DONORS CHOOSE (EDA-TSNE)

Objective: To predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved.

1.1 Importing the packages :

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
```

1.2 Reading the data :

In [2]:

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

Project data (train.csv)

In [3]:

```
#Printing shape and columns of project_data
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' 's
chool_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

```
#printing the first and last date, uptill which the data is in project_data
print(project_data.project_submitted_datetime.min())
print(project_data.project_submitted_datetime.max())
```

```
2016-04-27 00:27:36
2017-04-30 23:45:08
```

In [5]:

```
project_data.shape
```

Out[5]:

```
(109248, 17)
```

Note:

1. The project data contains 109248 entries, data from 2016-04-27 to 2017-04-30.
2. The project data has 17 attributes regarding project.

Resource Data (resource.csv)

In [6]:

```
#Printing the shape and columns of Resource data
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[6]:

	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

Note:

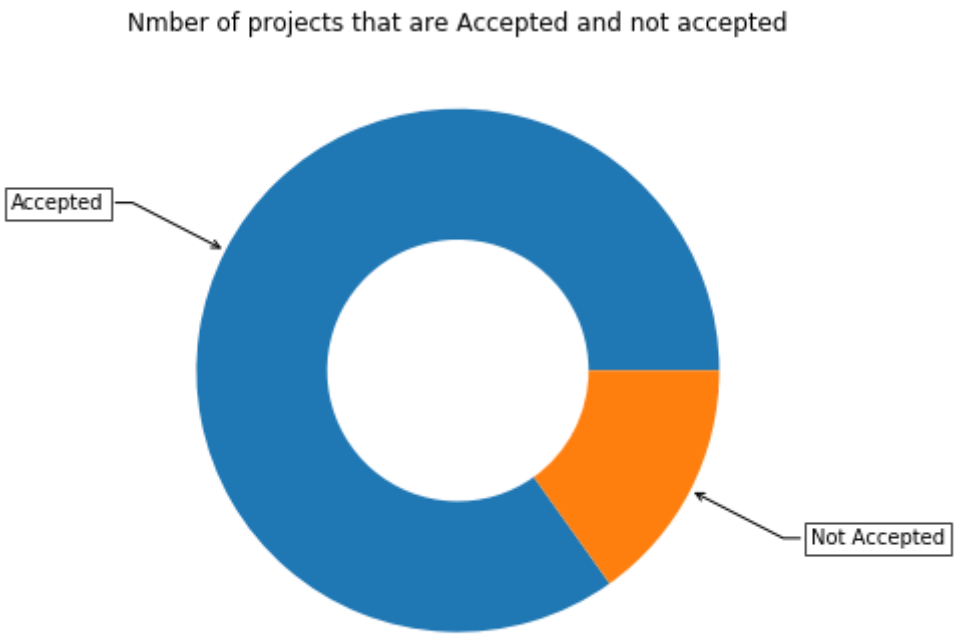
1. resource_data had contained 12344 duplicate entries.
2. Some projects has requested for multiple number of resource (number of project < number of resources)
3. The Resource data has 1541272 entries that contians the description, quantity and price of resource(s) required by the project.

1.3 Data Anaysis

In [7]:

```
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-glr-ga

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

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=0)

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

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

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

plt.show()
```

```
Number of projects thar are approved for funding  92706 , ( 84.8583040421792
7 %)
Number of projects thar are not approved for funding  16542 , ( 15.141695957
820739 %)
```



Note:

1. project_data has more data of the projects approved, hence it is an imbalance dataset.
2. Approximately 85% of the projects are approved for funding.
3. Nearly 15% of projects are NOT approved for funding.

1.4 Univariate Analysis

1.4.1 School State

In [8]:

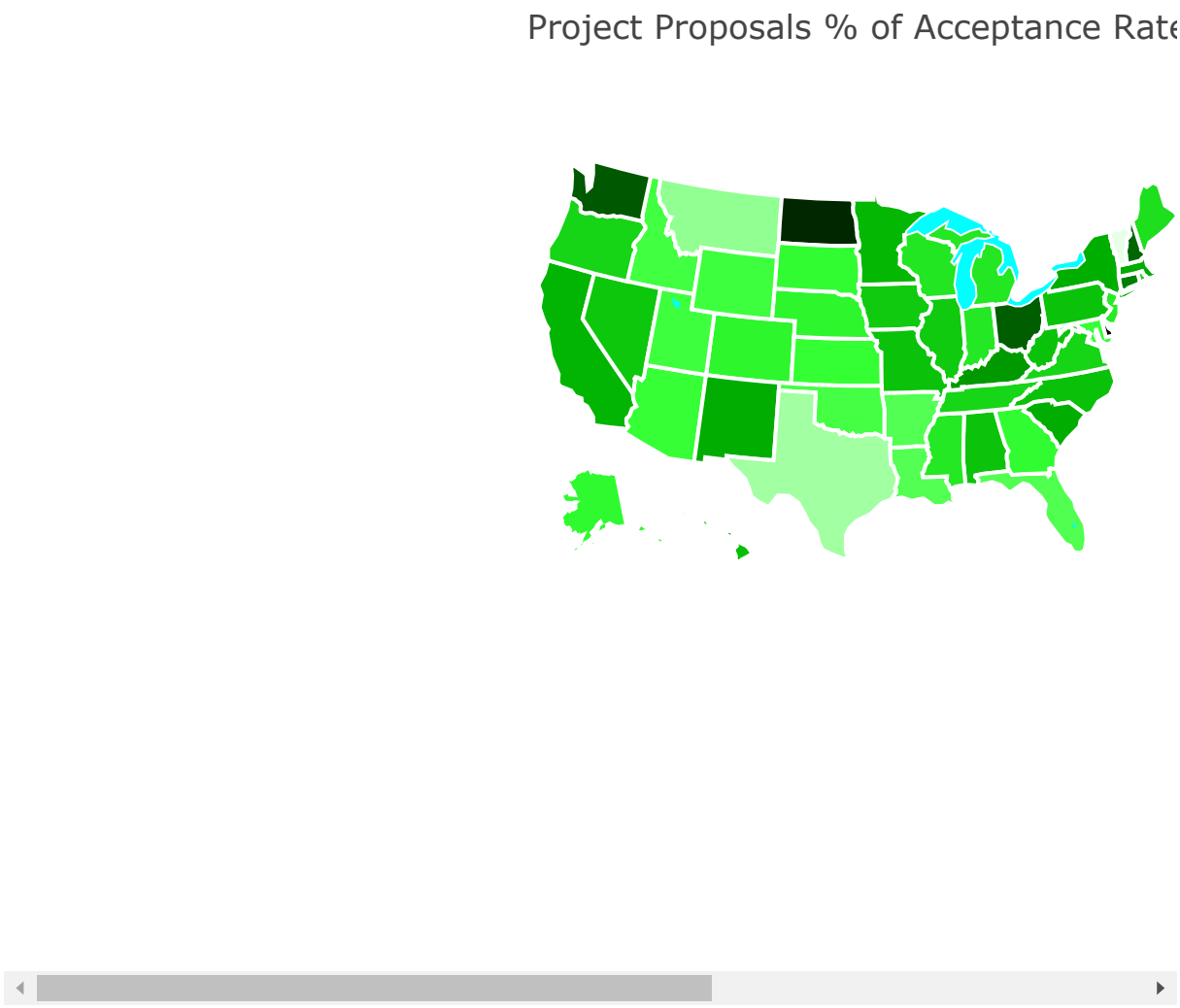
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].apply(np.me
temp.columns = ['state_code', 'num_proposals']
```

In [9]:

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

scl = [[0.0, 'rgb(230, 255, 230)'],[0.2, 'rgb(128, 255, 128)'],[0.4, 'rgb(51, 255, 51)'],\
            [0.6, 'rgb(0, 179, 0)'],[0.8, 'rgb(0, 77, 0)'],[1.0, 'rgb(0, 0, 0)']]

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(0, 255, 255)',
        ),
    )
```

In [10]:

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



In [11]:

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

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

Note:

1. Vermont, District of Columbia, Texas, Montana and Louisiana are the states with least percentage of project approvals ranging 80%-83%.
2. New Hampshire, Ohio, Washington, North Dakota and Delaware are the stated with high percentage of project approvals ranging 87%-90% approx.
3. The average approval rate is 85%.

Function for Stack Plot

In [12]:

```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/bar_st
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()
```

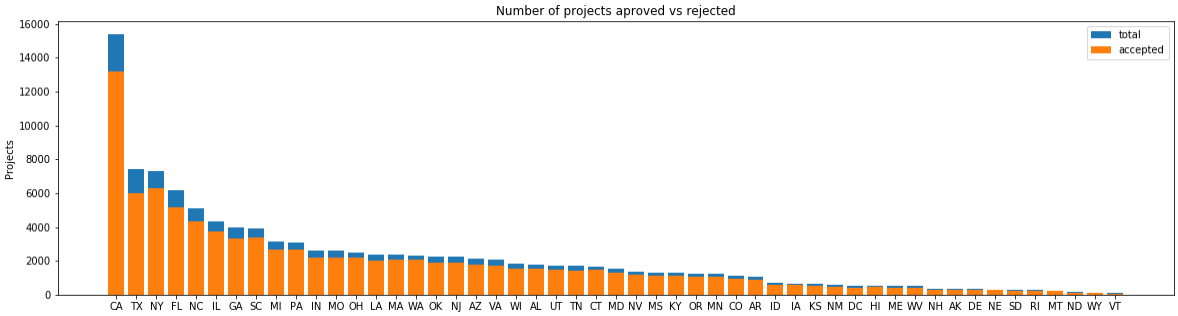
Function for Univariate Bar plot

In [13]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/40840
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).rese

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

    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 [14]:

```
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
```

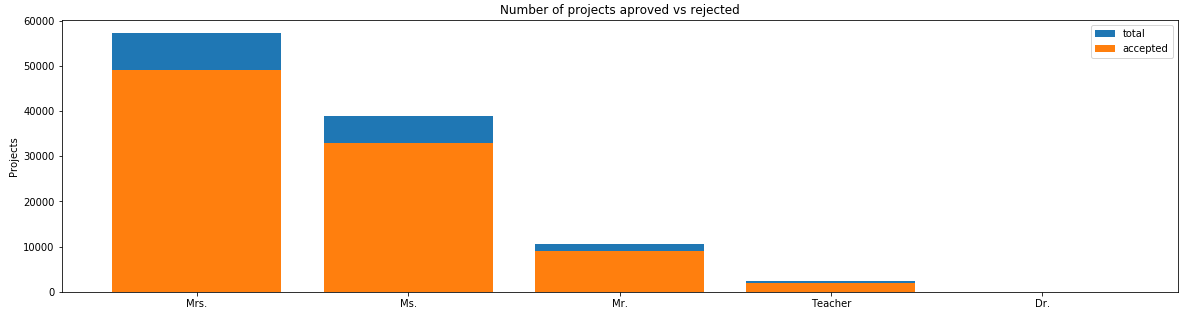
Note:

1. California has the highest total number of projects with the approval percentage of ~86%.
2. North Dakota has relatively very less total number of projects with the highest approval percentage of ~89%.
3. Every state has greater than 80% of project approval rate.

1.4.2 teacher_prefix

In [15]:

```
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
```

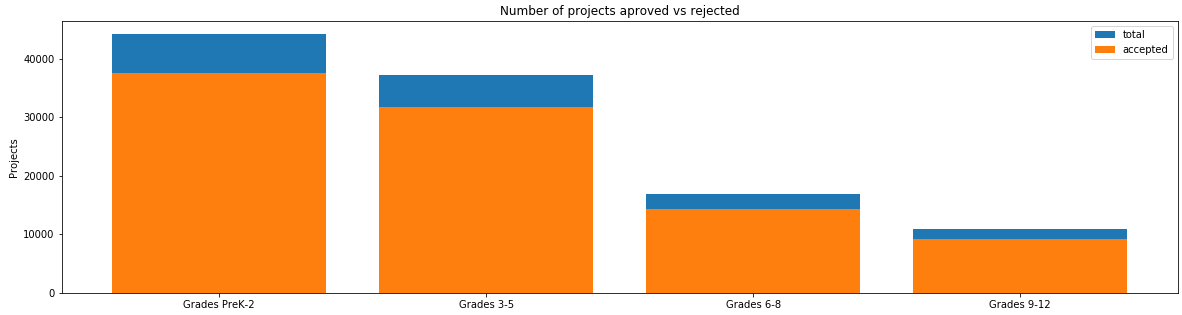
Note:

1. The total number of projects proposed by Teacher_prefixes (Mrs. and Ms.) are reltively much higher than others.
2. The teacher_prefix(Mrs) has the highest success rate of 85.55%.
3. The total number of projects proposed and the success rate(i.e 69%) is least with the teacher_prefix(Dr.).

1.4.3 project_grade_category

In [16]:

```
#univariate bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/bar

univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', top=Fals
```




```
  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
```

Note:

1. Project approval rate for grades 3-5 is highest(~85%) and least for grades 9-12(84%).
2. There is no siginficant difference between the success rates of different grade catgories, the success rate range from 83.7% to 85.4%..
3. The total number of projects for grades 9-12 is relativey much lesser than others.

1.4.4 project_subject_categories

In [17]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/473019
# 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

catogories = list(project_data['project_subject_categories'].values)
cat_list = []
for i in catogories:
    temp = ""
    for j in i.split(','):
        if 'The' in j.split():
            j=j.replace('The','')
        j = j.replace(' ','')
        temp+=j.strip()+" "
        temp = temp.replace('&','_')
    cat_list.append(temp.strip())
```

In [18]:

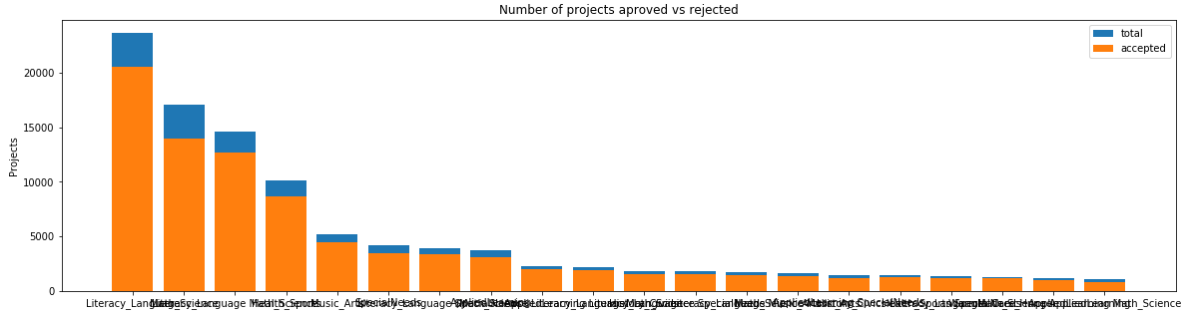
```
#adding clean_categories and dropping project_subject_categories
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[18]:

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

In [19]:

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



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

==========

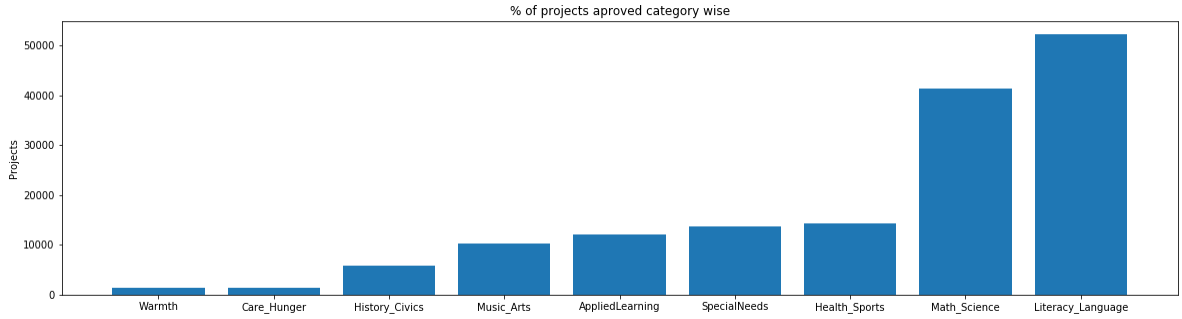
	clean_categories	project_is_approved	total	Avg
19	History_Civics Literacy_Language	1271	1421	0.894441
14	Health_Sports SpecialNeeds	1215	1391	0.873472
50	Warmth Care_Hunger	1212	1309	0.925898
33	Math_Science AppliedLearning	1019	1220	0.835246
4	AppliedLearning Math_Science	855	1052	0.812738

In [20]:

```
# 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 [21]:

```
# 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 [22]:

```
#categories in incresing order of approval rate
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
```

Note:

1. The projects of categories 'Literacy and Language' and 'Maths and Science' are higher in number.
2. The project of category 'Warmth Care and Hunger' and 'History and civis' has the least approval rate.
3. Each project category has greater than 80% of acceptance rate.

1.4.5 project_subject_subcategories

In [23]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/473019
# 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_catogories = list(project_data['project_subject_subcategories'].values)
sub_cat_list = []
for i in sub_catogories:
    temp = ""

    for j in i.split(','):
        if 'The' in j.split():
            j=j.replace('The','')
        j = j.replace(' ','')
        temp +=j.strip()+" "
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [24]:

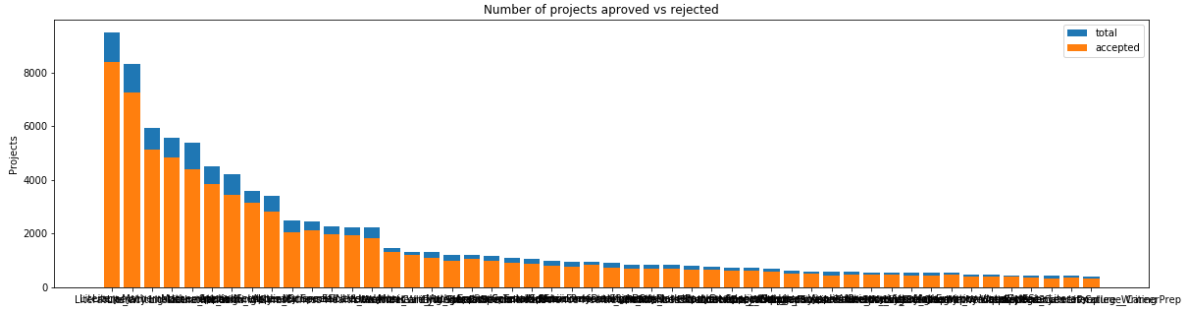
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[24]:

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

In [25]:

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



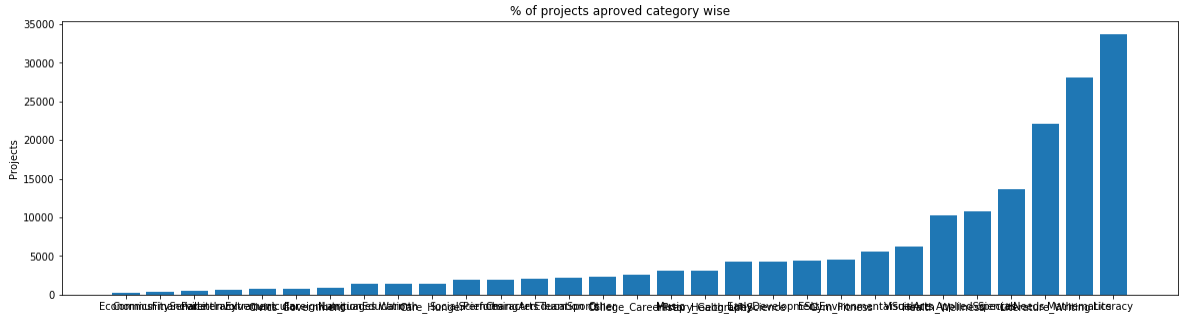

```
                clean_subcategories  project_is_approved  total       Avg
317                         Literacy                 8371   9486  0.882458
319              Literacy Mathematics                7260   8325  0.872072
331  Literature_Writing Mathematics                  5140   5923  0.867803
318     Literacy Literature_Writing                  4823   5571  0.865733
342                      Mathematics                 4385   5379  0.815207
==================================================
                    clean_subcategories  project_is_approved  total       Av
g
196       EnvironmentalScience Literacy                  389    444  0.87612
6
127                                ESL                  349    421  0.82897
9
79                  College_CareerPrep                  343    421  0.81472
7
17   AppliedSciences Literature_Writing                 361    420  0.85952
4
3    AppliedSciences College_CareerPrep                 330    405  0.81481
5
```

In [26]:

```
# 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 [27]:

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



In [28]:

```
# sub_categories in increasing order of approval rate
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
```

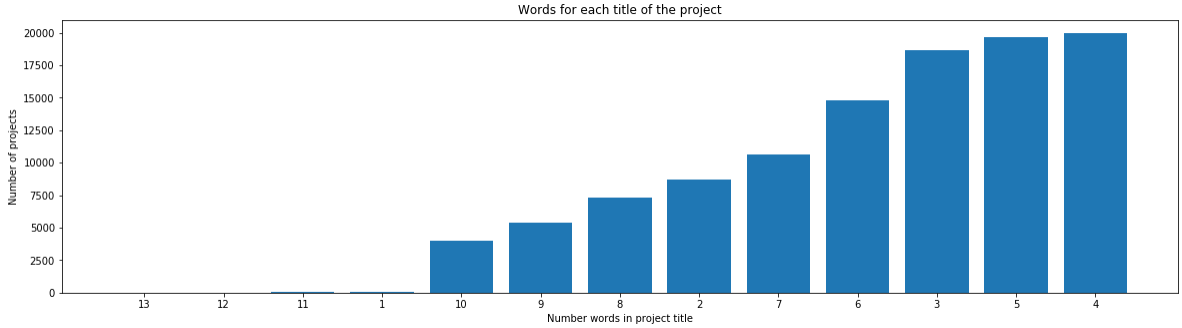
Note:

1. The project of Sub-category,'Literacy'has accepted projects higher in number.

1.4.6 Text features (Title)

In [29]:

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/374

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



Note:

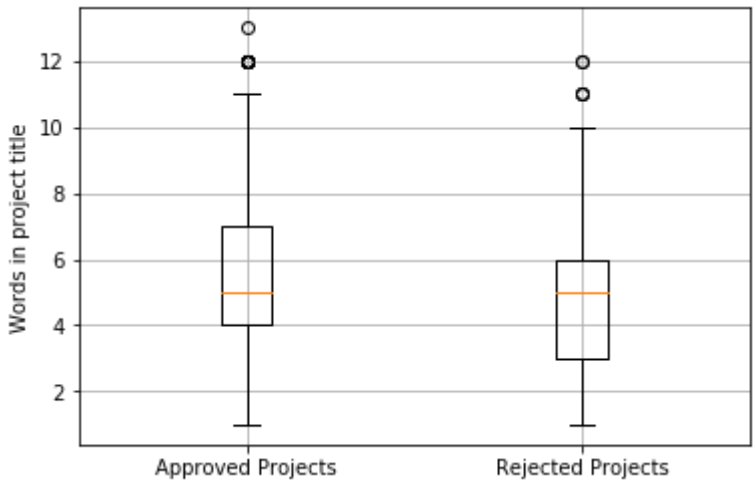
1. Most of the project titles are of 3-6 words that means most of the titles justified project motives in 3-6 words.
2. Very few of the project titles are of 1 word or 11 words.

In [30]:

```
approved_title_word_count = project_data[project_data['project_is_approved']==1]['project_t
approved_title_word_count = approved_title_word_count.values

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

In [31]:

```
# 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()
```

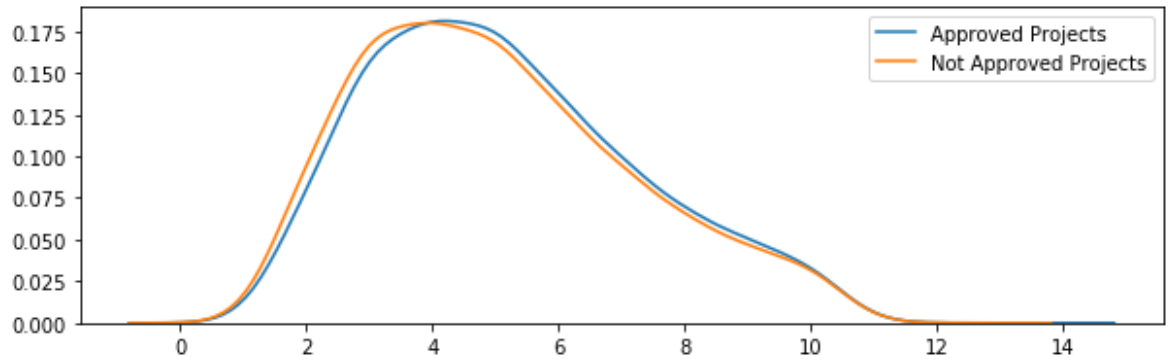


Note:

1. Most of the projects that got approved has 6 or more word in its title.
2. Both approved and non-approved projects has equal average number of words in their titles.

In [32]:

```
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()
```



Note:

1. This plot is not much informative, we cannot draw any strong conclusion from this plot.
2. approved_title_word_count is slightly greater than rejected_title_word_count.

1.4.7 Text features (Project Essay's)

In [33]:

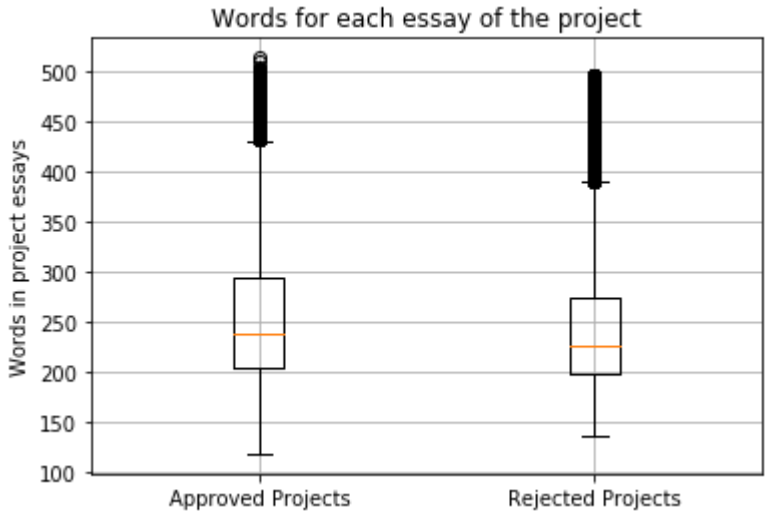
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [34]:

```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.spl
approved_word_count = approved_word_count.values

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

In [35]:

```
# 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()
```

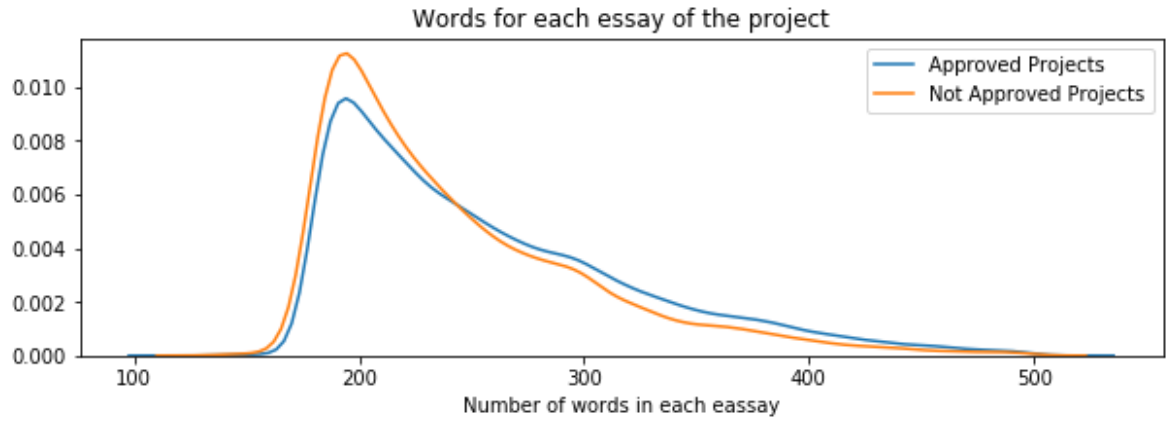


Note:

1. No strong conclusion can be drawn from the above plot.
2. Approved word count is slightly higher than rejected word count.

In [36]:

```
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()
```



Note:

1. Approved word count is bit higher than rejected word count.
2. Projects with higher number words in its essay are more likely to get approved.

1.4.8 Cost per project

In [37]:

```
# we get the cost of the project using resource.csv file
resource_data.head(5)
```

Out[37]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [38]:

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

Out[38]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [39]:

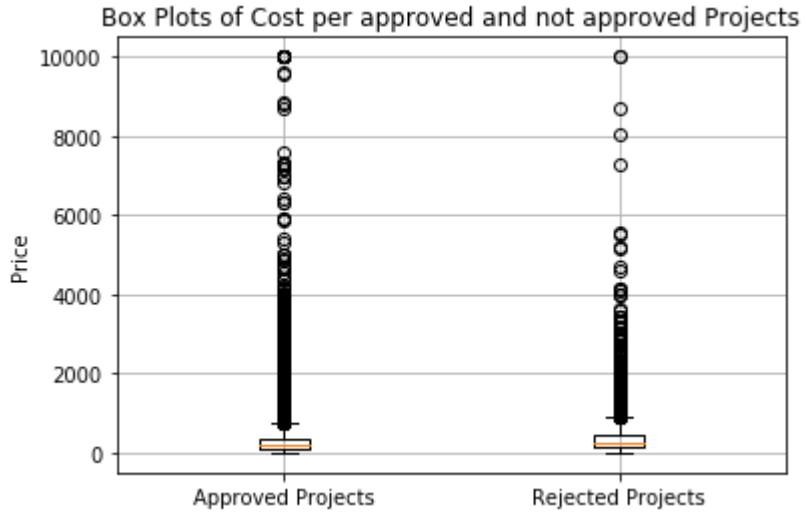
```
# join two dataframes in python: https://www.datacamp.com/community/tutorials/joining-dataf
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [40]:

```
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 [41]:

```
# 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()
```



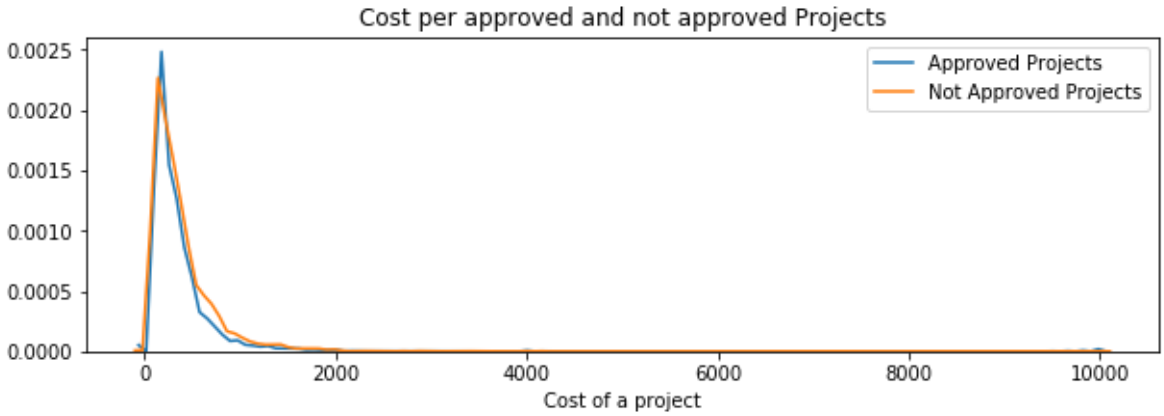
Note:

1. No strong conclusion can be drawn from the above plot.
2. Number of otliers are very high.

3. cost of project is not a good dimension to estimate the approval rate.

In [42]:

```
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()
```



Note:

1. No strong conclusion can be drawn from the above plot.
2. Cheaper projects are showing slightly higher approval rate.

In [43]:

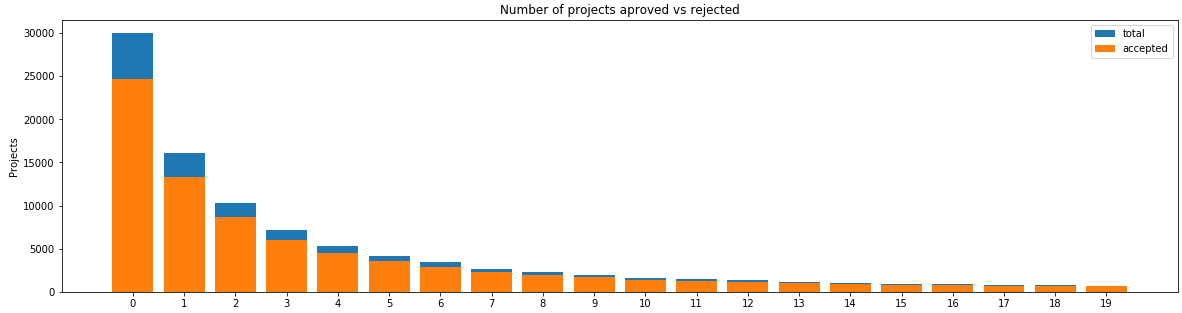
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

for i in range(0,101,5):
    x.add_row([i,np.round(np.percentile(approved_price,i), 3), np.round(np.percentile(rejec
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       |
+------------+------------------+----------------------+
```

1.4.9 teacher_number_of_previously_posted_projects (Assignment Part)

In [44]:

```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects', 'project_
```




```
   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
\
15                                            15                 818    942
16                                            16                 769    894
17                                            17                 712    803
18                                            18                 666    772
19                                            19                 632    710

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

Note:

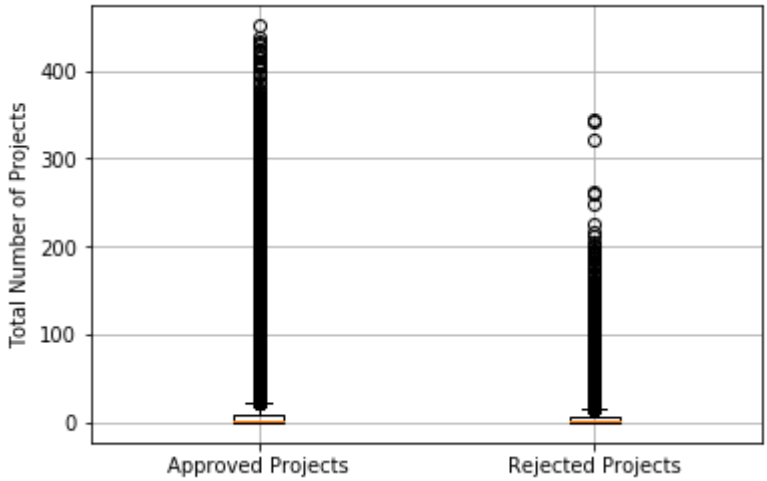
1. The approval rate is more that 80% for each number of previously posted projects by the teachers.
2. We can't say that the teacher with more number of previously posted projects has higher approval rate although the teacher which has previously posted 48 project has the approval rate of 96%.

In [45]:

```
approval = project_data[project_data['project_is_approved']==1]['teacher_number_of_previous
rejection = project_data[project_data['project_is_approved']==0]['teacher_number_of_previou
```

In [46]:

```
plt.boxplot([approval, rejection])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Total Number of Projects')
plt.grid()
plt.show()
```

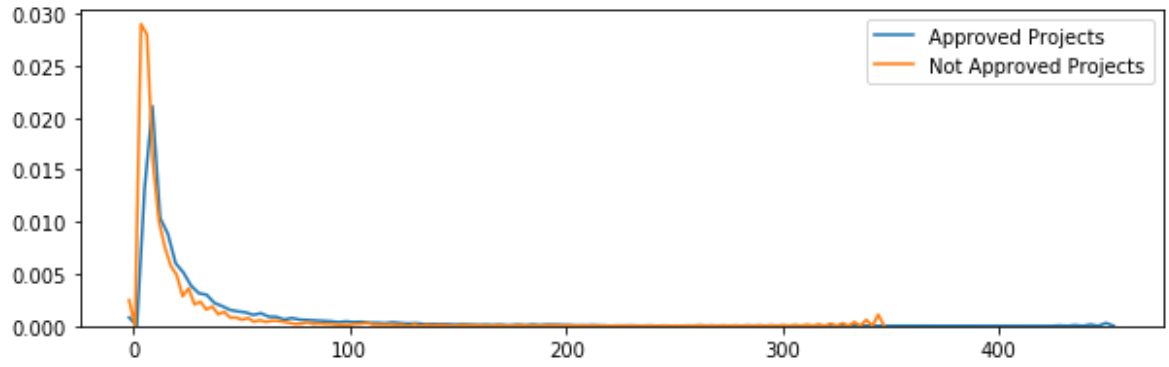


Note:

1. No strong conclusion can be drawn from the above plot.

In [47]:

```
plt.figure(figsize=(10,3))
sns.kdeplot(approval,label="Approved Projects", bw=0.6)
sns.kdeplot(rejection,label="Not Approved Projects", bw=0.6)
plt.legend()
plt.show()
```



Note:

1. At some extent there is a gradual increase in approval rate with increase of number of previously posted projects.

2. No strong conclusion can be drawn from the above plot.

1.4.10 project_resource_summary (Assignment Part)

In [48]:

```
#How to check if a string contains a number, https://stackoverflow.com/questions/19859282/c

import re
def hasNumbers(inputString):
    return bool(re.search(r'\d',inputString))

resource_summary=list(project_data['project_resource_summary'].values)
has_digits = []
for i in resource_summary:
    if (hasNumbers(i)==True):
        has_digits.append(1)
    else:
        has_digits.append(0)

project_data['summary_digits']=has_digits
```

In [49]:

```
project_data.head(5)
```

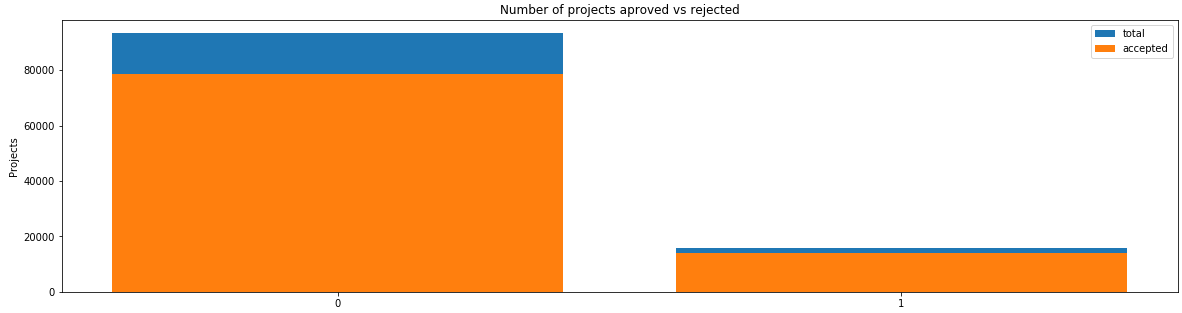
Out[49]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

5 rows × 21 columns

In [50]:

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




```
   summary_digits  project_is_approved  total       Avg
0               0                78616  93492  0.840885
1               1                14090  15756  0.894263
==================================================
   summary_digits  project_is_approved  total       Avg
0               0                78616  93492  0.840885
1               1                14090  15756  0.894263
```

Note:

1. Very few (14% approx) projects has digits in its resource summary.
2. The number of projects that contains digits in its resource summary has reatively higher approval rate than those doesn't have digits in its resource summary.
3. approval rates: (3.a) project resource that has digits: 89.4%, (3.b) project resource that doesn't have digits : 84%

1.5 Text Preprocessing

In [51]:

```
#Random Sampling of data, considering 5k random entries
# https: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Dataframe.smaple

approved_project = project_data[project_data['project_is_approved']==1].sample(n=5000, rand
rejected_project = project_data[project_data['project_is_approved']==0].sample(n=5000, rand
project_data = pd.concat([approved_project,rejected_project])
```

1.5.1 Essay Text

In [52]:

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

Out[52]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
35239	36406	p164226	18b42d3d4237f28ebe0145704b571ad3	Mrs.	CT	
66180	13120	p230144	83690eedd9ec0b0aef1d247a13fc9385	Ms.	NJ	

2 rows × 21 columns

In [53]:

```
# 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])
```

```
My students love to learn and I love to help them learn in a fun and excitin
g way! They love to be hands-on and are truly excited to come to school each
day and learn! They are curious, energized, and excited every day! My job, a
s their teacher, is to ensure that all students learn and foster their love
of learning! \r\n\r\nI teach in a small, rural community. We have a great sc
hool moral and students are always put first!My students need nonfiction boo
ks about cheetahs, polar bears, and dolphins to debunk the truth about these
animals from what's not true. Media literacy is super important to my studen
ts education, and I want them to realize the wealth of knowledge that a good
\"old fashioned\" book is to them. So many times my students believe EVERYTH
ING they see on the television or on the internet, that I want to teach and
remind them that reading and looking in a book is sometimes even more valuab
le for gaining the truth. These books will be used in our informational unit
on animals where students are required to become experts on an animal.nannan
==================================================
There is always anticipation and excitement on the first day of the new scho
ol year. Everything is new and exciting for my very young students. \r\n\r\n
My kindergarten students attend school in a very large urban district in Tex
as where resources are very limited.  \r\n\r\nMy very young students come to
school very  eager and excited to learn reading, writing, math and how to ge
t along with others. It is easy to remember that wonderful feeling of starti
ng school with new friends. \r\n\r\n\r\n\r\nThe ink and paper requested will
be used to create customized homework packets for my kindergarten students.
\r\n Each school year I make an effort to provide customized and differentia
ted homework for each of my students.  For example, when school starts, I pr
int their full names where they trace the dotted lines.  Once they have mast
ered this, then I hand write each of my students' full names on special writ
ing paper and use it as a \"master\" copy.  Each week I make copies from thi
s master copy as part of their homework.  As time passes, I am able to see m
ajor improvement in their printing skills.  In addition, I use the ink and p
aper to print words then sentences as part of their daily homework.  Math ho
mework sheets are also printed with these resources. \r\n Creating a customi
zed homework package takes a lot of planning, time, ink and paper!  The stic
ky wall pads will be used to model daily homework expectations.nannan
==================================================
Students enter class eager to learn. Technology engages young minds to make
learning exciting. “We need technology in every classroom and in every stude
nt and teacher’s hand, because it is the pen and paper of our time, and it i
s the lens through which we experience much of our world.” -D. Warlick\r\nTh
e students I teach come from different socioeconomic and ethnic background
s.\r\nI teach second graders at a Title One school. I am fortunate to have a
diverse group of second grade students. At our school, we have a high povert
y level. All of our students receive free or reduced-price lunch. Several st
udents receive resource and ELL services. One thing all my students have in
common is that they enjoy learning with technology. My students will use the
Chromebooks regularly. They will use them for math, language arts, science a
nd social studies. Students will work research topics with the Chromebooks a
nd use them to enhance their learning.  With the Chromebooks, the students w
ould be able to utilize more technology.These Chromebooks will help students
learn.  The Chromebooks will be used by my second grade students.  They will
use them as they work in collaborative groups.  Students will use them acros
```

```
s subject areas.  For example with social studies, students will use them to
research important historical figures like Dr. Martin Luther King Jr. and Ab
raham Lincoln.  Students will work in their groups to create digital present
ations on their historical figure to share with others.  In science, student
s will use the Chromebooks in collaborative groups to study weather pattern
s.  In reading, students will use the Chromebooks to listen to leveled books
and create digital retells.  In math, students will use the Chromebooks to f
ocus addition/subtraction practice.  These are only a few ways that the Chro
mebooks will be used in my classroom.nannan
```

In [54]:

```
# 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 [55]:

```
sent = decontracted(project_data['essay'].values[1000])
print(sent)
print("="*50)
```

```
Students enter class eager to learn. Technology engages young minds to make
learning exciting. “We need technology in every classroom and in every stude
nt and teacher’s hand, because it is the pen and paper of our time, and it i
s the lens through which we experience much of our world.” -D. Warlick\r\nTh
e students I teach come from different socioeconomic and ethnic background
s.\r\nI teach second graders at a Title One school. I am fortunate to have a
diverse group of second grade students. At our school, we have a high povert
y level. All of our students receive free or reduced-price lunch. Several st
udents receive resource and ELL services. One thing all my students have in
common is that they enjoy learning with technology. My students will use the
Chromebooks regularly. They will use them for math, language arts, science a
nd social studies. Students will work research topics with the Chromebooks a
nd use them to enhance their learning.  With the Chromebooks, the students w
ould be able to utilize more technology.These Chromebooks will help students
learn.  The Chromebooks will be used by my second grade students.  They will
use them as they work in collaborative groups.  Students will use them acros
s subject areas.  For example with social studies, students will use them to
research important historical figures like Dr. Martin Luther King Jr. and Ab
raham Lincoln.  Students will work in their groups to create digital present
ations on their historical figure to share with others.  In science, student
s will use the Chromebooks in collaborative groups to study weather pattern
s.  In reading, students will use the Chromebooks to listen to leveled books
and create digital retells.  In math, students will use the Chromebooks to f
ocus addition/subtraction practice.  These are only a few ways that the Chro
mebooks will be used in my classroom.nannan
==================================================
```

In [56]:

```
# \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)
```

```
Students enter class eager to learn. Technology engages young minds to make
learning exciting. “We need technology in every classroom and in every stude
nt and teacher’s hand, because it is the pen and paper of our time, and it i
s the lens through which we experience much of our world.” -D. Warlick  The
students I teach come from different socioeconomic and ethnic backgrounds.
I teach second graders at a Title One school. I am fortunate to have a diver
se group of second grade students. At our school, we have a high poverty lev
el. All of our students receive free or reduced-price lunch. Several student
s receive resource and ELL services. One thing all my students have in commo
n is that they enjoy learning with technology. My students will use the Chro
mebooks regularly. They will use them for math, language arts, science and s
ocial studies. Students will work research topics with the Chromebooks and u
se them to enhance their learning.  With the Chromebooks, the students would
be able to utilize more technology.These Chromebooks will help students lear
n.  The Chromebooks will be used by my second grade students.  They will use
them as they work in collaborative groups.  Students will use them across su
bject areas.  For example with social studies, students will use them to res
earch important historical figures like Dr. Martin Luther King Jr. and Abrah
am Lincoln.  Students will work in their groups to create digital presentati
ons on their historical figure to share with others.  In science, students w
ill use the Chromebooks in collaborative groups to study weather patterns.
In reading, students will use the Chromebooks to listen to leveled books and
create digital retells.  In math, students will use the Chromebooks to focus
addition/subtraction practice.  These are only a few ways that the Chromeboo
ks will be used in my classroom.nannan
```

In [57]:

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

```
Students enter class eager to learn Technology engages young minds to make l
earning exciting We need technology in every classroom and in every student
and teacher s hand because it is the pen and paper of our time and it is the
lens through which we experience much of our world D Warlick The students I
teach come from different socioeconomic and ethnic backgrounds I teach secon
d graders at a Title One school I am fortunate to have a diverse group of se
cond grade students At our school we have a high poverty level All of our st
udents receive free or reduced price lunch Several students receive resource
and ELL services One thing all my students have in common is that they enjoy
learning with technology My students will use the Chromebooks regularly They
will use them for math language arts science and social studies Students wil
l work research topics with the Chromebooks and use them to enhance their le
arning With the Chromebooks the students would be able to utilize more techn
ology These Chromebooks will help students learn The Chromebooks will be use
d by my second grade students They will use them as they work in collaborati
ve groups Students will use them across subject areas For example with socia
l studies students will use them to research important historical figures li
ke Dr Martin Luther King Jr and Abraham Lincoln Students will work in their
groups to create digital presentations on their historical figure to share w
ith others In science students will use the Chromebooks in collaborative gro
ups to study weather patterns In reading students will use the Chromebooks t
o listen to leveled books and create digital retells In math students will u
se the Chromebooks to focus addition subtraction practice These are only a f
ew ways that the Chromebooks will be used in my classroom nannan
```

In [58]:

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

In [59]:

```
# 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%|██████████| 10000/10000 [00:12<00:00, 785.86it/s]

In [60]:

```
# after preprocesing
preprocessed_essays[1000]
```

Out[60]:

```
'students enter class eager learn technology engages young minds make learni
ng exciting we need technology every classroom every student teacher hand pe
n paper time lens experience much world d warlick the students i teach come
different socioeconomic ethnic backgrounds i teach second graders title one
school i fortunate diverse group second grade students at school high povert
y level all students receive free reduced price lunch several students recei
ve resource ell services one thing students common enjoy learning technology
my students use chromebooks regularly they use math language arts science so
cial studies students work research topics chromebooks use enhance learning
with chromebooks students would able utilize technology these chromebooks he
lp students learn the chromebooks used second grade students they use work c
ollaborative groups students use across subject areas for example social stu
dies students use research important historical figures like dr martin luthe
r king jr abraham lincoln students work groups create digital presentations
historical figure share others in science students use chromebooks collabora
tive groups study weather patterns in reading students use chromebooks liste
n leveled books create digital retells in math students use chromebooks focu
s addition subtraction practice these ways chromebooks used classroom nanna
n'
```

1.5.2 Project title Text (Assignment Part)

In [61]:

```
# printing some random essays.
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])
```

```
Meow, Squeak, and Roar!
==================================================
Ink. Paper. RESULTS!
==================================================
Chromebooks for Learning
```

In [62]:

```
# Combining all the above statemennts
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%|██████████| 10000/10000 [00:00<00:00, 18376.95it/s]
```

In [63]:

```
# after preprocesing
print("{}".format(preprocessed_titles[150]))
```

```
ink paper results
```

1.6 Preparing data for models

In [64]:

```
project_data.columns
```

Out[64]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantit
y',
       'summary_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

1.6.1 Vectorizing Categorical data

1.6.1.1 Clean Categories

In [65]:

```
# 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, bina
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',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding  (10000, 9)
```

1.6.1.2 Sub-Clean-categories

In [66]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
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 encoding ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducat
ion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Characte
rEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_
Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness',
'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encoding  (10000, 30)
```

1.6.1.3 School State (Assignment Part)

In [67]:

```
from collections import Counter
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())

state_dict = dict(my_counter)

sorted_state_dict = dict(sorted(state_dict.items(),key=lambda kv: kv[1]))
vectorizer = CountVectorizer(vocabulary=list(sorted_state_dict.keys()), lowercase=False, bi
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())

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

```
['VT', 'WY', 'ND', 'DE', 'NE', 'MT', 'NH', 'RI', 'SD', 'WV', 'AK', 'NM', 'H
I', 'ME', 'DC', 'KS', 'IA', 'ID', 'CO', 'MN', 'KY', 'AR', 'NV', 'OR', 'MS',
'MD', 'CT', 'TN', 'AL', 'UT', 'WI', 'VA', 'OH', 'NJ', 'WA', 'MA', 'AZ', 'M
O', 'IN', 'OK', 'LA', 'MI', 'PA', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX',
'CA']
Shape of matrix after one hot encoding  (10000, 51)
```

1.6.1.4 Teacher Prefixes (Assignment Part)

In [68]:

```
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np
prefix = list(set(project_data['teacher_prefix'].values))
vectorizer = CountVectorizer(vocabulary=prefix,lowercase=False,binary=True)

vectorizer.fit(project_data['teacher_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())

prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].values.astype('U'))
print("Shape of matrix after one hot encoding ",prefix_one_hot.shape)
```

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

1.6.1.5 project_grade_category (Assignment Part)

In [69]:

```
prefix = list(set(project_data['project_grade_category'].values))
vectorizer = CountVectorizer(vocabulary=prefix,lowercase=False,binary=True)
vectorizer.fit(project_data['project_grade_category'].values.astype('U'))
print(vectorizer.get_feature_names())

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

```
['Grades 3-5', 'Grades 6-8', 'Grades 9-12', 'Grades PreK-2']
Shape of matrix after one hot encoding  (10000, 4)
```

1.7 Vectorizing Text data

1.7.1 Bow:Essay

In [70]:

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

```
Shape of matrix after one hot encoding  (10000, 6108)
```

1.7.2 Bow:Project_title (Assignment Part)

In [71]:

```
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encoding ",title_bow.shape)
```

```
Shape of matrix after one hot encoding  (10000, 664)
```

1.8 TFIDF vectorizer

1.8.1 TFIDF on essays

In [72]:

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

```
Shape of matrix after one hot encoding  (10000, 6108)
```

1.8.2 TFIDF on project title (Assignment Part)

In [73]:

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

```
Shape of matrix after one hot encoding  (10000, 664)
```

1.9 Using Pretrained Models: Avg W2V

In [74]:

```
import pickle
import itertools
from gensim.models.word2vec import Text8Corpus
from glove import Corpus, Glove
```

In [75]:

```
# 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')


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

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

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

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


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

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

```
Loading Glove Model

1917495it [09:54, 3225.40it/s]

Done. 1917495  words loaded!
all the words in the coupus 1526924
the unique words in the coupus 23068
The number of words that are present in both glove vectors and our coupus 22
089 ( 95.756 %)
word 2 vec length 22089
```

In [76]:

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

with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

1.9.2 AVG W2V on project_essay

In [77]:

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

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

```
100%|██████████| 10000/10000 [00:06<00:00, 1555.67it/s]

10000
300
```

1.9.2 AVG W2V on project_title (Assignment Part)

In [78]:

```
avg_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles.append(vector)

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

```
100%|██████████| 10000/10000 [00:06<00:00, 1696.72it/s]

10000
300
```

1.10 Using Pretrained Models: TFIDF weighted W2V

1.10.1 TFIDF Weighted W2V on project_essay

In [79]:

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

```
22348
```

In [80]:

```
tfidf_w2v_vectors = [];
for sentence in tqdm(preprocessed_essays):
    vector = np.zeros(300)
    tf_idf_weight=0;
    for word in sentence.split():
        try:
            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

        except:
            pass
    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 [00:47<00:00, 224.88it/s]

10000
300
```

1.10.1 TFIDF Weighted W2V on project_title

In [81]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
print(len(tfidf_words))     #To get the length of tfidf vector
```

```
5366
```

In [82]:

```
tfidf_w2v_vectors_titles = [];
for sentence in tqdm(preprocessed_titles):
    vector = np.zeros(300)
    tf_idf_weight=0;
    for word in sentence.split():
        try:
            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

        except:
            pass
    if tf_idf_weight !=0:
        vector /=tf_idf_weight
    tfidf_w2v_vectors_titles.append(vector)

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

```
100%|██████████| 10000/10000 [00:00<00:00, 14775.67it/s]

10000
300
```

1.11 Vectorizing Numerical features

Standardizing price

In [83]:

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

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

```
Mean : 323.177644, Standard deviation : 352.44291350615816
```

Standardizing Number of previously posted projects by teachers

In [84]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.prepro
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.
# Reshape your data either using array.reshape(-1, 1)

project_scalar = StandardScaler()
project_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.resh
print(f"Mean : {project_scalar.mean_[0]}, Standard deviation : {np.sqrt(project_scalar.var_

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized = project_scalar.transform(projec
```

```
C:\Users\VANSHIKA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning:

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


Mean : 9.2928, Standard deviation : 24.92901659031098

C:\Users\VANSHIKA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:59
5: DataConversionWarning:

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

In [85]:

```
price_standardized
```

Out[85]:

```
array([[-0.63010955],
       [-0.63331574],
       [-0.41058463],
       ...,
       [ 0.27732819],
       [-0.11138724],
       [-0.13505065]])
```

1.12 Merging all the above features

In [86]:

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

```
(10000, 9)
(10000, 30)
(10000, 6108)
(10000, 1)
```

Assignment 2: Apply TSNE

2.1 t-SNE BoW encoding of project_title

In [87]:

```
# 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,state_one_hot,
            prefix_one_hot,grade_one_hot,title_bow,
            price_standardized,
            teacher_number_of_previously_posted_projects_standardized))
X.shape
```

Out[87]:

```
(10000, 765)
```

In [88]:

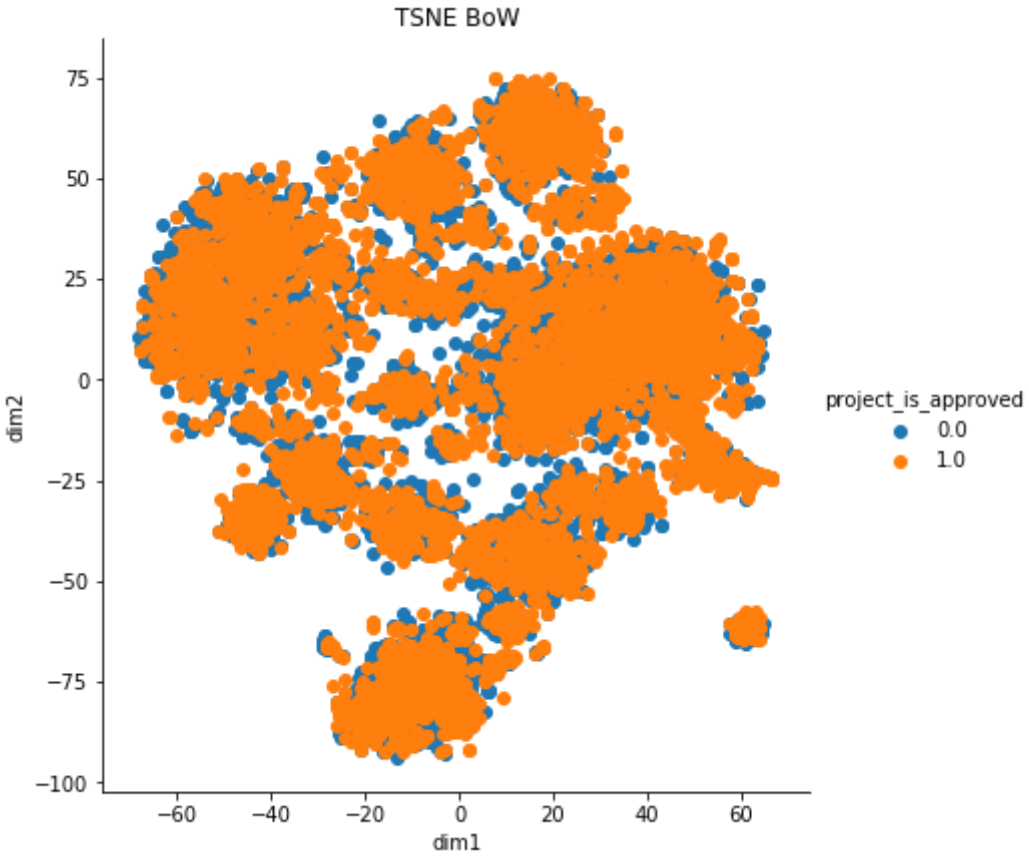
```
X_dense = X.todense()
type(X_dense)
```

Out[88]:

```
numpy.matrix
```

In [89]:

```
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 50, n_iter = 5000)

tsne_data = model.fit_transform(X_dense)
y = project_data['project_is_approved']

tsne_data = np.vstack((tsne_data.T, y)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("dim1", "dim2", "project_is_approved"))

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", height=6).map(plt.scatter, 'dim1', 'dim2'
plt.title("TSNE BoW")
plt.show()
```



Note:

1. Not much improvement on plot has been obeserved on changing preplexity and n_iter.
2. The approved and rejected projects are highly overlapped.

2.1 t-SNE for TFIDF

In [90]:

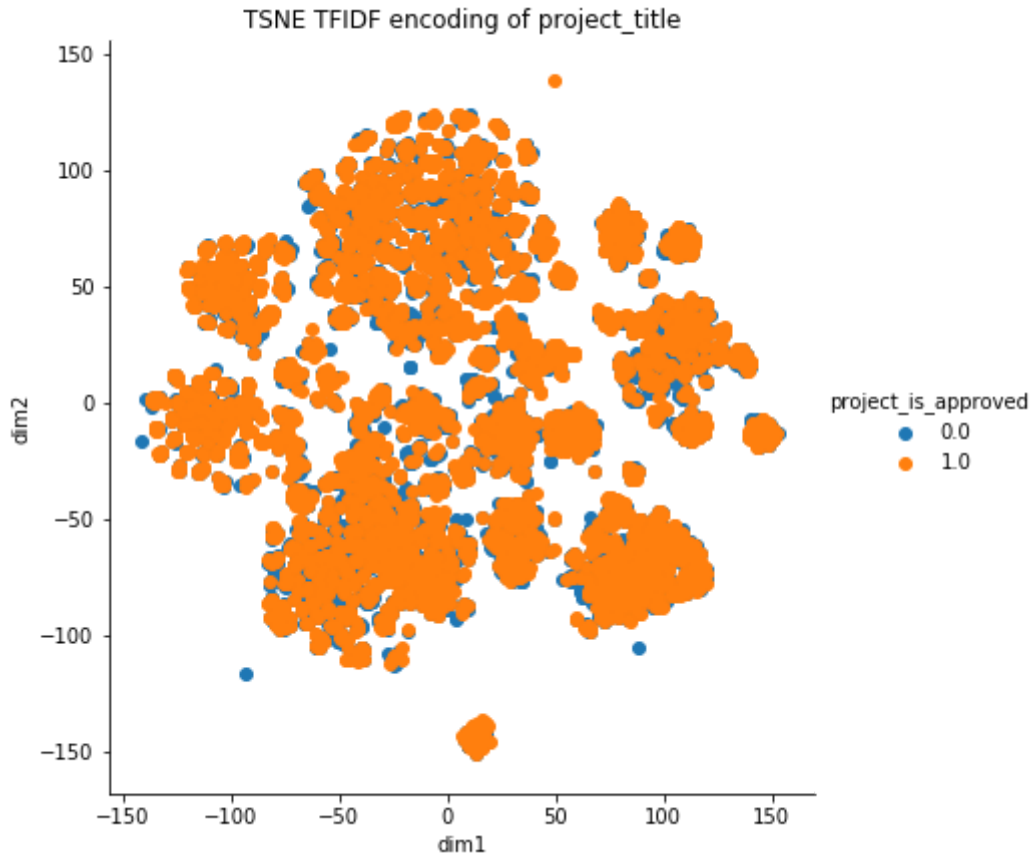
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot,state_one_hot,
            prefix_one_hot,grade_one_hot,text_tfidf,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

X_dense = X.todense()
type(X_dense)
```

Out[90]:

```
numpy.matrix
```

In [91]:

```
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 30, n_iter = 5000)

tsne_data = model.fit_transform(X_dense)

tsne_data = np.vstack((tsne_data.T, y)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("dim1", "dim2", "project_is_approved"))

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", height=6).map(plt.scatter, 'dim1', 'dim2'
plt.title("TSNE TFIDF encoding of project_title")
plt.show()
```



Note:

1. Not much improvement on plot has been obeserved on changing preplexity and n_iter.
2. The approved and rejected projects are highly overlapped.

2.3 TSNE with AVG W2V encoding of project_title

In [94]:

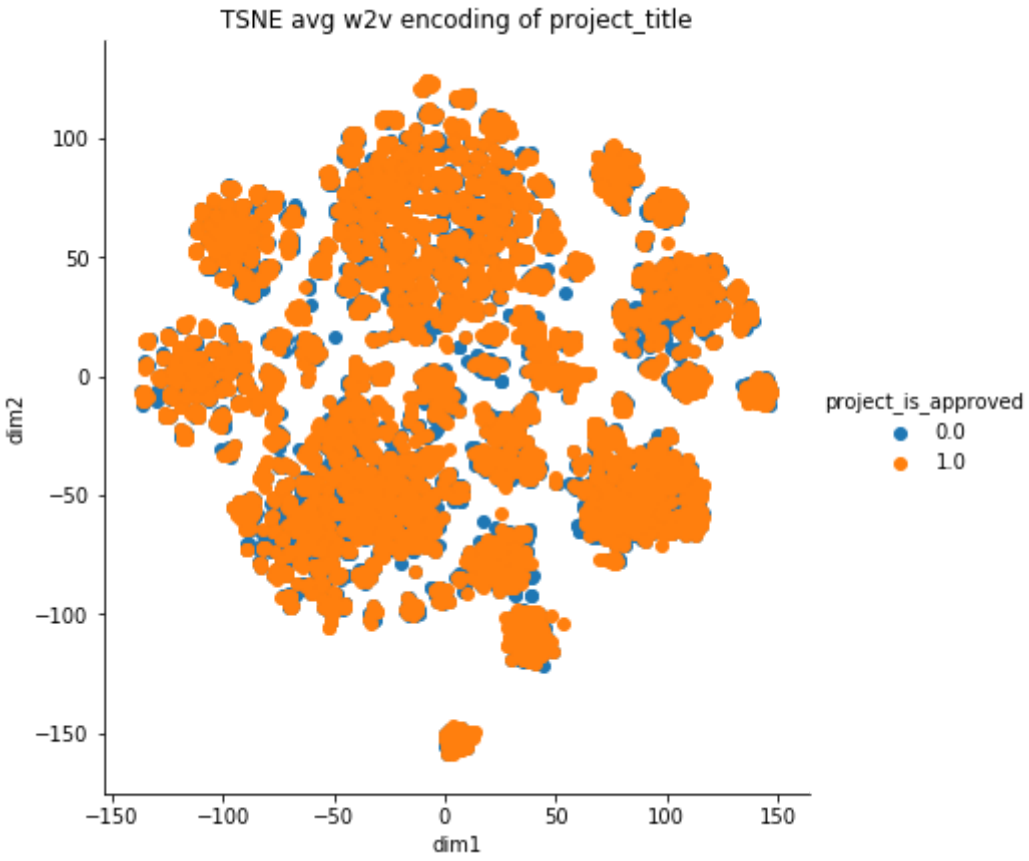
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot,state_one_hot,
            prefix_one_hot,grade_one_hot,avg_w2v_vectors_titles,
            price_standardized,
            teacher_number_of_previously_posted_projects_standardized))

X_dense = X.todense()
type(X_dense)
```

Out[94]:

```
numpy.matrix
```

In [95]:

```
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 30, n_iter = 5000)

tsne_data = model.fit_transform(X_dense)

tsne_data = np.vstack((tsne_data.T, y)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("dim1", "dim2", "project_is_approved"))

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", height=6).map(plt.scatter, 'dim1', 'dim2'
plt.title("TSNE avg w2v encoding of project_title")
plt.show()
```



Note:

1. Not much improvement on plot has been obeserved on changing preplexity and n_iter.
2. The approved and rejected projects are highly overlapped.

2.4 TSNE with TFIDF Weighted W2V encoding of project_title

In [96]:

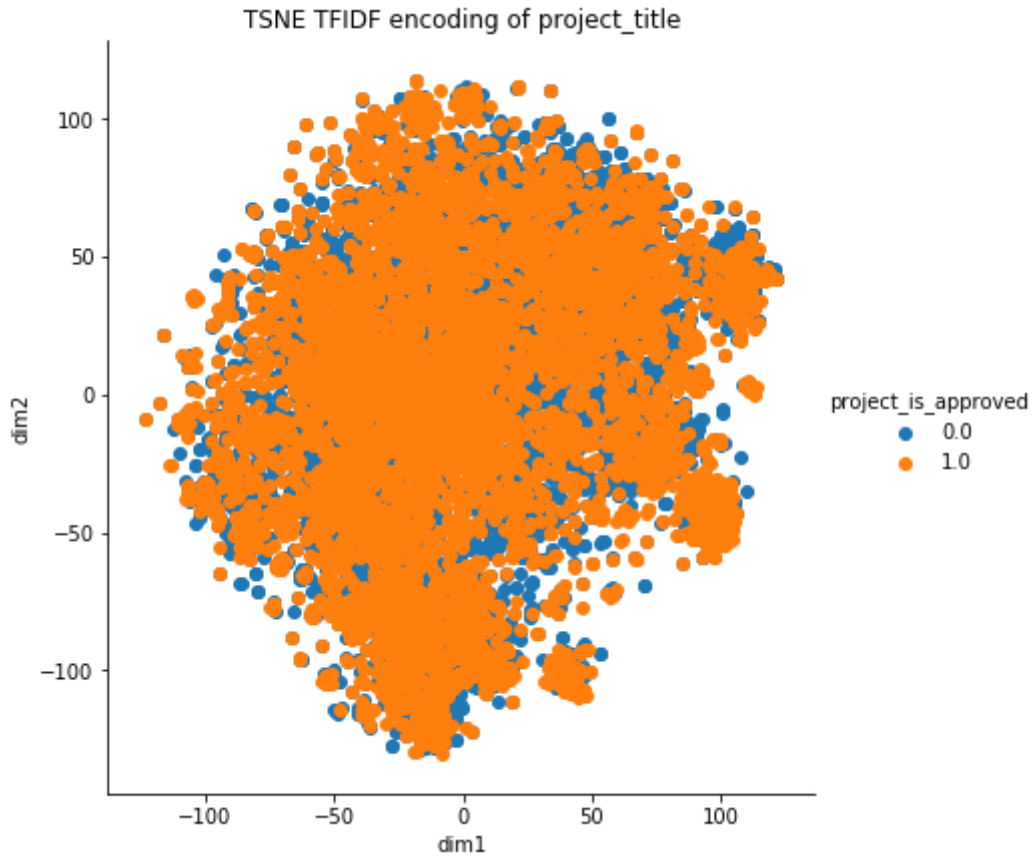
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot,state_one_hot,
            prefix_one_hot,grade_one_hot,tfidf_w2v_vectors_titles,
            price_standardized))

X_dense = X.todense()
type(X_dense)
```

Out[96]:

```
numpy.matrix
```

In [97]:

```
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0, perplexity = 30, n_iter = 5000)

tsne_data = model.fit_transform(X_dense)

tsne_data = np.vstack((tsne_data.T, y)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("dim1", "dim2", "project_is_approved"))

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="project_is_approved", height=6).map(plt.scatter, 'dim1', 'dim2'
plt.title("TSNE TFIDF encoding of project_title")
plt.show()
```



Note:

1. Not much improvement on plot has been obeserved on changing preplexity and n_iter.
2. The approved and rejected projects are highly overlapped.

Conclusion:

1. Vermont, District of Columbia, Texas, Montana and Louisiana are the states with least percentage of project approvals ranging 80%-83%.
2. New Hampshire, Ohio, Washington, North Dakota and Delaware are the stated with high percentage of project approvals ranging 87%-90% approx.
3. More number of projects in a particular category does not ensure higher approval rate.
4. The project based on "Literacy"and education" has higher approval rate.
5. Projects by Mrs.,MS.,and Mr. has relatively higher approval rate.
6. Price is not a good feature to see check whether the project will be approved or not.
7. Project summary with numerical values mentioned has higher approval rate.
8. The number of projects that contains digits in its resource summary has reatively higher approval rate than those doesn't have digits in its resource summary.
9. Projects with higher number words in its essay are more likely to get approved.
10. TSNE didn't work well to separate the two categories.