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	Descri
`project_id`	A unique identifier for the proposed project. **Example:** `p03`
`project_title`	Title of the project. **Exam** • `Art Will Make You Ha` • `First Grade`
`project_grade_category`	Grade level of students for which the project is targeted. One of the foll enumerated va • `Grades Pr` • `Grades` • `Grades` • `Grades`
`project_subject_categories`	One or more (comma-separated) subject categories for the project fro following enumerated list of va • `Applied Lear` • `Care & Hu` • `Health & Sp` • `History & Ci` • `Literacy & Lang` • `Math & Sci` • `Music & The` • `Special N` • `Wa` **Exam** • `Music & The` • `Literacy & Language, Math & Sci`
`school_state`	State where school is located (Two-letter U.S. postal (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_co **Example**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the pr **Exam** • `Lite` • `Literature & Writing, Social Scie`
`project_resource_summary`	An explanation of the resources needed for the project. **Exam** • `My students need hands on literacy materials to mar sensory ne`
`project_essay_1`	First application e
`project_essay_2`	Second application e
`project_essay_3`	Third application e
`project_essay_4`	Fourth application e

Feature	Descri
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-0` `12:43:56`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Exa** `bdf8baa8fedef6bfeec7ae4ff1c1`
`teacher_prefix`	Teacher's title. One of the following enumerated va • • • • • • `Teac`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same te **Exampl**

* 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 [126]:

```
%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 chart_studio import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.1 Reading Data

In [127]:

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

In [128]:

```
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' 'sc
hool_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 [129]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(6)
```

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

Out[129]:

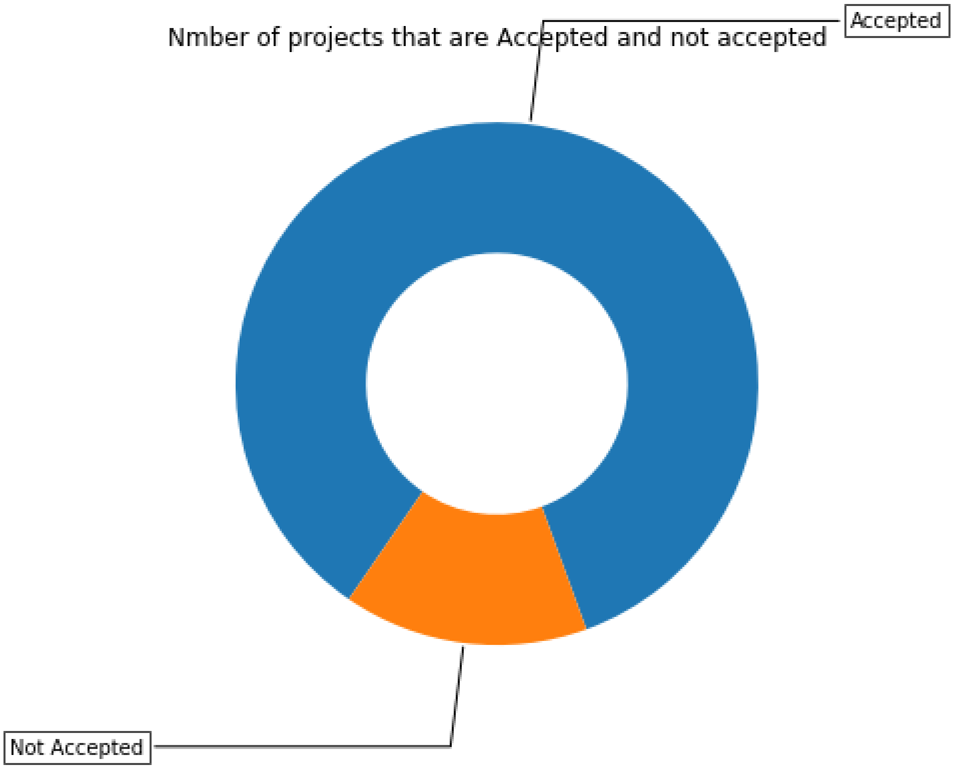
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95
5	p069063	Last to Finish: A Story About the Smartest Boy...	1	16.99

1.2 Data Analysis

In [130]:

```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.htm
l#sphx-glr-gallery-pie-and-polar-charts-pie-and-donut-labels-py


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

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

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

wedges, texts = ax.pie(data, wedgeprops=dict(width=0.5), startangle=-70)

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 , ( 84.85830404217927
%)
Number of projects thar are not approved for funding  16542 , ( 15.1416959578
20739 %)
```

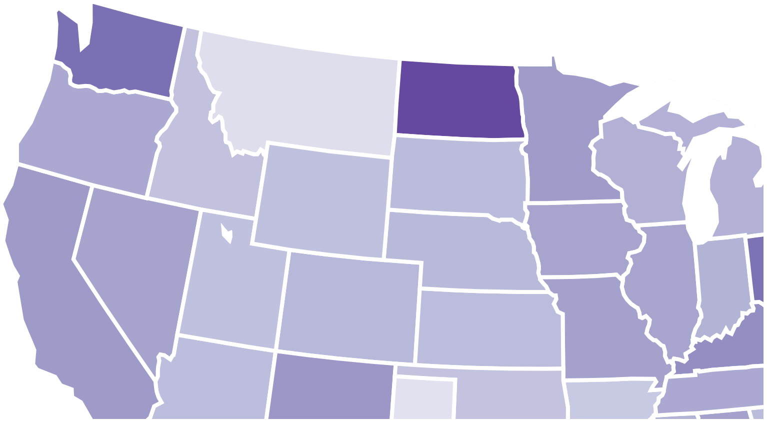


1.2.1 Univariate Analysis: School State

In [131]:

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

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].apply(np.mean)).reset_index()
# apply function in pandas dataframe, takes in the argument as a lambda function applying to each row
#  by passing the row contents and return the result without modifying the original dataframe
# if you have data which contain only 0 and 1, then the mean = percentage (think about it)
temp.columns = ['state_code', 'num_proposals']

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

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)']]

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

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

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

Project Proposals % of Acceptance Rate by US States



In [132]:

```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letters
tabbrev.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
```

Summary: The State "DE" (Delaware) appears to be the one with highest approval rate reaching almost 90% followed by North Dakota and Washington with 88% and 87% respectively. Poor Vermont (VT) it has the lowest acceptance rate with 80% and DC being 80.2 , followed by Texas mearly beating VT by one percent higher.

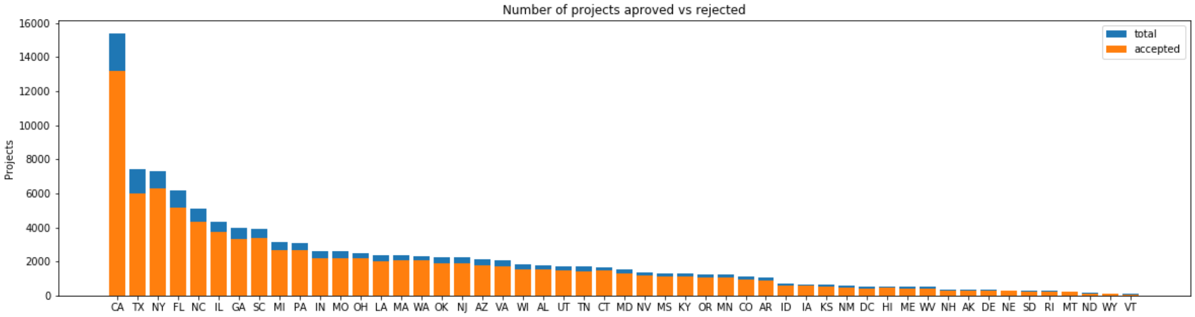
In [133]:

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

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

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

In [134]:

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

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

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

    if top:
        temp = temp[0:top]

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

```
In [121]: univariate_barplots(project_data, 'school_state', 'project_is_approved', False
          )
```



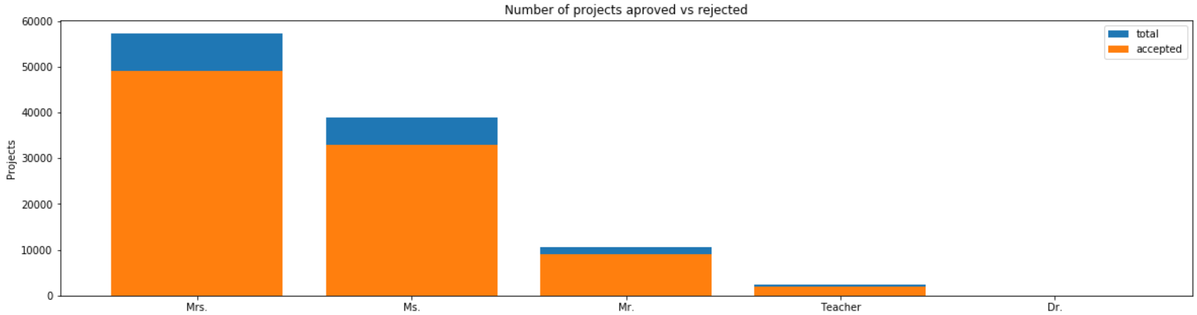
```
   school_state  project_is_approved  total       Avg
4            CA                13205  15388  0.858136
43           TX                 6014   7396  0.813142
34           NY                 6291   7318  0.859661
9            FL                 5144   6185  0.831690
27           NC           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

There is high variablity in the number of projects submitted for approval. Just look at CA, it has over 15000 submissions and VT just 80.

1.2.2 Univariate Analysis: teacher_prefix

```
In [135]: univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , to
          p=False)
```



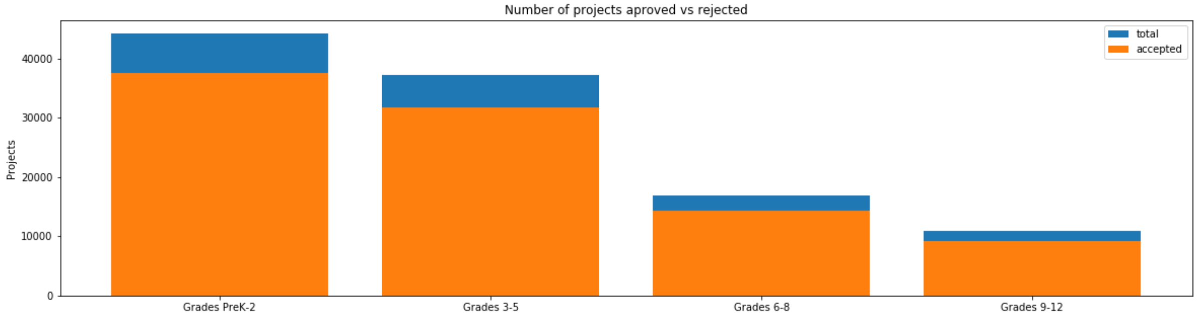
```
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.     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

Summary: Most of the proposals on approved/rejected side are sent by Mrs. and least by Dr. Like they say Women dominate in every field. The number of proposals made by Mrs and Ms are nearly 7 times the number of male submissions.

1.2.3 Univariate Analysis: project_grade_category

In [136]:

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



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

Summary: Most number of proposals are sent for Grades PreK-2, but the highest acceptance rate is for Grades 3-5, with lowest approval rate of around 83.7% for Grades 9-12 and an overall avg acceptance rate of 84%

1.2.4 Univariate Analysis: project_subject_categories

In [137]:

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

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

In [138]:

```
cat_list[0:10]
```

Out[138]:

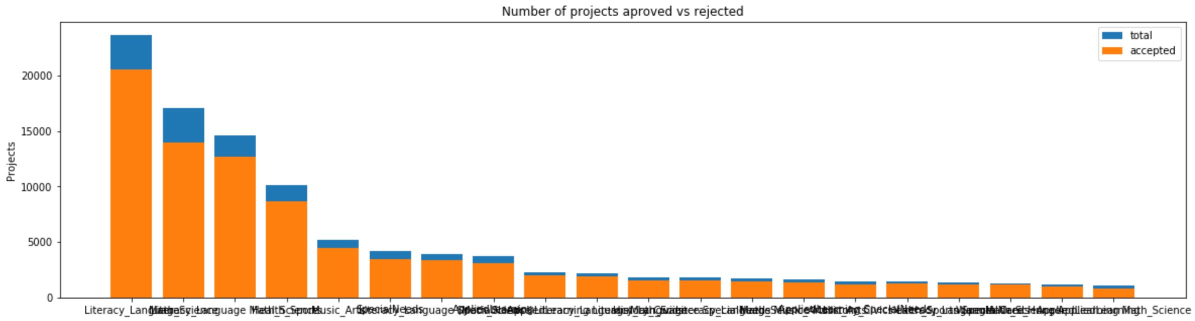
```
['Literacy_Language',
 'History_Civics Health_Sports',
 'Health_Sports',
 'Literacy_Language Math_Science',
 'Math_Science',
 'Literacy_Language SpecialNeeds',
 'Literacy_Language SpecialNeeds',
 'Math_Science',
 'Health_Sports',
 'Literacy_Language']
```

In [139]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[139]:

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

In [140]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', t
op=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
```

Summary: Based on the above plot, the highest number of submissions category goes for Literacy_Language and lowest submissions category being Appliedlearning Math_Science. From the plot we can conclude that Literacy and language are given higher importance compared to applied skills and math. Also when it comes to hunger and warmth, no society would deny a submission, which is eveident from the highest 92.5% acceptance rate.

In [141]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

In [142]:

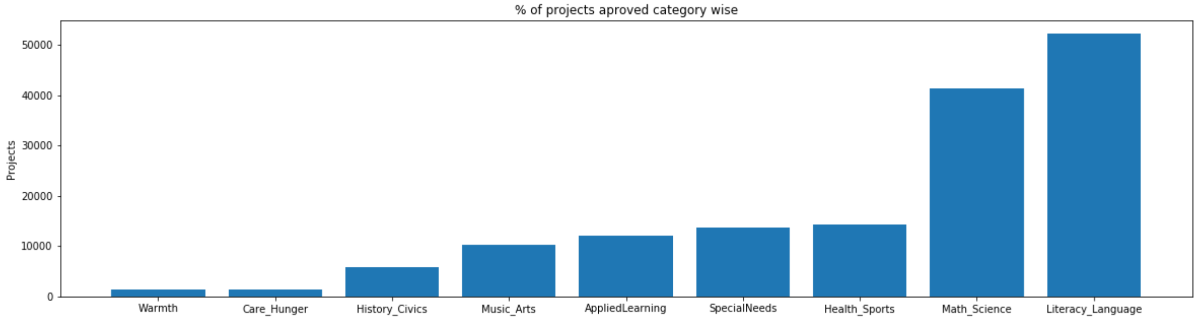
```
my_counter
```

Out[142]:

```
Counter({'Literacy_Language': 52239,
         'History_Civics': 5914,
         'Health_Sports': 14223,
         'Math_Science': 41421,
         'SpecialNeeds': 13642,
         'AppliedLearning': 12135,
         'Music_Arts': 10293,
         'Warmth': 1388,
         'Care_Hunger': 1388})
```

In [143]:

```
# 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 [144]:

```
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
```

Summary: The bar graph pretty much speaks to itself, the highest approval rate category being Literacy_Language and lowest being Warmth.

1.2.5 Univariate Analysis: project_subject_subcategories

In [145]:

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

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

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

In [146]:

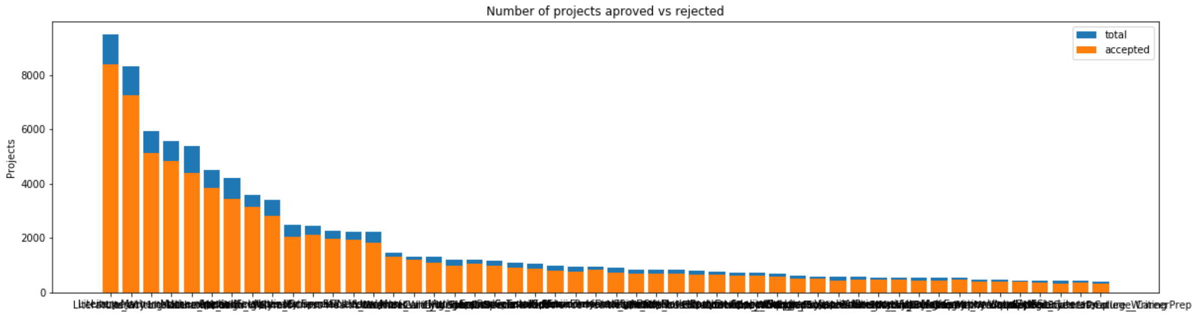
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[146]:

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

In [147]:

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

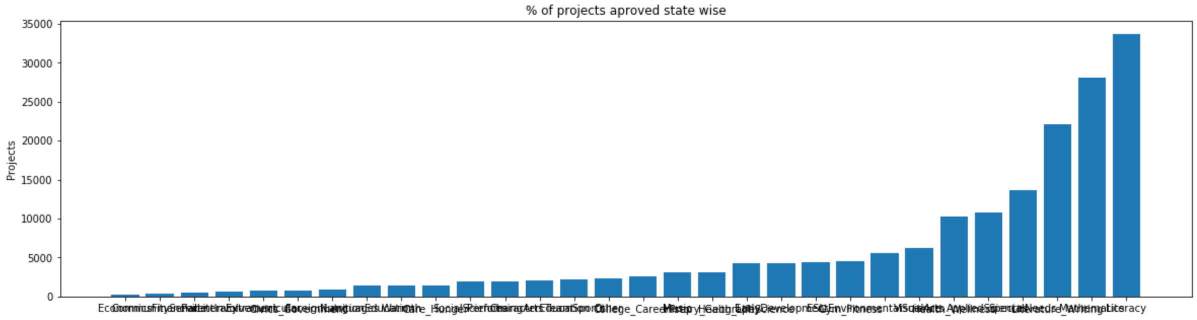


```
                    clean_subcategories  project_is_approved  total       Avg
317                            Literacy                 8371   9486  0.882458
319                Literacy Mathematics                 7260   8325  0.872072
331  Literature_Writing Mathematics                     5140   5923  0.867803
318     Literacy Literature_Writing                     4823   5571  0.865733
342                         Mathematics                 4385   5379  0.815207
==================================================
                        clean_subcategories  project_is_approved  total       Avg
196        EnvironmentalScience Literacy                     389    444  0.876126
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
```

summary: 1) Plot literally showcases that the subcategory with highest submissions and acceptance is non other than Literacy. 2) Since most of the students are in school under kindergarten any project for college and careerprep doesn't make much sense, well that is what the graph shows with just 81% acceptance rate.

In [24]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [25]:
```
# 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()
```



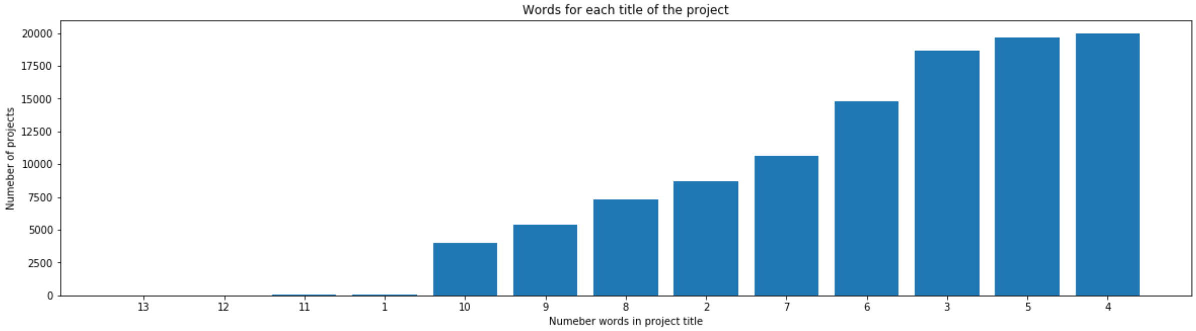
In [26]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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 [27]:

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


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

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

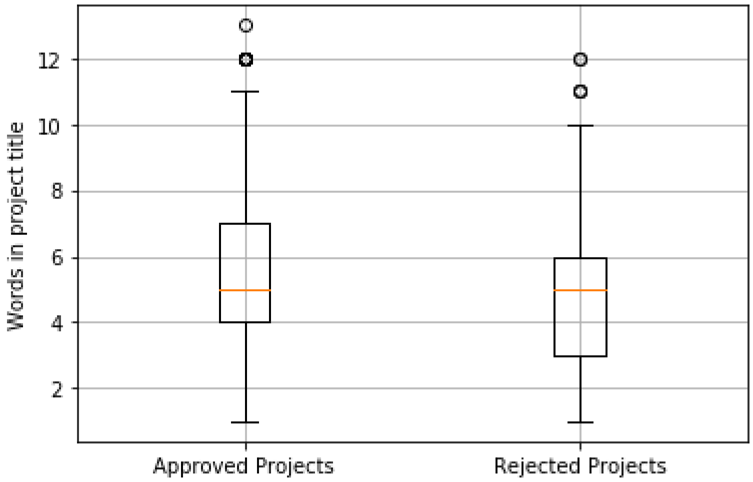


Summary: The max length of the title appears to be 13, with smallest being around 4 words. Very few projects have massive title lengths, most of them have very few words around 4 to 6.

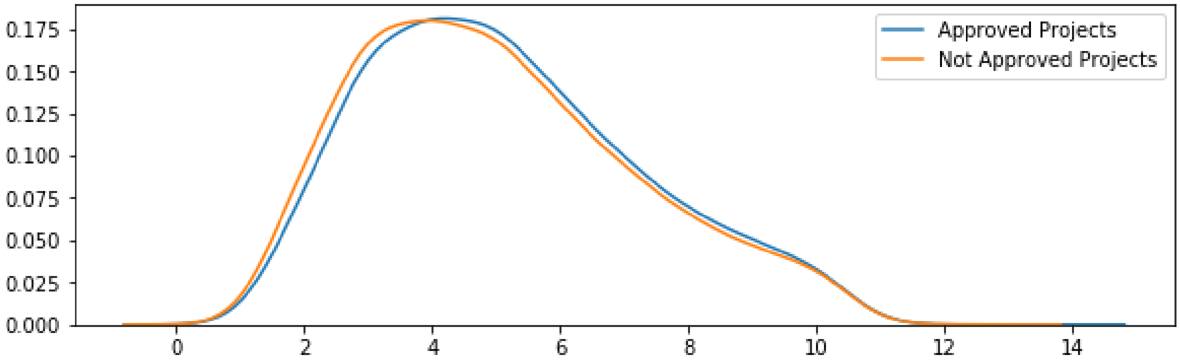
In [28]:

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

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

In [29]:
```
# 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 [30]:
```
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()
```



Summary: When the words in title are too less, the rejection rate is higher.

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

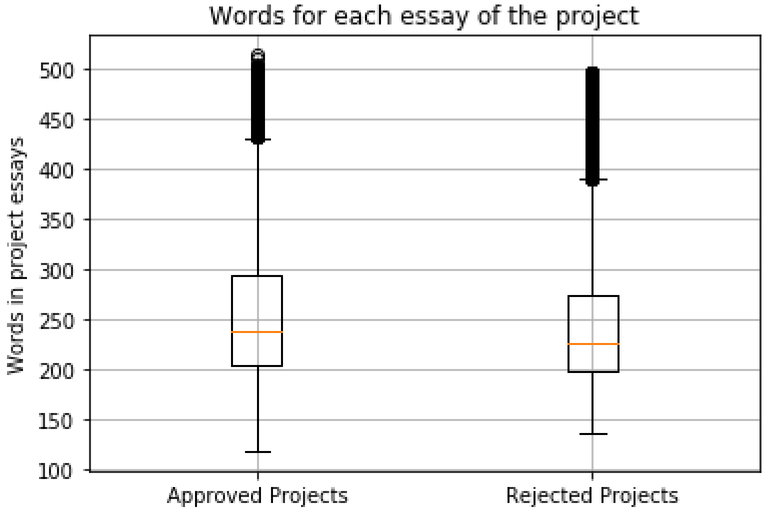
In [31]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [32]:

```
approved_word_count = project_data[project_data['project_is_approved']==1]['es
say'].str.split().apply(len)
approved_word_count = approved_word_count.values

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

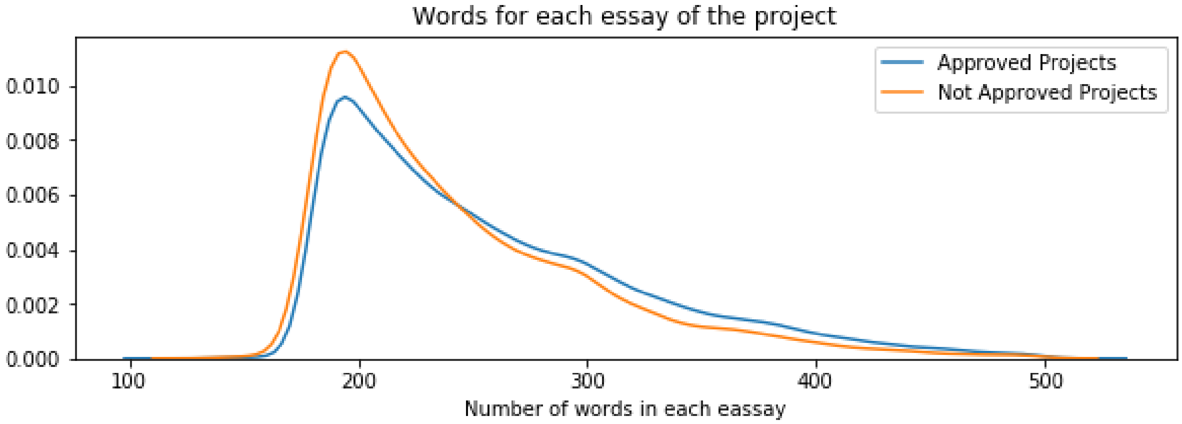
In [33]:

```
# 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 [34]:

```
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()
```



Summary: If the number of words in essay are fewer the chances of its rejection are higher.

1.2.8 Univariate Analysis: Cost per project

In [35]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[35]:

	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 [36]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[36]:

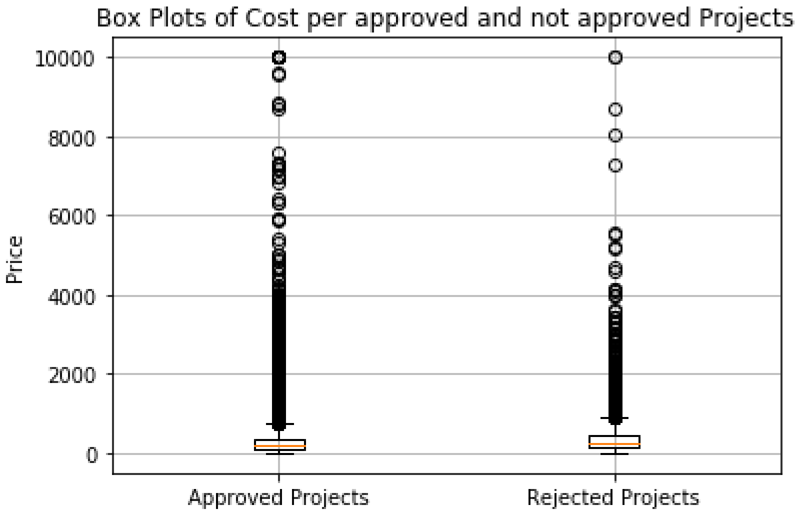
	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [37]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [38]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].values

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

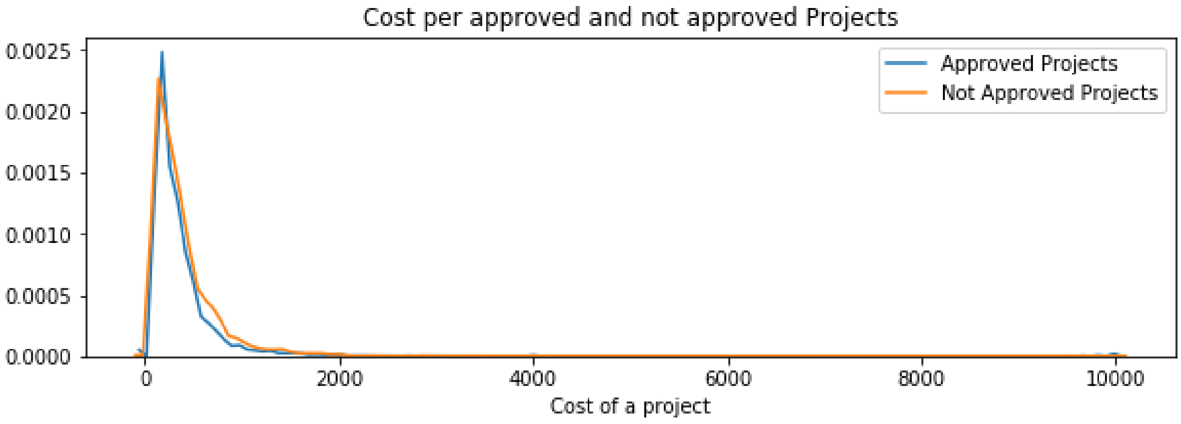
In [39]:

```
# 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 [40]:

```
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()
```



Summary: People are always sceptical in matters that costs too much. The graph kind of tells the same. When the cost of project is high, it's approval rate is low.

In [41]:

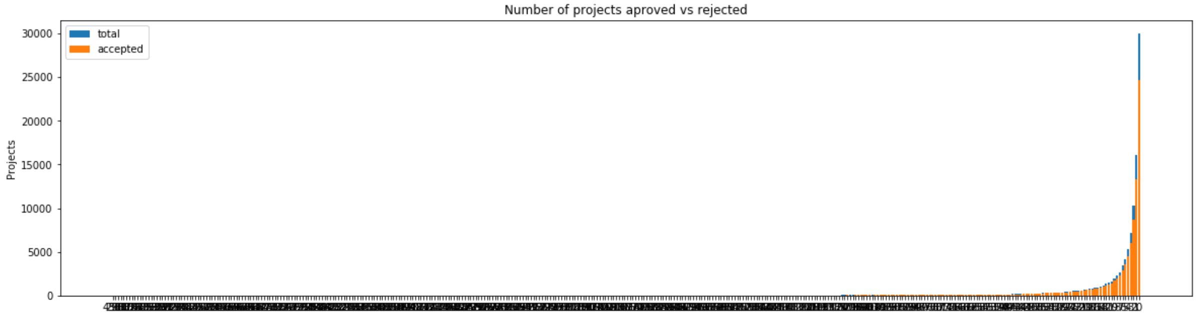
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install 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(rejected_price,i), 3)])
print(x)
```

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

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 [42]:
```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_project
s', 'project_is_approved' , top=False)
```



```
      teacher_number_of_previously_posted_projects  project_is_approved  total
\
373                                            451                    1      1
292                                            296                    1      1
302                                            306                    1      1
304                                            308                    1      1
305                                            310                    1      1

     Avg
373  1.0
292  1.0
302  1.0
304  1.0
305  1.0
==================================================
   teacher_number_of_previously_posted_projects  project_is_approved  total
\
4                                             4                 4452   5266
3                                             3                 5997   7110
2                                             2                 8705  10350
1                                             1                13329  16058
0                                             0                24652  30014

        Avg
4  0.845423
3  0.843460
2  0.841063
1  0.830054
0  0.821350
```

Summary: We observed that for the teachers who have previously posted higher number of projects, the proposal acceptance rate is higher almost 100% compared to teachers who previously submitted fewer none projects whose proposal acceptance rate ranges from 84% to lowest of 82%

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.

Univariate analysis on project_resource_summary - based on summary word count

In [43]:
```
approved_summary_word_count = project_data[project_data["project_is_approved"]
==1]["project_resource_summary"].str.split().apply(len)
approved_summary_word_count = approved_summary_word_count.values
```

In [44]:
```
print(approved_summary_word_count)
```

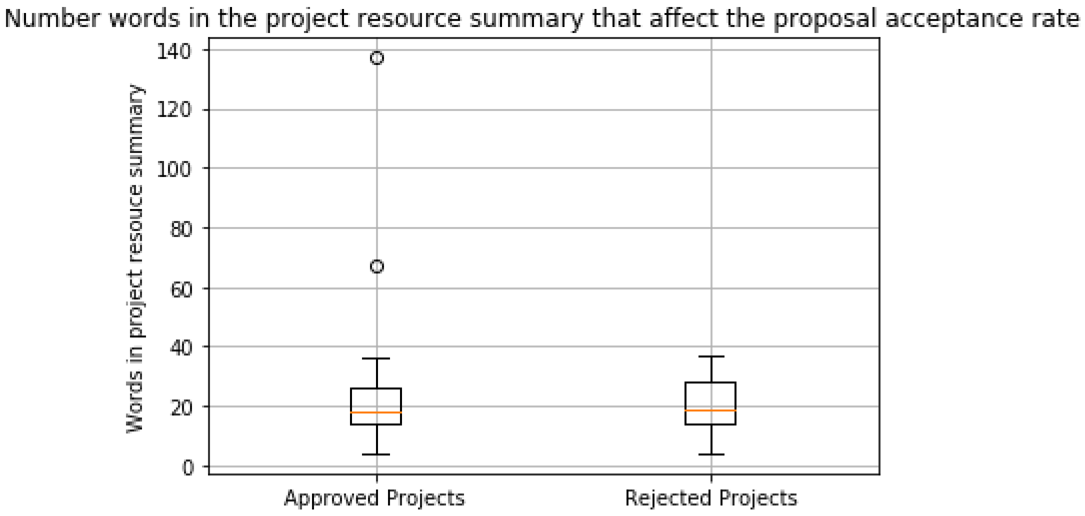
```
[11 20 26 ... 36 15 27]
```

In [45]:
```
rejected_summary_word_count = project_data[project_data["project_is_approved"]
==0]["project_resource_summary"].str.split().apply(len)
rejected_summary_word_count = rejected_summary_word_count.values
```

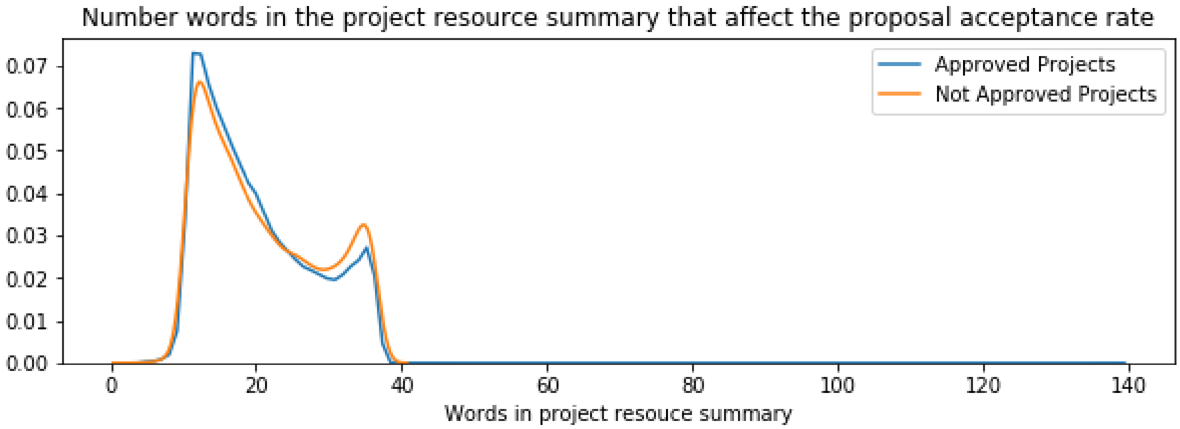
In [46]:
```
print(rejected_summary_word_count)
```

```
[13 19 32 ... 19 11 18]
```

In [47]:
```
# representing the values using box plot and pdfs for better visualization
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_summary_word_count, rejected_summary_word_count])
plt.title('Number words in the project resource summary that affect the propos
al acceptance rate')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project resouce summary')
plt.grid()
plt.show()
```



In [48]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_summary_word_count, hist=False, label="Approved Project
s")
sns.distplot(rejected_summary_word_count, hist=False, label="Not Approved Proj
ects")
plt.title('Number words in the project resource summary that affect the propos
al acceptance rate')
plt.xlabel('Words in project resouce summary')
plt.legend()
plt.show()
```



Summary: It is evident that there are certain extreme points in accepted box plot. But when we look at the 50th percentile approx 20 words in summary, the approval rate is higher than rejection rate, and as the number of words in resource summary increase beyond 30+, the rejection rate is higher.

Univariate analysis on project_resource_summary - based on presence of numerics in summary

In [49]:
```
# function to return the presence of numbers in the resource summary
# function uses regular expressions re.search to identify the presence of numb
ers
def return_true_if_numbers_in_summary(string_data):
    return bool(re.search(r"\d",string_data))
```

In [50]:

```
project_data["resource_summary_has_digits"] = project_data["project_resource_summary"].apply(return_true_if_numbers_in_summary)
project_data.head(5)
```

Out[50]:

	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

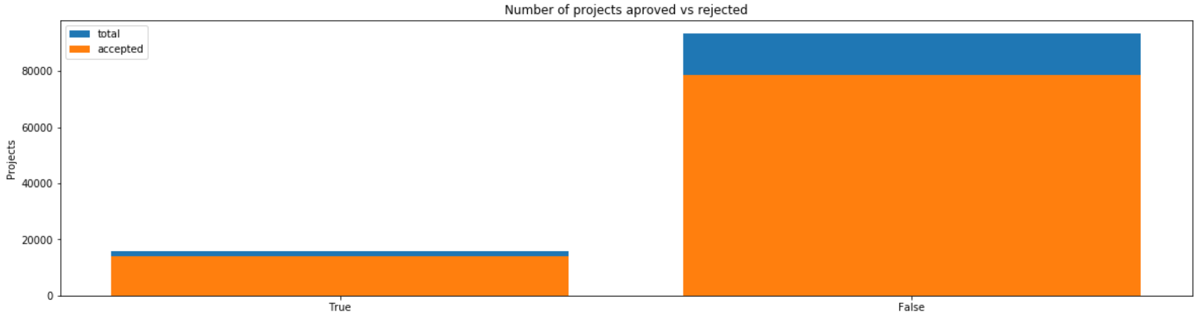
In [51]:

```
project_data.iloc[16]
```

Out[51]:

```
Unnamed: 0
127215
id
p174627
teacher_id                                                       4ad7e280fddf
f889e1355cc9f29c3b89
teacher_prefix
Mrs.
school_state
FL
project_submitted_datetime
2017-01-18 10:59:05
project_grade_category
Grades PreK-2
project_title                                                  Making Great L
EAP's With Leapfrog!
project_essay_1                                 My Preschool children, ages 3
-5 years old with...
project_essay_2                                 Having a set of Leapfrog iPad
s and educational...
project_essay_3
NaN
project_essay_4
NaN
project_resource_summary                        My students need 2 LeapPad th
at will engage th...
teacher_number_of_previously_posted_projects
1
project_is_approved
1
clean_categories                                                   Literacy_L
anguage SpecialNeeds
clean_subcategories                                                         L
iteracy SpecialNeeds
essay                                           My Preschool children, ages 3
-5 years old with...
price
298.43
quantity
7
resource_summary_has_digits
True
Name: 16, dtype: object
```

In [52]:
```
univariate_barplots(project_data, 'resource_summary_has_digits', 'project_is_a
pproved', top=False)
```



```
   resource_summary_has_digits  project_is_approved  total       Avg
1                         True                14090  15756  0.894263
0                        False                78616  93492  0.840885
==================================================
   resource_summary_has_digits  project_is_approved  total       Avg
1                         True                14090  15756  0.894263
0                        False                78616  93492  0.840885
```

Summary: It is evident that when the summary contains numerics, the proposal acceptance rate is higher almost an avg of 89% compared to the cases where the acceptance rate is mere 84% when there is no numeric data in summary.

1.3 Text preprocessing

1.3.1 Essay Text

In [53]:
```
project_data.head(2)
```

Out[53]:

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

2 rows × 21 columns

In [54]:

```
# 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 level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and ex
periences to us that open our eyes to new cultures, beliefs, and respect.\"Th
e limits of your language are the limits of your world.\"-Ludwig Wittgenstein
Our English learner's have a strong support system at home that begs for more
resources.  Many times our parents are learning to read and speak English alo
ng side of their children.  Sometimes this creates barriers for parents to be
able to help their child learn phonetics, letter recognition, and other readi
ng skills.\r\n\r\nBy providing these dvd's and players, students are able to
continue their mastery of the English language even if no one at home is able
to 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 wil
l be specially chosen by the English Learner Teacher and will be sent home re
gularly to watch.  The videos are to help the child develop early reading ski
lls.\r\n\r\nParents that do not have access to a dvd player will have the opp
ortunity to check out a dvd player to use for the year.  The plan is to use t
hese videos and educational dvd's for the years to come for other EL student
s.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year al
l love learning, at least most of the time. At our school, 97.3% of the stude
nts receive free or reduced price lunch. Of the 560 students, 97.3% are minor
ity students. \r\nThe school has a vibrant community that loves to get togeth
er and celebrate. Around Halloween there is a whole school parade to show off
the beautiful costumes that students wear. On Cinco de Mayo we put on a big f
estival with crafts made by the students, dances, and games. At the end of th
e year the school hosts a carnival to celebrate the hard work put in during t
he school year, with a dunk tank being the most popular activity.My students
will use these five brightly colored Hokki stools in place of regular, statio
nary, 4-legged chairs. As I will only have a total of ten in the classroom an
d not enough for each student to have an individual one, they will be used in
a variety of ways. During independent reading time they will be used as speci
al chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of th
e day they will be used by the students who need the highest amount of moveme
nt in their life in order to stay focused on school.\r\n\r\nWhenever asked wh
at the classroom is missing, my students always say more Hokki Stools. They c
an't get their fill of the 5 stools we already have. When the students are si
tting in group with me on the Hokki Stools, they are always moving, but at th
e 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 wh
o head over to the kidney table to get one of the stools who are disappointed
as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for
7 hours a day. The Hokki stools will be a compromise that allow my students t
o do desk work and move at the same time. These stools will help students to
meet their 60 minutes a day of movement by allowing them to activate their co
re 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 s
till.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with
plain walls, rows of desks, and a teacher in front of the room? A typical day
in our room is nothing like that. I work hard to create a warm inviting theme
```

```
d room for my students look forward to coming to each day.\r\n\r\nMy class is
made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r
\nThey attend a Title I school, which means there is a high enough percentage
of free and reduced-price lunch to qualify. Our school is an \"open classroom
\" concept, which is very unique as there are no walls separating the classro
oms. These 9 and 10 year-old students are very eager learners; they are like
sponges, absorbing all the information and experiences and keep on wanting mo
re.With these resources such as the comfy red throw pillows and the whimsical
nautical hanging decor and the blue fish nets, I will be able to help create
the mood in our classroom setting to be one of a themed nautical environment.
Creating a classroom environment is very important in the success in each and
every child's education. The nautical photo props will be used with each chil
d as they step foot into our classroom for the first time on Meet the Teacher
evening. I'll take pictures of each child with them, have them developed, and
then hung in our classroom ready for their first day of 4th grade.  This kind
gesture will set the tone before even the first day of school! The nautical t
hank you cards will be used throughout the year by the students as they creat
e thank you cards to their team groups.\r\n\r\nYour generous donations will h
elp me to help make our classroom a fun, inviting, learning environment from
day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to g
et our classroom ready. Please consider 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 lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations. \r\n\r\nThe materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or red
uced price lunch.  Despite their disabilities and limitations, my students lo
ve coming to school and come eager to learn and explore.Have you ever felt li
ke you had ants in your pants and you needed to groove and move as you were i
n a meeting? This is how my kids feel all the time. The want to be able to mo
ve as they learn or so they say.Wobble chairs are the answer and I love then
because they develop their core, which enhances gross motor and in Turn fine
motor skills. \r\nThey also want to learn through games, my kids don't want t
o 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
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher d
emonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school ha
s 803 students which is makeup is 97.6% African-American, making up the large
st segment of the student body. A typical school in Dallas is made up of 23.
2% African-American students. Most of the students are on free or reduced lun
ch. We aren't receiving doctors, lawyers, or engineers children from rich bac
kgrounds or neighborhoods. As an educator I am inspiring minds of young child
ren and we focus not only on academics but one smart, effective, efficient, a
nd disciplined students with good character.In our classroom we can utilize t
he Bluetooth for swift transitions during class. I use a speaker which does
n't amplify the sound enough to receive the message. Due to the volume of my
speaker my students can't hear videos or books clearly and it isn't making th
e lessons as meaningful. But with the bluetooth speaker my students will be a
ble to hear and I can stop, pause and replay it at any time.\r\nThe cart will
allow me to have more room for storage of things that are needed for the day
and has an extra part to it I can use.  The table top chart has all of the le
tter, words and pictures for students to learn about different letters and it
```

```
is more accessible.nannan
==================================================
```

In [55]:

```python
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [56]:

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

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations. \r\n\r\nThe materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or red
uced price lunch.  Despite their disabilities and limitations, my students lo
ve coming to school and come eager to learn and explore.Have you ever felt li
ke you had ants in your pants and you needed to groove and move as you were i
n a meeting? This is how my kids feel all the time. The want to be able to mo
ve as they learn or so they say.Wobble chairs are the answer and I love then
because they develop their core, which enhances gross motor and in Turn fine
motor skills. \r\nThey 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 [57]:

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

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations.     The 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 co
ming to school and come eager to learn and explore.Have you ever felt like yo
u had ants in your pants and you needed to groove and move as you were in a m
eeting? 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 becaus
e they develop their core, which enhances gross motor and in Turn fine motor
skills.   They also want to learn through games, my kids do not want to sit a
nd do worksheets. They want to learn to count by jumping and playing. Physica
l engagement is the key to our success. The number toss and color and shape m
ats can make that happen. My students will forget they are doing work and jus
t have the fun a 6 year old deserves.nannan
```

In [58]:

```
#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 lan
guage delays cognitive delays gross fine motor delays to autism They are eage
r beavers and always strive to work their hardest working past their limitati
ons The 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 lun
ch Despite their disabilities and limitations my students love coming to scho
ol and come eager to learn and explore Have you ever felt like you had ants i
n 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
their core which enhances gross motor and in Turn fine motor skills They also
want to learn through games my kids do not want to sit and do worksheets They
want to learn to count by jumping and playing Physical engagement is the key
to our success 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 ol
d deserves nannan
```

In [59]:

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

In [60]:

```
# 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 [02:36<00:00, 700.19it/s]
```

In [61]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[61]:
```
'my kindergarten students varied disabilities ranging speech language delays
cognitive delays gross fine motor delays autism they eager beavers always str
ive work hardest working past limitations the materials ones i seek students
i teach title i school students receive free reduced price lunch despite disa
bilities limitations students love coming school come eager learn explore hav
e ever felt like ants pants needed groove move meeting this kids feel time th
e want able move learn say wobble chairs answer i love develop core enhances
gross motor turn fine motor skills they also want learn games kids not want s
it worksheets they want learn count jumping playing physical engagement key s
uccess the number toss color shape mats make happen my students forget work f
un 6 year old deserves nannan'
```

1.3.2 Project title Text

In [62]:
```
# similarly you can preprocess the titles also
def preprocess_text_func(text_data):
    sent = decontracted(text_data)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    return sent.lower()
```

In [63]:
```
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    preprocessed_titles.append(preprocess_text_func(sentance))
```

```
100%|████████████████████████████████████████████████████████████|
109248/109248 [00:07<00:00, 15073.85it/s]
```

In [64]:
```
print(preprocessed_titles[5:12])
```

```
['flexible seating mrs jarvis terrific third graders', 'chromebooks special e
ducation reading program', 'it 21st century', 'targeting more success class',
'just for love reading pure pleasure', 'reading changes lives', 'elevating ac
ademics parent rapports through technology']
```

In [65]:
```
print(project_data["project_title"].values[5:12])
```

```
["Flexible Seating for Mrs. Jarvis' Terrific Third Graders!!"
 'Chromebooks for Special Education Reading Program'
 "It's the 21st Century" 'Targeting More Success in Class'
 'Just For the Love of Reading--\\r\\nPure Pleasure'
 'Reading Changes Lives'
 'Elevating Academics and Parent Rapports Through Technology']
```

1. 4 Preparing data for models

In [66]:
```
project_data.columns
```

Out[66]:
```
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_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantit
y',
       'resource_summary_has_digits'],
      dtype='object')
```

we are going to consider

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

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

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

1.4.1 Vectorizing Categorical data

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

In [67]:

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


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [68]:

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


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducati
on', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geo
graphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Env
ironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'Spec
ialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [69]:

```python
def perform_one_hot_encoding(listdata, category,fillnan_value=""):
    vectorizer =  CountVectorizer(vocabulary=listdata, lowercase=False, binary=True)
    vectorizer.fit(project_data[category].fillna(fillnan_value).values)
    print(vectorizer.get_feature_names())
    print("="*50)
    return vectorizer.transform(project_data[category].fillna(fillnan_value).values)
```

In [70]:
```
# One hot encoding for school state
countries_list = sorted(project_data["school_state"].value_counts().keys())
school_state_one_hot = perform_one_hot_encoding(countries_list, "school_state"
)
print("Shape of matrix after one hot encodig ",school_state_one_hot.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'I
A', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO',
'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR',
'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
==================================================
Shape of matrix after one hot encodig  (109248, 51)
```

In [71]:
```
# Please do the similar feature encoding with state, teacher_prefix and projec
t_grade_category also
# One hot encoding for teacher_prefix
teacher_prefix_list = sorted(project_data["teacher_prefix"].fillna("Mrs.").val
ue_counts().keys())
print (teacher_prefix_list)
teacher_prefix_one_hot = perform_one_hot_encoding(teacher_prefix_list, "teache
r_prefix", "Mrs.")
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
```

```
['Dr.', 'Mr.', 'Mrs.', 'Ms.', 'Teacher']
['Dr.', 'Mr.', 'Mrs.', 'Ms.', 'Teacher']
==================================================
Shape of matrix after one hot encodig  (109248, 5)
```

In [72]:
```
# One hot encoding for project_grade_category
grade_list = sorted(project_data["project_grade_category"].value_counts().keys
())
grade_one_hot = perform_one_hot_encoding(grade_list, "project_grade_category")
print("Shape of matrix after one hot encodig ",grade_one_hot.shape)
```

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

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [73]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

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

1.4.2.2 Bag of Words on `project_title`

In [74]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_titles = CountVectorizer(min_df=10)
text_bow_titles = vectorizer_titles.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",text_bow_titles.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3329)
```

In [75]:

```
# Similarly you can vectorize for title also
```

1.4.2.3 TFIDF vectorizer

In [76]:

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

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

1.4.2.4 TFIDF Vectorizer on `project_title`

In [77]:

```
# Similarly you can vectorize for title also
vectorizer_titles = TfidfVectorizer(min_df=10)
text_tfidf_titles = vectorizer_titles.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",text_tfidf_titles.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3329)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [78]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/408403
9
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 coup
us", \
      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-pickle-to-save-and-load-variables-in-python/

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

```
'''
```

Out[78]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/40
84039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    
f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in t
qdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n       
embedding = np.array([float(val) for val in splitLine[1:]])\n        model[wo
rd] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return 
model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ==================
==========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/
s]\nDone. 1917495  words loaded!\n\n# ============================\n\nwords =
[]\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in p
reproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in t
he coupus", len(words))\nwords = set(words)\nprint("the unique words in the c
oupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\np
rint("The number of words that are present in both glove vectors and our coup
us",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,
3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in wor
ds:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("wo
rd 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle 
files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-v
ariables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as
f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [79]:

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

In [80]:

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

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

```
100%|████████████████████████████████████████████████████████████████████████
| 109248/109248 [01:20<00:00, 1355.95it/s]

109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [81]:

```
# Similarly you can vectorize for title also
avg_w2v_vectors_titles = []; # the avg-w2v for each project_title is stored in this list
for sentence in tqdm(preprocessed_titles): # for each project_title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/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%|██████████████████████████████████████████████████|
109248/109248 [00:04<00:00, 26027.74it/s]

109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [82]:

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

In [83]:

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

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

```
100%|████████████████████████████████████████████████████████████████████
| 109248/109248 [3:12:14<00:00,  9.47it/s]

109248
300
```

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

In [84]:

```
# Similarly you can vectorize for title also
tfidf_w2v_vectors_titles = []; # the avg-w2v for each project_title is stored
 in this list
for sentence in tqdm(preprocessed_titles): # for each project_title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/revie
w
    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 v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles.append(vector)

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

```
100%|████████████████████████████████████████████████████████████████|
109248/109248 [00:09<00:00, 11761.98it/s]

109248
300
```

1.4.3 Vectorizing Numerical features

In [85]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

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

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

In [86]:
```
price_standardized
```

Out[86]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

In [87]:
```
# Vectorizing teacher_number_of_previously_posted_projects
teacher_number_of_previously_posted_projects_scalar = StandardScaler()
teacher_number_of_previously_posted_projects_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_[0]}, Standard deviation : {np.sqrt(teacher_number_of_previously_posted_projects_scalar.var_[0])}")

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

```
Mean : 11.153165275336848, Standard deviation : 27.77702641477403
```

In [88]:
```
teacher_number_of_previously_posted_projects_standardized
```

Out[88]:
```
array([[-0.40152481],
       [-0.14951799],
       [-0.36552384],
       ...,
       [-0.29352189],
       [-0.40152481],
       [-0.40152481]])
```

1.4.4 Merging all the above features

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

In [89]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

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

In [90]:

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

Out[90]:

```
(109248, 16663)
```

In [91]:

```python
# Citation: https://cmdlinetips.com/2019/07/how-to-slice-rows-and-columns-of-sparse-matrix-in-python/
# Search String: Slicing throught the scipy matrix.

select_ind = np.array(np.arange(15000)) # Choosing 15 thousand datapoints out of the entire dataset
# The scipy returns data in COO matrix format, which cannot be sliced, so converting into CSR format (Compressed Sparce Row)
# print (X.tocsr()[select_ind,:])
X_small = (X.tocsr()[select_ind,:])
print (X_small.__class__)
```

```
<class 'scipy.sparse.csr.csr_matrix'>
```

In [92]:

```python
# Changing the format back to COO
X_small = X_small.tocoo()
print (X_small.shape)
```

```
(15000, 16663)
```

In [93]:

```python
print (price_standardized.__class__)
```

```
<class 'numpy.ndarray'>
```

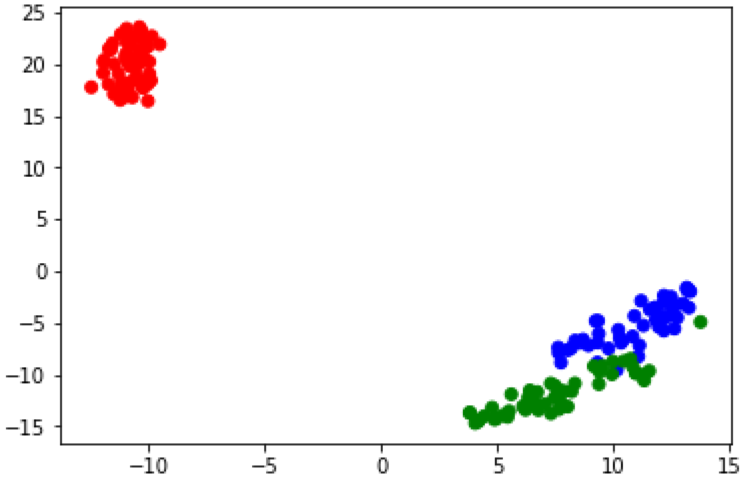
Assignment 2: Apply TSNE

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

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

In [94]:

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

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

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

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

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



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

In [95]:

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

# Gathering all data and buidling the data matrix
# Categorical
print (school_state_one_hot.shape)
print (categories_one_hot.shape)
print (sub_categories_one_hot.shape)
print (teacher_prefix_one_hot.shape)
print (grade_one_hot.shape)
print (text_bow_titles.shape)
# Numerical
print (price_standardized.shape)
print (teacher_number_of_previously_posted_projects_standardized.shape)
```

```
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 5)
(109248, 4)
(109248, 3329)
(109248, 1)
(109248, 1)
```

In [96]:

```
# Data Matrix 1
# with the same hstack function we are concatinating a sparse matrix and a den
se matirx :)
tsne_bow_data_matrix = hstack((school_state_one_hot, categories_one_hot, sub_c
ategories_one_hot,teacher_prefix_one_hot,grade_one_hot, text_bow_titles, price
_standardized, teacher_number_of_previously_posted_projects_standardized))
tsne_bow_data_matrix.shape
```

Out[96]:

```
(109248, 3430)
```

In [97]:

```
# Taking 5 thousand instances of data matrix
select_ind = np.array(np.arange(5000))
tsne_bow_data = (tsne_bow_data_matrix.tocsr()[select_ind,:]).tocoo()
print (tsne_bow_data.shape)
```

```
(5000, 3430)
```

In [104]:

```
# Labels for the above 5000 thousand datapoints
tsne_data_labels = project_data['project_is_approved'][:5000].values
print (tsne_data_labels.shape)
tsne_data_labels.__class__
```

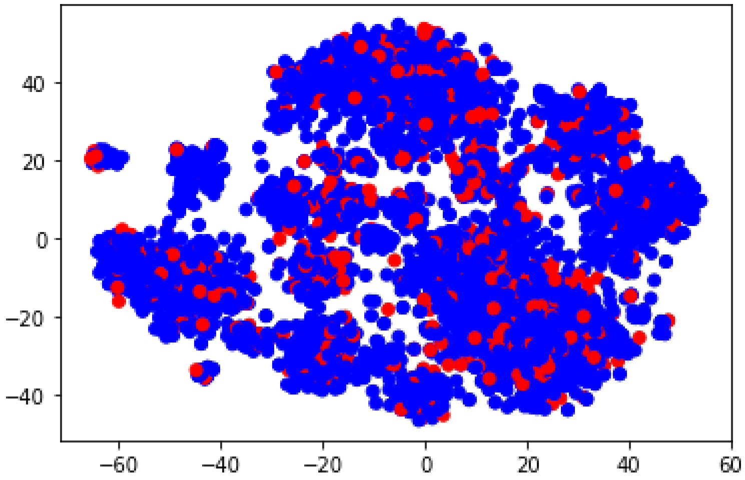
```
(5000,)
```

Out[104]:

```
numpy.ndarray
```

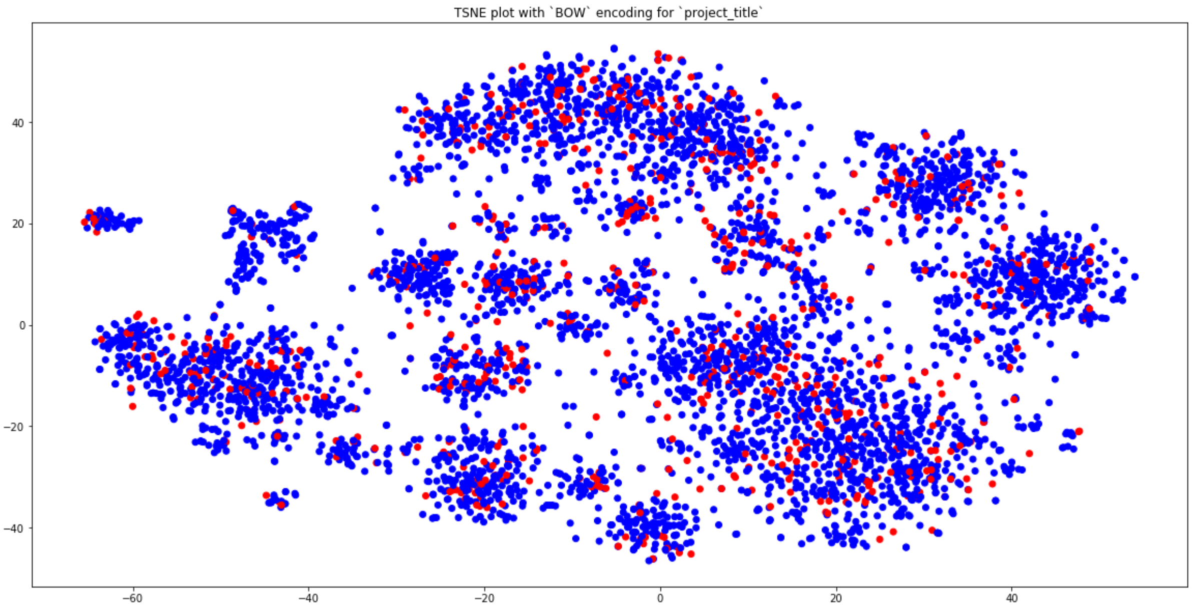
In [105]:

```
# Plotting the TSNE Graph
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)

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

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



In [107]:

```
# Plotting the above same graph with labels and title
plt.figure(figsize=(20,10)) # https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))
plt.xlabel("") # To the best of my knowledge, the x,y axeses of T-SNE plot are not interpretable. They are just projections of high dimension data into lower dimensions
plt.ylabel("") # TSNE just tries to preserve the relative similarities from higher dimensions and projects them into lower dimensions.
plt.title("TSNE plot with `BOW` encoding for `project_title`")
plt.show()
```



Summary: The data is very much scattered with extreme overlaps. Clustering of data is not possible either. So coming a to conclusion is not possible with the TSNE plot where BOW encoding for project_title is performed.

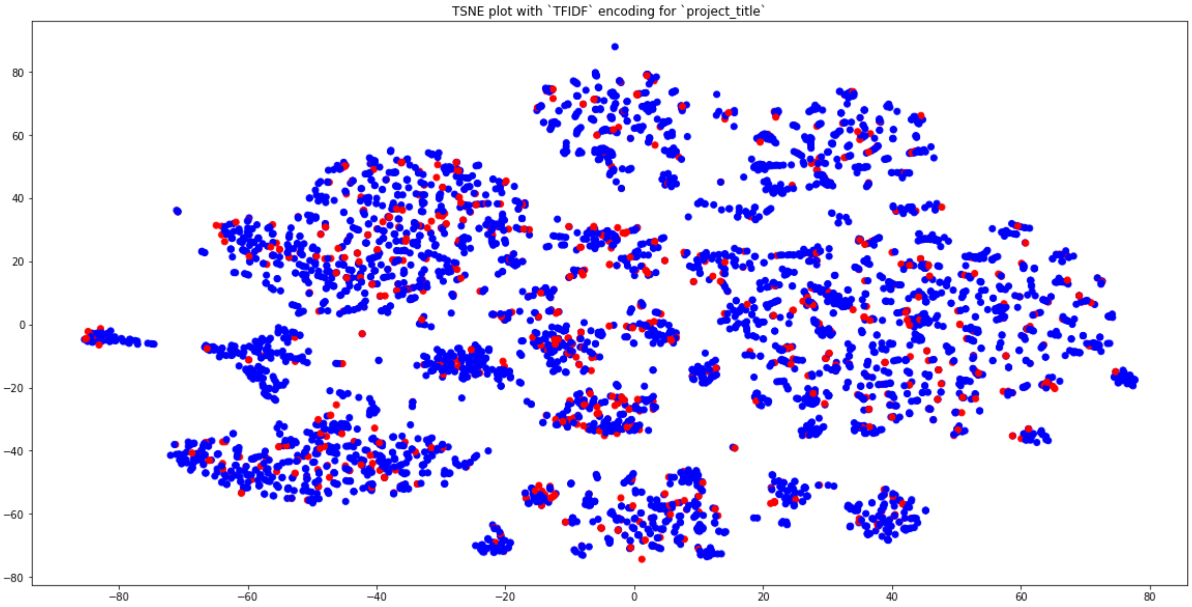
2.2 TSNE with `TFIDF` encoding of `project_title` feature

In [108]:

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

# Gathering all data and buidling the data matrix
# Categorical
print (school_state_one_hot.shape)
print (categories_one_hot.shape)
print (sub_categories_one_hot.shape)
print (teacher_prefix_one_hot.shape)
print (grade_one_hot.shape)
print (text_tfidf_titles.shape)
# Numerical
print (price_standardized.shape)
print (teacher_number_of_previously_posted_projects_standardized.shape)
#
tsne_tfidf_data_matrix = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,teacher_prefix_one_hot,grade_one_hot, text_tfidf_titles, price_standardized, teacher_number_of_previously_posted_projects_standardized))
tsne_tfidf_data_matrix.shape
# taking 5000 instances
tsne_tfidf_data = (tsne_tfidf_data_matrix.tocsr()[select_ind,:]).tocoo()
print (tsne_tfidf_data.shape)

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

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

for_tsne_tfidf = np.hstack((X_embedding, tsne_data_labels.reshape(-1,1)))
for_tsne_tfidf_df = pd.DataFrame(data=for_tsne_tfidf, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
```

```
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 5)
(109248, 4)
(109248, 3329)
(109248, 1)
(109248, 1)
(5000, 3430)
```

In [109]:
```
plt.figure(figsize=(20,10)) # https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
plt.scatter(for_tsne_tfidf_df['Dimension_x'], for_tsne_tfidf_df['Dimension_y'], c=for_tsne_tfidf_df['Score'].apply(lambda x: colors[x]))
plt.title("TSNE plot with `TFIDF` encoding for `project_title`")
plt.show()
```



Summary: Same as the previous TSNE plot. The overlapping points doesn't yield any information to categorize the project as approved-vs-not approved.

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

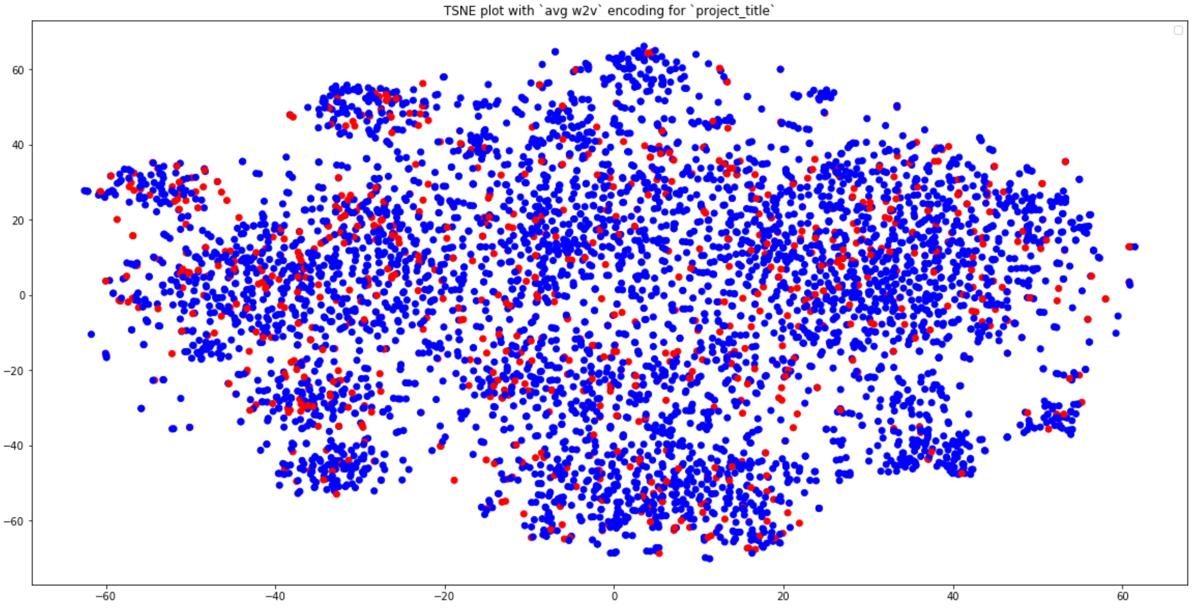
In [110]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
tsne_avgw2v_data_matrix = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,teacher_prefix_one_hot,grade_one_hot, avg_w2v_vectors_titles, price_standardized, teacher_number_of_previously_posted_projects_standardized))
tsne_avgw2v_data_matrix.shape
# taking 5000 instances
tsne_avgw2v_data = (tsne_avgw2v_data_matrix.tocsr()[select_ind,:]).tocoo()
print (tsne_avgw2v_data.shape)

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

X_embedding = tsne.fit_transform(tsne_avgw2v_data.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix

for_tsne_avgw2v = np.hstack((X_embedding, tsne_data_labels.reshape(-1,1)))
for_tsne_avgw2v_df = pd.DataFrame(data=for_tsne_avgw2v, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
```

```
(5000, 401)
```

```
In [112]: plt.figure(figsize=(20,10)) # https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
          plt.scatter(for_tsne_avgw2v_df['Dimension_x'], for_tsne_avgw2v_df['Dimension_y'], c=for_tsne_avgw2v_df['Score'].apply(lambda x: colors[x]))
          plt.title("TSNE plot with `avg w2v` encoding for `project_title`")
          plt.show()
```

```
No handles with labels found to put in legend.
```



Summary: It is impossible to fathom the clustering of similar datapoints from the above plot. Let's try TFIDF Wieghted W2V this time.

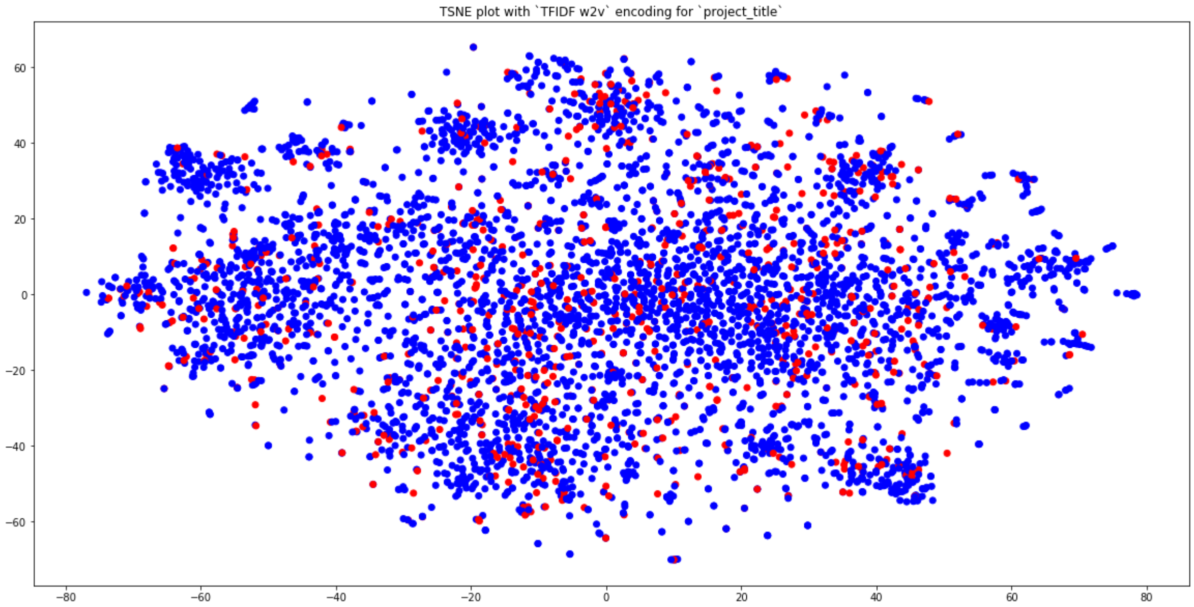
2.4 TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature

In [114]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
tsne_tfidf_w2v_data_matrix = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,teacher_prefix_one_hot,grade_one_hot, tfidf_w2v_vectors_titles, price_standardized, teacher_number_of_previously_posted_projects_standardized))
tsne_tfidf_w2v_data_matrix.shape
# taking 5000 instances
tsne_tfidf_w2v_data = (tsne_tfidf_w2v_data_matrix.tocsr()[select_ind,:]).tocoo()
print (tsne_tfidf_w2v_data.shape)

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

X_embedding = tsne.fit_transform(tsne_tfidf_w2v_data.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix

for_tsne_tfidf_w2v = np.hstack((X_embedding, tsne_data_labels.reshape(-1,1)))
for_tsne_tfidf_w2v_df = pd.DataFrame(data=for_tsne_tfidf_w2v, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
```

```
(5000, 401)
```

In [115]:
```
plt.figure(figsize=(20,10)) # https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
plt.scatter(for_tsne_tfidf_w2v_df['Dimension_x'], for_tsne_tfidf_w2v_df['Dimension_y'], c=for_tsne_tfidf_w2v_df['Score'].apply(lambda x: colors[x]))
plt.title("TSNE plot with `TFIDF w2v` encoding for `project_title`")
plt.show()
```



Summary: Apparently TF-IDF W2V TSNE plot for project_titles also has a lot of overlapping points and it is futile to draw any conclusions. Let's give one more shot in combining all the cateogtical and numerical features and draw a concatinated TSNE plot.

Combined TSNE Plot using all features

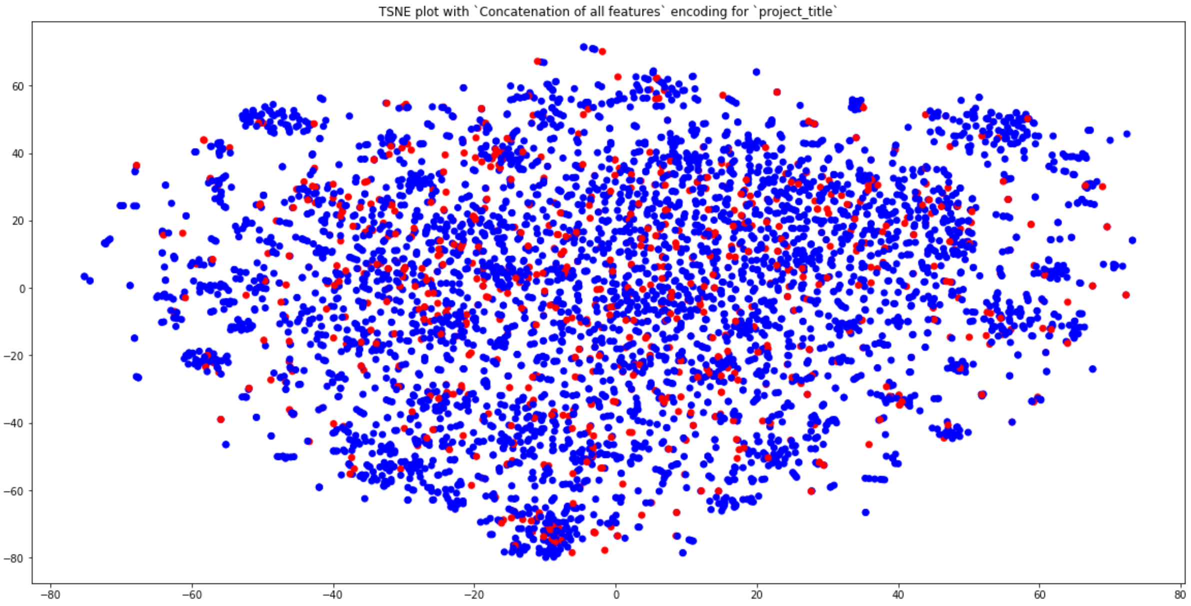
In [116]:

```
# All catgorical+Numeric
tsne_full_data_matrix = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,teacher_prefix_one_hot,grade_one_hot, text_bow_titles, text_tfidf_titles,avg_w2v_vectors_titles, tfidf_w2v_vectors_titles, price_standardized, teacher_number_of_previously_posted_projects_standardized))
tsne_full_data_matrix.shape
# taking 5000 instances
tsne_full_data = (tsne_full_data_matrix.tocsr()[select_ind,:]).tocoo()
print (tsne_full_data.shape)

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

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

for_tsne_full = np.hstack((X_embedding, tsne_data_labels.reshape(-1,1)))
for_tsne_full_df = pd.DataFrame(data=for_tsne_full, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
```

```
(5000, 7359)
```

In [119]:

```
plt.figure(figsize=(20,10)) # https://stackoverflow.com/questions/332289/how-do-you-change-the-size-of-figures-drawn-with-matplotlib
plt.scatter(for_tsne_full_df['Dimension_x'], for_tsne_full_df['Dimension_y'], c=for_tsne_full_df['Score'].apply(lambda x: colors[x]))
plt.title("TSNE plot with `Concatenation of all features` encoding for `project_title`")
plt.show()
```



Summary: Even the combined TSNE plot fails to address the clustering of similar datapoints. Maybe the number of instances that were chosen to draw this plot could be the reason behind this. Only if higher computational efficiencient machine was available, a TSNE plot with more datapoints, a different learning rate or perplexity could yeild a better results.

2.5 Summary

The State "DE" (Delaware) appears to be the one with highest approval rate reaching almost 90% followed by North Dakota and Washington with 88% and 87% respectively. Poor Vermont (VT) it has the lowest acceptance rate with 80% and DC being 80.2 , followed by Texas mearly beating VT by one percent higher.

Every state has greater than 80% success rate in approval There is high variablity in the number of projects submitted for approval. Just look at CA, it has over 15000 submissions and VT just 80.

Most of the proposals on approved/rejected side are sent by Mrs. and least by Dr. Like they say Women dominate in every field. The number of proposals made by Mrs and Ms are nearly 7 times the number of male submissions.

Summary: Most number of proposals are sent for Grades PreK-2, but the highest acceptance rate is for Grades 3-5, with lowest approval rate of around 83.7% for Grades 9-12 and an overall avg acceptance rate of 84%

The first and second position categories for approved projects are backed by Literacy_Language, Math_science 52239, 41421 respectively. Lowest being Warmth just a mear 1388.

Based on the above plot, the highest number of submissions category goes for Literacy_Language and lowest submissions category being Appliedlearning Math_Science. From the plot we can conclude that Literacy and language are given higher importance compared to applied skills and math. Also when it comes to hunger and warmth, no society would deny a submission, which is eveident from the highest 92.5% acceptance rate.

1) Plot literally showcases that the subcategory with highest submissions and acceptance is non other than Literacy. 2) Since most of the students are in school under kindergarten any project for college and careerprep doesn't make much sense, well that is what the graph shows with just 81% acceptance rate

The max length of the title appears to be 13, with smallest being around 4 words. Very few projects have massive title lengths, most of them have very few words around 4 to 6.

If the number of words in essay are fewer the chances of its rejection are higher.

When the words in title are too less, the rejection rate is higher.

People are always sceptical in matters that costs too much. The graph kind of tells the same. When the cost of project is high, it's approval rate is low.

People are always sceptical in matters that costs too much. The graph kind of tells the same. When the cost of project is high, it's approval rate is low.

It is evident that there are certain extreme points in accepted box plot. But when we look at the 50th percentile approx 20 words in summary, the approval rate is higher than rejection rate, and as the number of words in resource summary increase beyond 30+, the rejection rate is higher.

It is evident that when the summary contains numerics, the proposal acceptance rate is higher almost an avg of 89% compared to the cases where the acceptance rate is mere 84% when there is no numeric data in summary.

None of the TSNE plots drawn above can give solid base for clustering or grouping similar datapoints

In [ ]: