T
tsne_df = pd.DataFrame(data = tsne_data, columns=("dim1", "dim2", "project_is_app

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'dim1'
plt.title("TSNE for Final Matrix")
plt.show()
```

```
(10000, 2)
```



Observations

Points are not separated well and therefore it is hard to find does a paerticular point get approval or not.

2.5 Summary

1. For the Number of Projects

   Number of projects thar are approved for funding 92706 , (84.8583040422 %)
   Number of projects thar are not approved for funding 16542 , (15.1416959578 %)
2. clean category with Maximum project approval : Warmth Care_Hunger
   clean category with Minimum project approval : AppliedLearning Math_Science
3. The number of approvals in case of project resource summary that contains numerical digits is less and Rejection Rates are Higher.