DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed p **Example:** `p036502`

Feature	Description
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the follow enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) su categories for the project from the fo enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Ma & Science`

Feature	Description
school_state	State where school is located (Two-l U.S. postal code). **Example:** WY
project_subject_subcategories	One or more (comma-separated) su subcategories for the project. **Exam** • Literacy • Literature & Writing, Social Sciences
project_resource_summary	An explanation of the resources nee the project. **Example:** • My students need hands literacy materials to manage sensory needs!
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application w submitted. **Example:** 2016-04-28 12:43:56.245
teacher_id	A unique identifier for the teacher of proposed project. **Example:** bdf8baa8fedef6bfeec7ae4ff1c

Feature	Description
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previ submitted by the same teacher. **Exa** 2

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_4:__ "Close by sharing why your project will make a difference"

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [3]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
```

```
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

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

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

1.1 Reading Data

In [4]:
```
import os

dirpath = os.getcwd()
print("current directory is : " + dirpath)
project_data = pd.read_csv('train_data.csv')
#project_data = pd.read_csv('test_data.csv')
resource_data = pd.read_csv('resources.csv')
```

```
current directory is : C:\Users\ssinghai\Downloads
```

In [151]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :\n", project_data.columns.values)
print('-'*50)
print("The attributes of project data :", project_data.head())
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data :
 ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
--------------------------------------------------
The attributes of project data :    Unnamed: 0       id
        teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc          Mrs.

1      140945  p258326  897464ce9ddc600bced1151f324dd63a           Mr.

2       21895  p182444  3465aaf82da834c0582ebd0ef8040ca0           Ms.

3          45  p246581  f3cb9bffbba169bef1a77b243e620b60          Mrs.

4      172407  p104768  be1f7507a41f8479dc06f047086a39ec          Mrs.
```

```
  school_state project_submitted_datetime project_grade_category  \
0           IN        2016-12-05 13:43:57          Grades PreK-2
1           FL        2016-10-25 09:22:10             Grades 6-8
2           AZ        2016-08-31 12:03:56             Grades 6-8
3           KY        2016-10-06 21:16:17          Grades PreK-2
4           TX        2016-07-11 01:10:09          Grades PreK-2

            project_subject_categories     project_subject_subcategorie
s  \
0                   Literacy & Language                      ESL, Literac
y
1    History & Civics, Health & Sports  Civics & Government, Team Sport
s
2                      Health & Sports    Health & Wellness, Team Sport
s
3  Literacy & Language, Math & Science              Literacy, Mathematic
s
4                       Math & Science                       Mathematic
s

                                      project_title  \
0   Educational Support for English Learners at Home
1              Wanted: Projector for Hungry Learners
2  Soccer Equipment for AWESOME Middle School Stu...
3                            Techie Kindergarteners
4                            Interactive Math Tools

                                     project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...
2  \r\n\"True champions aren't always the ones th...
3  I work at a unique school filled with both ESL...
4  Our second grade classroom next year will be m...

                                     project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...             NaN
1  The projector we need for our school is very c...             NaN
```

```
2  The students on the campus come to school know...             NaN
3  My students live in high poverty conditions wi...             NaN
4  For many students, math is a subject that does...             NaN

  project_essay_4                          project_resource_summary  \
0             NaN  My students need opportunities to practice beg...
1             NaN  My students need a projector to help with view...
2             NaN  My students need shine guards, athletic socks,...
3             NaN  My students need to engage in Reading and Math...
4             NaN  My students need hands on practice in mathemat...

   teacher_number_of_previously_posted_projects  project_is_approved
0                                             0                    0
1                                             7                    1
2                                             1                    0
3                                             4                    1
4                                             1                    1
```

In [152]:
```
print("Number of data points in resource data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[152]:

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

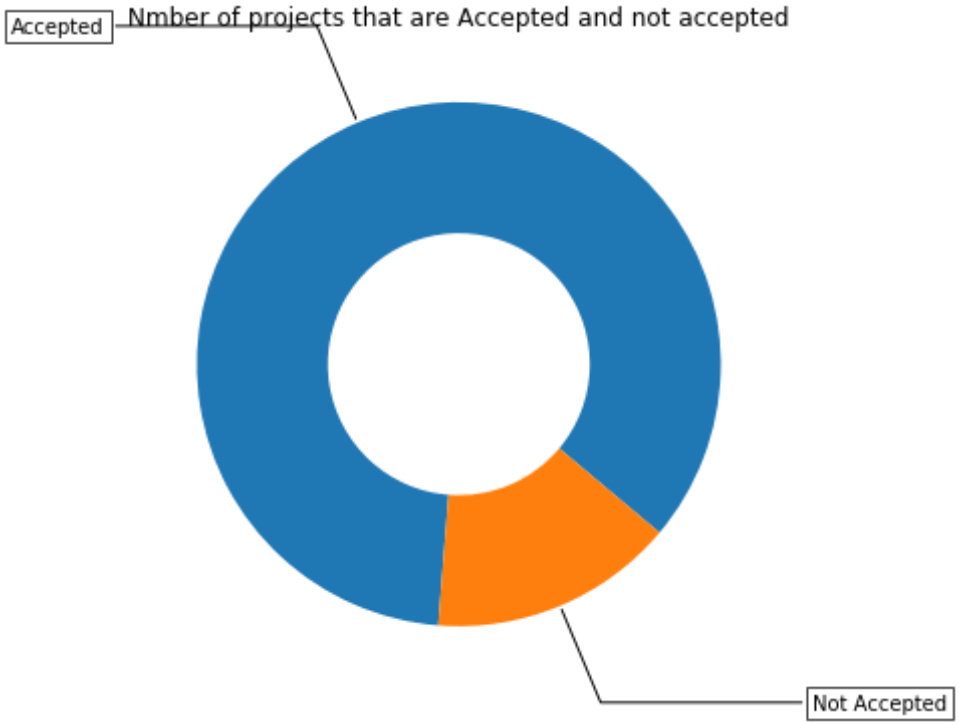
1.2 Data Analysis

In [153]:
```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_lab
els.html#sphx-glr-gallery-pie-and-polar-charts-pie-and-donut-labels-py
```

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

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

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

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

bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
kw = dict(xycoords='data', textcoords='data', arrowprops=dict(arrowstyl
e="-"),
          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 , ( 84.85830404
22 %)
Number of projects thar are not approved for funding  16542 , ( 15.1416
```

959578 %)



In [ ]:
```
SUMMARY : This is an imbalanced data set
```

1.2.1 Univariate Analysis: School State

In [154]:
```
# 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()
```

```
# if you have data which contain only 0 and 1, then the mean = percenta
ge (think about it)
temp.columns = ['state_code', 'num_proposals']

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

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(1
88,189,220)'],\
            [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')
'''
```

Out[154]:
```
'# How to plot US state heatmap: https://datascience.stackexchange.com/
```

```
a/9620\n\nscl = [[0.0, \'rgb(242,240,247)\'],[0.2, \'rgb(218,218,235)
\'],[0.4, \'rgb(188,189,220)\'],            [0.6, \'rgb(158,154,200)
\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\ndata = [
dict(\n        type=\'choropleth\',\n        colorscale = scl,\n
autocolorscale = False,\n        locations = temp[\'state_code\'],\n
     z = temp[\'num_proposals\'].astype(float),\n        locationmode =
\'USA-states\',\n        text = temp[\'state_code\'],\n        marker =
dict(line = dict (color = \'rgb(255,255,255)\',width = 2)),\n        co
lorbar = dict(title = "% of pro")\n    ) ]\n\nlayout = dict(\n        t
itle = \'Project Proposals % of Acceptance Rate by US States\',\n
  geo = dict(\n            scope=\'usa\',\n            projection=dict(
type=\'albers usa\' ),\n            showlakes = True,\n            lake
color = \'rgb(255, 255, 255)\',\n        ),\n    )\n\nfig = go.Figure(d
ata=data, layout=layout)\noffline.iplot(fig, filename=\'us-map-heat-map
\')\n'
```

In [155]:

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

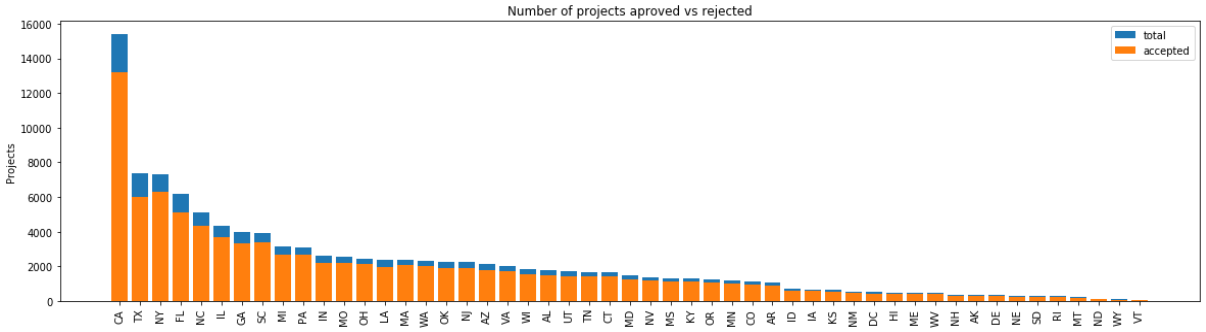
```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.800000
7          DC       0.802326
43         TX       0.813142
26         MT       0.816327
18         LA       0.831245
==================================================
States with highest % approvals
   state_code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

In [161]:
```
#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(rotation=90)

    plt.xticks(ind, list(data[xtick].values))

    plt.xticks(rotation=90)

    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()
```

In [162]:
```
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 [163]:
```
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

==========

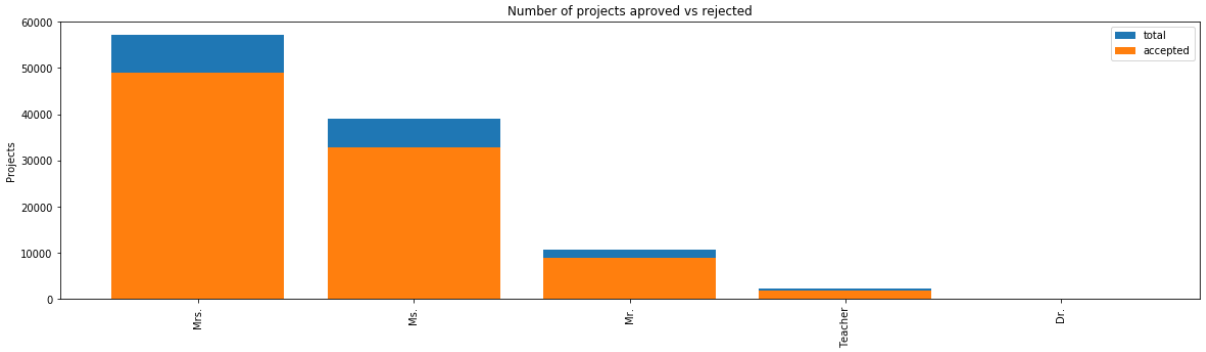
	school_state	project_is_approved	total	Avg
39	RI	243	285	0.852632
26	MT	200	245	0.816327
28	ND	127	143	0.888112
50	WY	82	98	0.836735
46	VT	64	80	0.800000

SUMMARY:

1. Every state has greater than 80% success rate in approval
2. Hishest number of project submission were done from the state of CA
3. Lowest number of project submission came form state VT

1.2.2 Univariate Analysis: teacher_prefix

In [164]:
```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)
```



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

==========

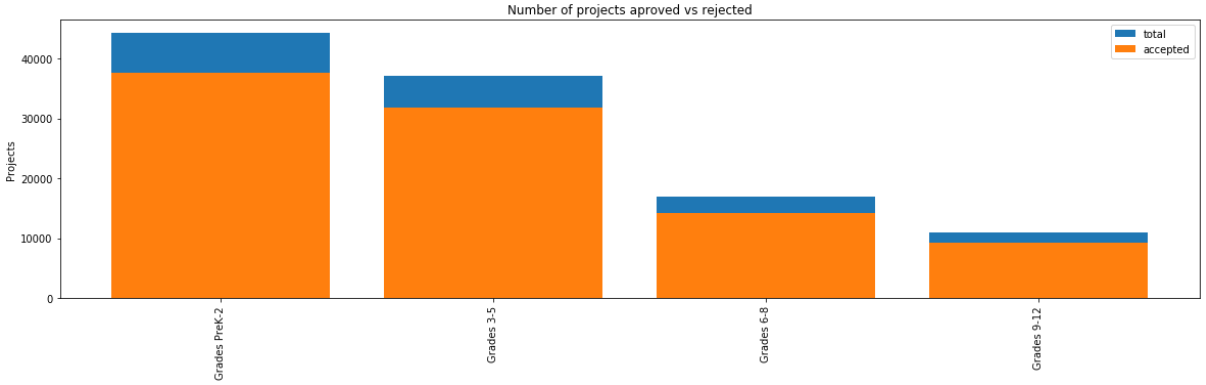
	teacher_prefix	project_is_approved	total	Avg
2	Mrs.	48997	57269	0.855559
3	Ms.	32860	38955	0.843537
1	Mr.	8960	10648	0.841473
4	Teacher	1877	2360	0.795339
0	Dr.	9	13	0.692308

SUMMARY :

1. Project submitted by females (Mrs and Ms) is very high as compared to males (Mr)
2. Success rate for Mrs, Ms and Mr is ~85% while for Dr title it is least

1.2.3 Univariate Analysis: project_grade_category

In [165]:
```
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:

1. Number project submissions decrease as the Grades increase
2. Project approval rate is ~85 for all grades

1.2.4 Univariate Analysis: project_subject_categories

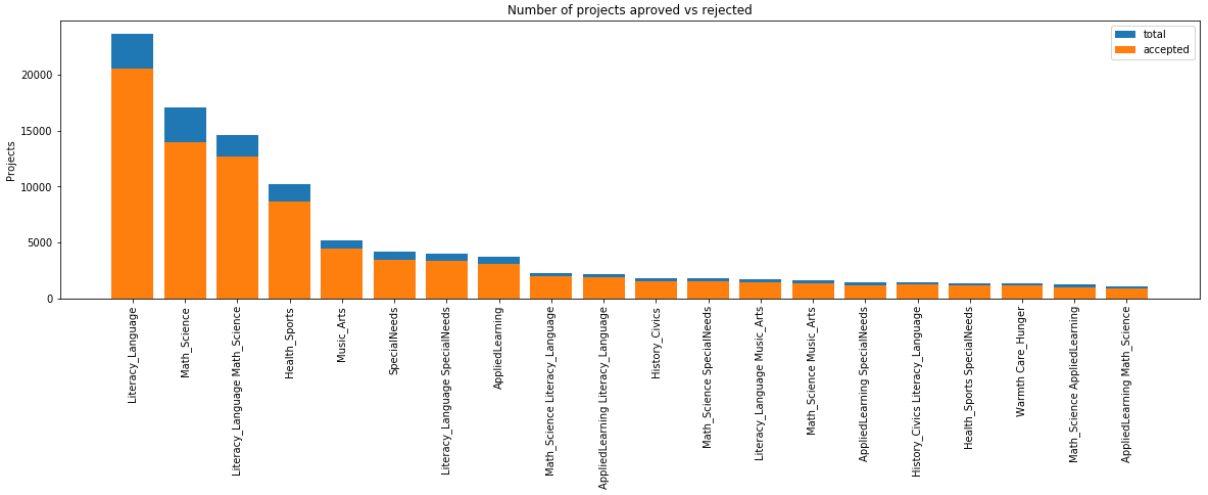
In [166]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stacko
verflow.com/a/47301924/4084039

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

In [167]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[167]:

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

In [168]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=20)
```



	clean_categories	project_is_approved	total	Avg
24	Literacy_Language	20520	23655	0.86747
0				
32	Math_Science	13991	17072	0.81952

```
9
28  Literacy_Language Math_Science                12725  14636  0.86943
2
8                     Health_Sports                 8640  10177  0.84897
3
40                       Music_Arts                 4429   5180  0.85501
9
==================================================
                     clean_categories  project_is_approved  total
Avg
19  History_Civics Literacy_Language                 1271   1421  0.894
441
14        Health_Sports SpecialNeeds                 1215   1391  0.873
472
50                Warmth Care_Hunger                 1212   1309  0.925
898
33      Math_Science AppliedLearning                 1019   1220  0.835
246
4       AppliedLearning Math_Science                  855   1052  0.812
738
```

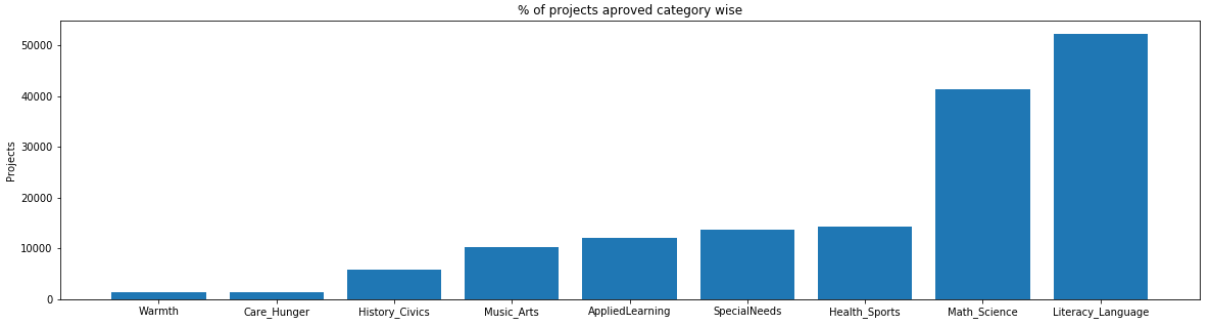
SUMMARY:

1. AppliedLearning Math_Science has lowest submissions
2. Warmth Care_Hunger has highest approval rate

In [169]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

In [170]:
```
# 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 [171]:
```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

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

1.2.5 Univariate Analysis: project_subject_subcategories

In [172]:
```
sub_catogories = list(project_data['project_subject_subcategories'].val
```

```
ues)
# remove special characters from list of strings python: https://stacko
verflow.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 & H
unger"
    for j in i.split(','): # it will split it in three parts ["Math & S
cience", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory b
ased on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are g
oing 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 [173]:

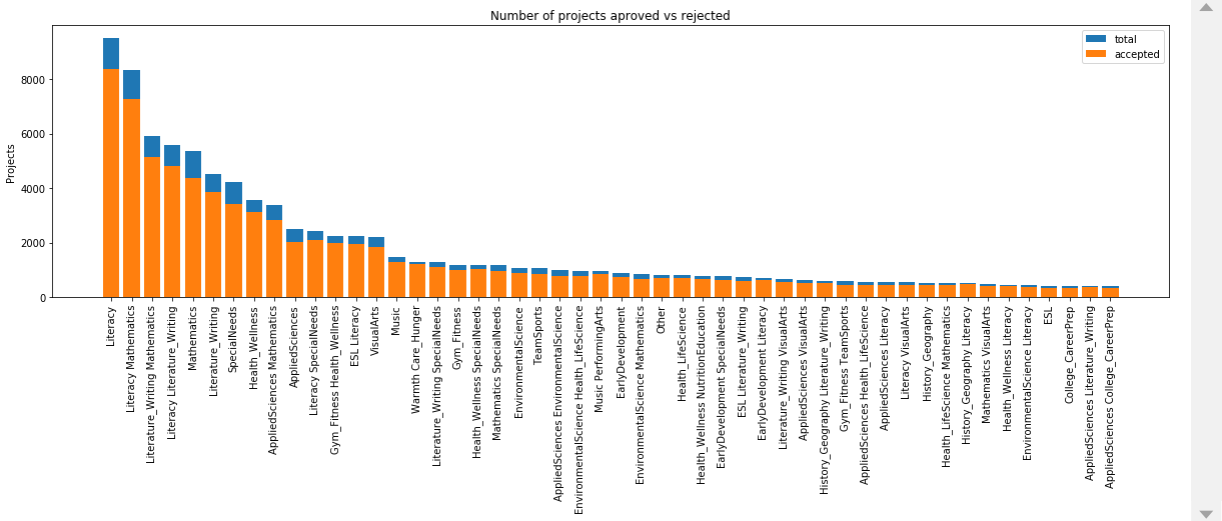
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=Tr
ue)
project_data.head(2)
```

Out[173]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state

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

In [174]:
```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top=50)
```



	clean_subcategories	project_is_approved	total	
Avg 317	Literacy	8371	9486	0.88

```
2458
319            Literacy Mathematics                 7260   8325  0.87
2072
331  Literature_Writing Mathematics                 5140   5923  0.86
7803
318     Literacy Literature_Writing                 4823   5571  0.86
5733
342                      Mathematics                4385   5379  0.81
5207
==================================================
                    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
```

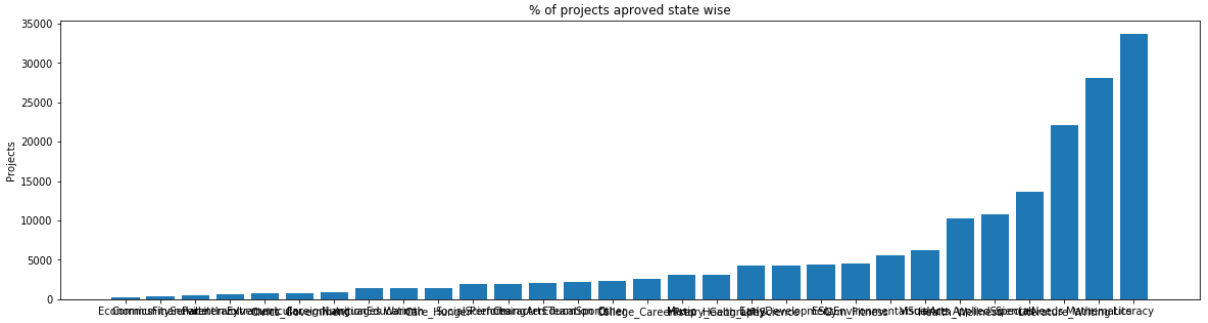
SUMMARY:

1. Literacy has the highest project submissions
2. AppliedSciences College_CareerPrep has the lowest project submissions
3. For all subcategories approval is above 80%

In [175]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [176]:
```
# 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()
```



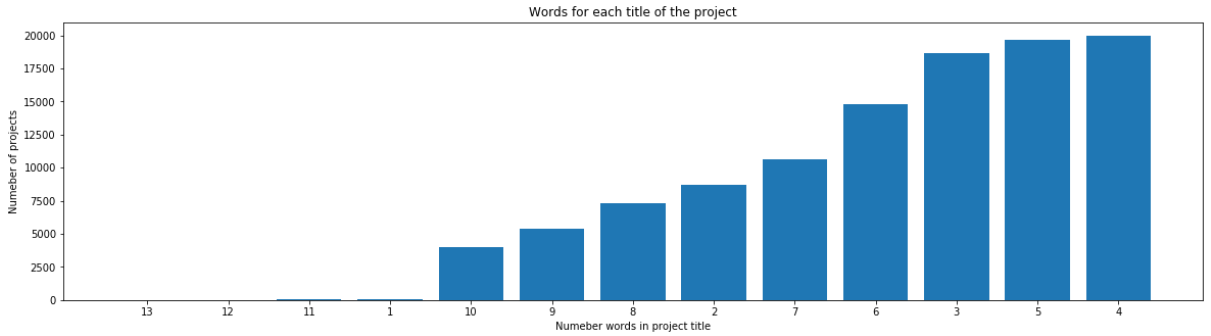
In [177]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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
```

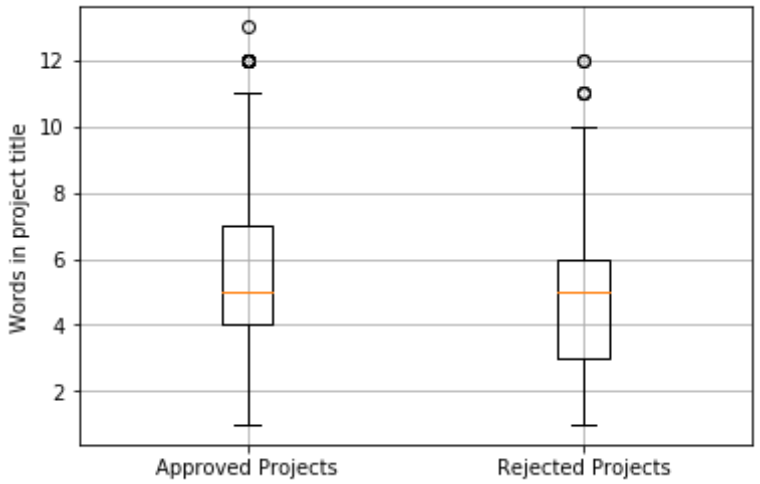
1.2.6 Univariate Analysis: Text features (Title)

In [178]:
```
#How to calculate number of words in a string in DataFrame: https://sta
ckoverflow.com/a/37483537/4084039
word_count = project_data['project_title'].str.split().apply(len).value
_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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

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

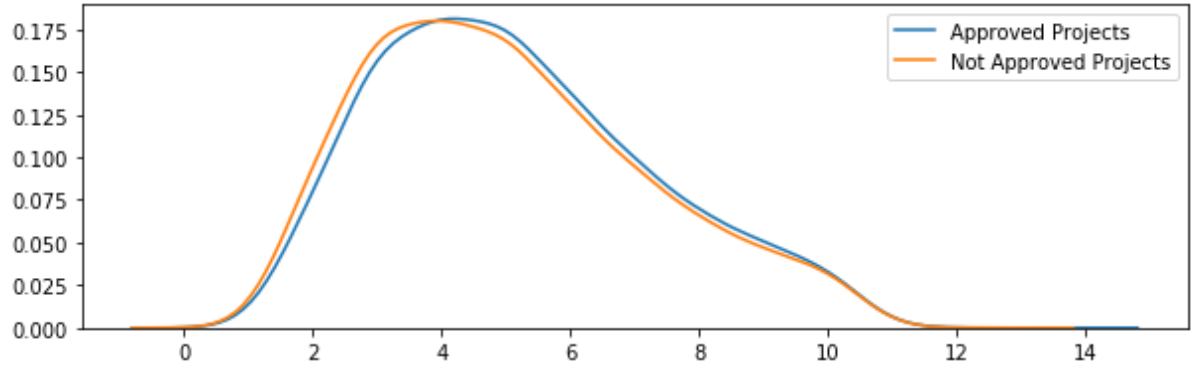


In [179]:
```
approved_title_word_count = project_data[project_data['project_is_approved']==1]['project_title'].str.split().apply(len)
approved_title_word_count = approved_title_word_count.values

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 [180]:
```
# 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 [181]:
```
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:

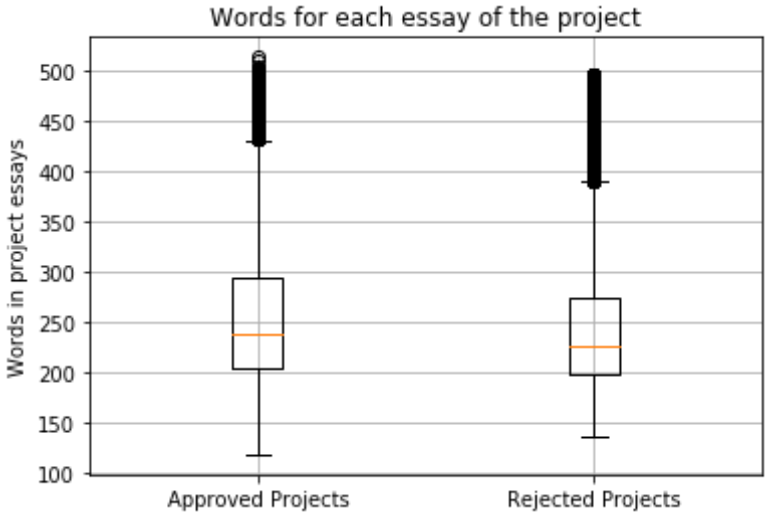
1. Project is rejected when it has either when it has very less words i.e. <1 or it has more words i.e. >12

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

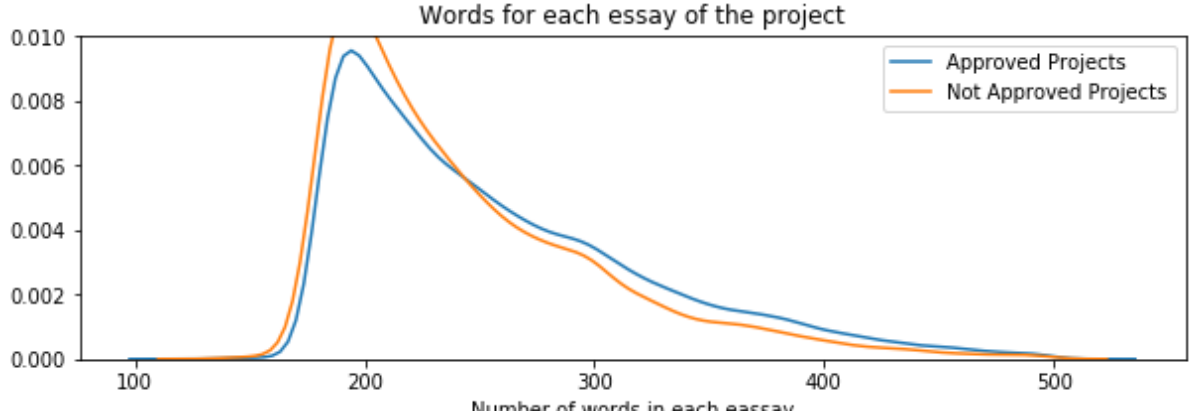
In [182]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [183]:
```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.split().apply(len)
approved_word_count = approved_word_count.values

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

In [184]:
```
# 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 [185]:
```
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 Proje
cts")
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```



Number of words in each essay

SUMMARY:

1. Highest rejection is when the essay has around 200 words
2. Highest approval is also at 200 words
3. When the wordsa re between 230 to 470 project approval rate is higher

1.2.8 Univariate Analysis: Cost per project

In [186]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[186]:

	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 [187]:
```
# 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[187]:

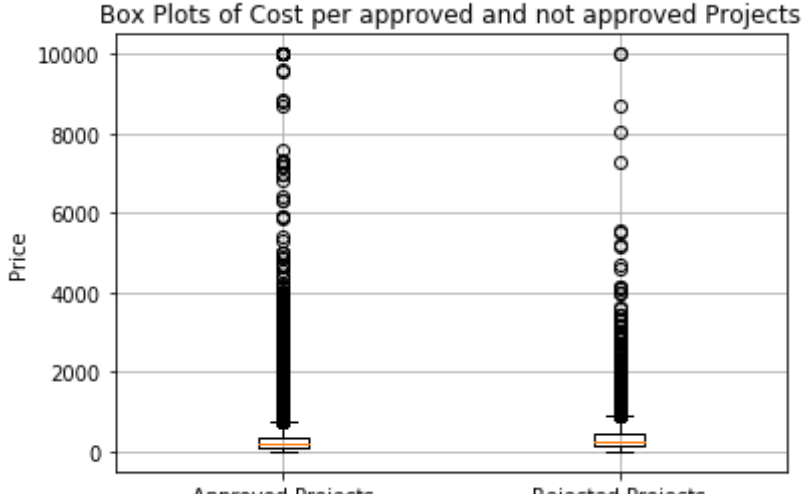
	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [188]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
print(project_data.columns.values)
```

```
['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category' 'project_title'
 'project_essay_1' 'project_essay_2' 'project_essay_3' 'project_essay_
4'
 'project_resource_summary' 'teacher_number_of_previously_posted_projec
ts'
 'project_is_approved' 'clean_categories' 'clean_subcategories' 'essay'
 'price' 'quantity']
```

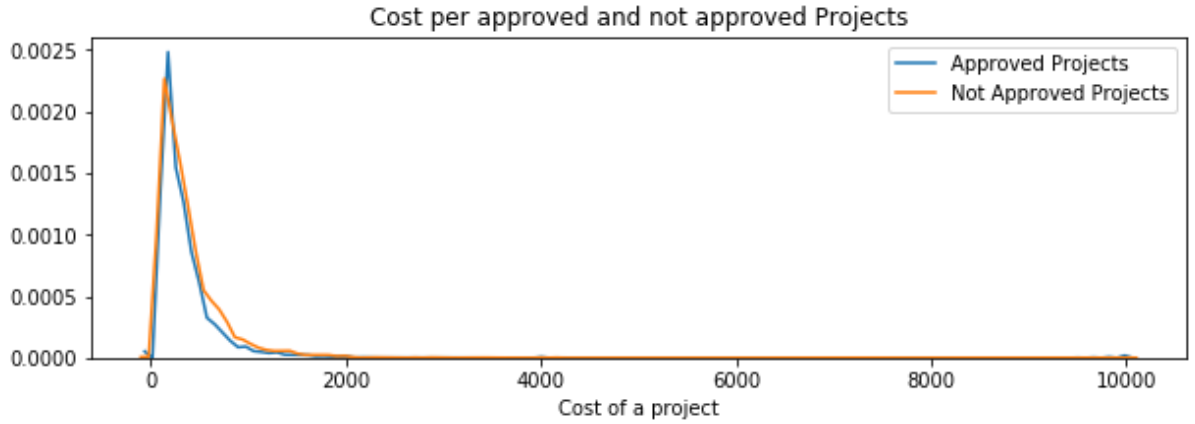
In [189]:
```
approved_price = project_data[project_data['project_is_approved']==1][
'price'].values

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

In [190]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.ht
ml
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()
```



Approved Projects Rejected Projects

In [191]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_price, hist=False, label="Approved Projects")
sns.distplot(rejected_price, hist=False, label="Not Approved Projects")
plt.title('Cost per approved and not approved Projects')
plt.xlabel('Cost of a project')
plt.legend()
plt.show()
```



In [192]:
```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pi
p3 install prettytable

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

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

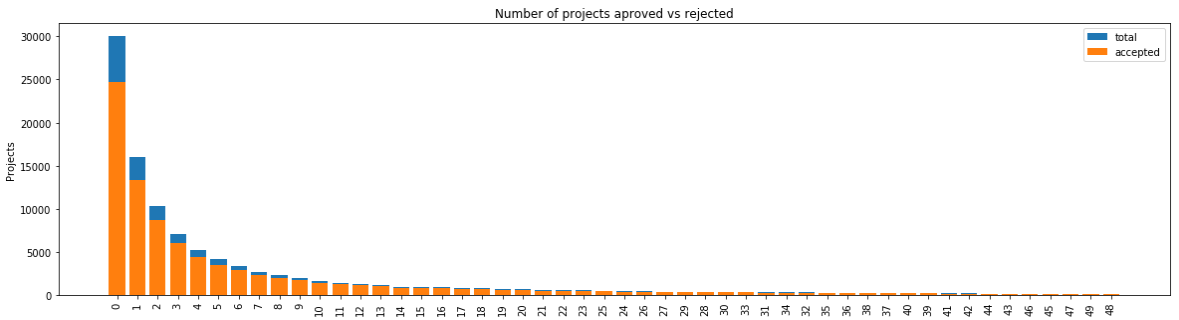
SUMMARY:

1. Projects with Low cost are more approved

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 [195]:
```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_
projects', 'project_is_approved' , top=50)
```

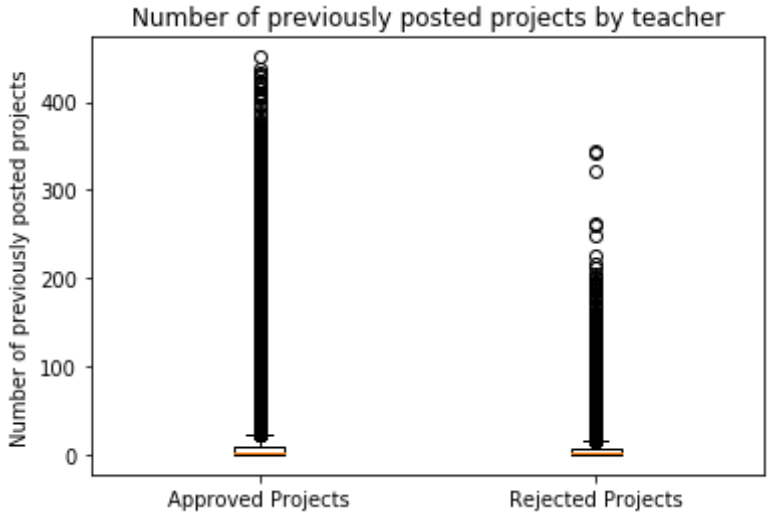



```
   teacher_number_of_previously_posted_projects  project_is_approved
total  \
0                                             0                24652
30014
1                                             1                13329
16058
2                                             2                 8705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

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved
  total  \
46                                           46                  149
    164
45                                           45                  141
    153
47                                           47                  129
```

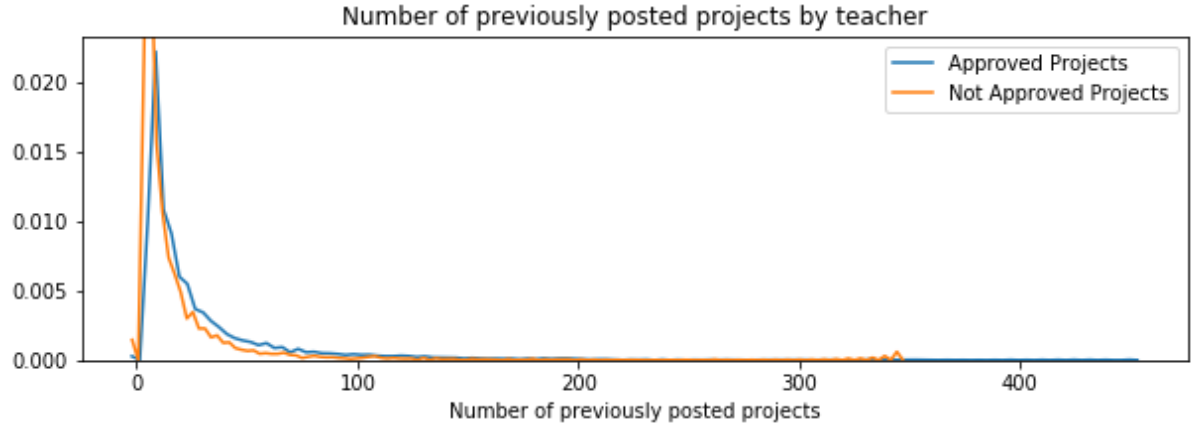
```
    144
49                                  49                  128
    143
48                                  48                  135
    140

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

In [300]:
```
approved_previously_posted_projects_count=project_data[project_data['pr
oject_is_approved']==1]['teacher_number_of_previously_posted_projects']
.values
rejected_previously_posted_projects_count=project_data[project_data['pr
oject_is_approved']==0]['teacher_number_of_previously_posted_projects']
.values
```

In [301]:
```
plt.boxplot([approved_previously_posted_projects_count,rejected_previou
sly_posted_projects_count],autorange=True)
plt.title('Number of previously posted projects by teacher')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel("Number of previously posted projects")
plt.show()
```



In [302]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_previously_posted_projects_count, hist=False, lab
el="Approved Projects")
sns.distplot(rejected_previously_posted_projects_count, hist=False, lab
el="Not Approved Projects")
plt.title('Number of previously posted projects by teacher')
plt.xlabel('Number of previously posted projects')
plt.legend()
plt.show()
```



In [303]:
```
# http://zetcode.com/python/prettytable/
x = PrettyTable()
x.field_names = ["Percentile", "Approved Projects", "Not Approved Proje
cts"]

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

```
+------------+-------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+-------------------+-----------------------+
|     0      |        0.0        |          0.0          |
|     5      |        0.0        |          0.0          |
|     10     |        0.0        |          0.0          |
|     15     |        0.0        |          0.0          |
|     20     |        0.0        |          0.0          |
|     25     |        0.0        |          0.0          |
|     30     |        1.0        |          0.0          |
|     35     |        1.0        |          1.0          |
|     40     |        1.0        |          1.0          |
|     45     |        2.0        |          1.0          |
|     50     |        2.0        |          2.0          |
|     55     |        3.0        |          2.0          |
|     60     |        4.0        |          3.0          |
|     65     |        5.0        |          3.0          |
|     70     |        7.0        |          4.0          |
|     75     |        9.0        |          6.0          |
|     80     |        13.0       |          8.0          |
|     85     |        19.0       |          11.0         |
|     90     |        30.0       |          17.0         |
|     95     |        57.0       |          31.0         |
|    100     |       451.0       |         345.0         |
+------------+-------------------+-----------------------+
```

SUMMARY:

1. Selection rate is greater than 80% for all projects
2. Numbers of new teachers who submitted the projects for the first time were more than the teachers who had submitted the project in the past

1.2.10 Univariate Analysis: project_resource_summary

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

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

In [306]:
```
# printing some random project_resource_summary.
print(project_data['project_resource_summary'].values[0])
print("="*50)
print(project_data['project_resource_summary'].values[150])
print("="*50)
print(project_data['project_resource_summary'].values[1000])
print("="*50)
print(project_data['project_resource_summary'].values[20000])
print("="*50)
print(project_data['project_resource_summary'].values[99999])
print("="*50)
```

```
My students need opportunities to practice beginning reading skills in
English at home.
==================================================
My students need 5 Hokki stools to increase their movement even while s
itting.
==================================================
My students need nautical themed items such as red throw pillows and ph
oto booth props for a great start to a new 4th grade year!
==================================================
```

```
My students need wobble chairs, number toss games and colors and shapes
mats to make our learning fun, hands on and physically engaging!
==================================================
My students need a CD bluetooth player so they can hear their music cle
arly and I can already have it on my phone for great transitions. Plus
the flannel/ easel for mobility.
==================================================
```

In [6]:
```
df1 = project_data[project_data['project_resource_summary'].str.contain
s('[0-9]')]
print(df1[['project_resource_summary','project_is_approved']][0:5])
print(df1.shape)
```

```
                            project_resource_summary  project_is_appro
ved
12  My students need 3D and 4D life science activi...
  0
14  My students need 5 tablets for our classroom t...
  0
16  My students need 2 LeapPad that will engage th...
  1
19  My students need 7 Hokki stools to encourage a...
  1
25  My students need the learning centers and mult...
  0
(15756, 17)
```

In [312]:
```
#https://erikrood.com/Python_References/count_frequency_value_occurs_da
taframe_final.html
count_num_approved= df1.groupby('project_is_approved').size()

print(count_num_approved)

print("\nTotal projects with a number in 'project_resource_summary': ",
 len(df1.index))

print("\nPercentage of approved projects among the ones with numeric va
lues:", count_num_approved*100/len(df1.index))
```

```
print("\nPercentage of approved projects among all the projects submitted:", count_num_approved*100/len(project_data.index))
```

```
project_is_approved
0     1666
1    14090
dtype: int64

Total projects with a number in 'project_resource_summary':  15756

Percentage of approved projects among the ones with numeric values: project_is_approved
0    10.57375
1    89.42625
dtype: float64

Percentage of approved projects among all the projects submitted: project_is_approved
0     1.524971
1    12.897261
dtype: float64
```

SUMMARY:

1. Close to 90% projects were approved among the projects which had numeric value in 'project_resource_summary'
2. Overall 12% projects were approved among the projects which had numeric value in 'project_resource_summary'; Having numeric value in 'project_resource_summary' does not increase the chances of project getting approved

1.3 Text preprocessing

1.3.1 Essay Text

In [222]:
```
project_data.head(2)
```

Out[222]:

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

In [223]:
```
# 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 s
econd 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 experiences to us that open our eyes to new cultures, beliefs, and respect.\"The 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 along side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading 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 will be specially chosen by the English Learner Teacher and will be sent home regularly to watch. The videos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\nnannan
==========
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together 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 festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity.My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special 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 the day they will be used by the students who need the

highest amount of movement in their life in order to stay focused on sc
hool.\r\n\r\nWhenever asked what the classroom is missing, my students
always say more Hokki Stools. They can't get their fill of the 5 stools
we already have. When the students are sitting in group with me on the
Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki St
ools are the first to be taken. There are always students who head over
to the kidney table to get one of the stools who are disappointed as th
ere 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 stud
ents to 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 core muscles for balance while they sit. For many of my
students, these chairs will take away the barrier that exists in school
s for a child who can't sit still.nannan
==========
How do you remember your days of school? Was it in a sterile environmen
t with plain walls, rows of desks, and a teacher in front of the room?
A typical day in our room is nothing like that. I work hard to create a
warm inviting themed room for my students look forward to coming to eac
h day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and gir
ls of mixed races in Arkansas.\r\nThey attend a Title I school, which m
eans 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 classrooms. These 9 and 10
year-old students are very eager learners; they are like sponges, absor
bing all the information and experiences and keep on wanting more.With
these resources such as the comfy red throw pillows and the whimsical n
autical hanging decor and the blue fish nets, I will be able to help cr
eate the mood in our classroom setting to be one of a themed nautical e
nvironment. Creating a classroom environment is very important in the s
uccess in each and every child's education. The nautical photo props wi
ll be used with each child as they step foot into our classroom for the
first time on Meet the Teacher evening. I'll take pictures of each chil
d 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 thank you cards will
be used throughout the year by the students as they create thank you ca
rds to their team groups.\r\n\r\nYour generous donations will help me t

o help make our classroom a fun, inviting, learning environment from da
y one.\r\n\r\nIt costs lost of money out of my own pocket on resources
to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==========
My kindergarten students have varied disabilities ranging from speech a
nd language delays, cognitive delays, gross/fine motor delays, to autis
m. They are eager beavers and always strive to work their hardest worki
ng past their limitations. \r\n\r\nThe materials we have are the ones I
seek out for my students. I teach in a Title I school where most of the
students receive free or reduced price lunch. Despite their disabiliti
es and limitations, my students love coming to school and come eager to
learn and explore.Have you ever felt like you had ants in your pants an
d you needed to groove and move as you were in a meeting? This is how m
y 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 d
evelop their core, which enhances gross motor and in Turn fine motor sk
ills. \r\nThey also want to learn through games, my kids don't want to
sit and do worksheets. They want to learn to count by jumping and playi
ng. 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 tea
cher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\n
My school has 803 students which is makeup is 97.6% African-American, m
aking up the largest segment of the student body. A typical school in D
allas is made up of 23.2% African-American students. Most of the studen
ts are on free or reduced lunch. We aren't receiving doctors, lawyers,
or engineers children from rich backgrounds or neighborhoods. As an edu
cator I am inspiring minds of young children and we focus not only on a
cademics but one smart, effective, efficient, and disciplined students
with good character.In our classroom we can utilize the Bluetooth for s
wift transitions during class. I use a speaker which doesn't amplify th
e 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 the
lessons as meaningful. But with the bluetooth speaker my students will
be able to hear and I can stop, pause and replay it at any time.\r\nThe
cart will allow me to have more room for storage of things that are nee

```
ded for the day and has an extra part to it I can use.  The table top c
hart has all of the letter, words and pictures for students to learn ab
out different letters and it is more accessible.nannan
==================================================
```

In [224]:
```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 [225]:
```python
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech a
nd language delays, cognitive delays, gross/fine motor delays, to autis
m. They are eager beavers and always strive to work their hardest worki
ng past their limitations. \r\n\r\nThe materials we have are the ones I
seek out for my students. I teach in a Title I school where most of the
students receive free or reduced price lunch.  Despite their disabiliti
es and limitations, my students love coming to school and come eager to
learn and explore.Have you ever felt like you had ants in your pants an
d you needed to groove and move as you were in a meeting? This is how m
y 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 d
```

```
evelop their core, which enhances gross motor and in Turn fine motor sk
ills. \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 playi
ng. 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 [226]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remov
e-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 a
nd language delays, cognitive delays, gross/fine motor delays, to autis
m. They are eager beavers and always strive to work their hardest worki
ng past their limitations.     The materials we have are the ones I see
k out for my students. I teach in a Title I school where most of the st
udents receive free or reduced price lunch.  Despite their disabilities
and limitations, my students love coming to school and come eager to le
arn and explore.Have you ever felt like you had ants in your pants and
you needed to groove and move as you were in a meeting? This is how my
kids feel all the time. The want to be able to move as they learn or so
they say.Wobble chairs are the answer and I love then because they deve
lop their core, which enhances gross motor and in Turn fine motor skill
s.   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 colo
r 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 [227]:
```
#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 a
nd language delays cognitive delays gross fine motor delays to autism T
```

```
hey are eager beavers and always strive to work their hardest working p
ast their limitations The materials we have are the ones I seek out for
my students I teach in a Title I school where most of the students rece
ive free or reduced price lunch Despite their disabilities and limitati
ons my students love coming to school and come eager to learn and explo
re Have you ever felt like you had ants in your pants and you needed to
groove and move as you were in a meeting This is how my kids feel all t
he 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 w
hich 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 The
y want to learn to count by jumping and playing Physical engagement is
the key to our success The number toss and color and shape mats can mak
e that happen My students will forget they are doing work and just have
the fun a 6 year old deserves nannan
```

In [228]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
t'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves'
, 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselve
s', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
s', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'th
is', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'h
ave', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between',
'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'h
ow', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 's
o', 'than', 'too', 'very', \
```

```
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should',
"should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",
'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is
n't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
 "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [229]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████
██████| 109248/109248 [01:34<00:00, 1156.32it/s]
```

In [230]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[230]:
```
'my kindergarten students varied disabilities ranging speech language d
elays cognitive delays gross fine motor delays autism they eager beaver
s always strive work hardest working past limitations the materials one
s i seek students i teach title i school students receive free reduced
price lunch despite disabilities limitations students love coming schoo
l come eager learn explore have ever felt like ants pants needed groove
move meeting this kids feel time the want able move learn say wobble ch
airs answer i love develop core enhances gross motor turn fine motor sk
ills they also want learn games kids not want sit worksheets they want
```

```
learn count jumping playing physical engagement key success the number
toss color shape mats make happen my students forget work fun 6 year ol
d deserves nannan'
```

1.3.2 Project title Text

In [ ]:
```
# similarly you can preprocess the titles also
```

In [231]:
```
project_data.head(2)
```

Out[231]:

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

In [233]:
```
print (project_data['project_title'].values[10])
print ("*"*50)
print (project_data['project_title'].values[100])
print ("*"*50)
```

```
print (project_data['project_title'].values[5000])
print ("*"*50)
print (project_data['project_title'].values[9000])
```

```
Reading Changes Lives
**************************************************
21st Century learners, 21st century technology!
**************************************************
Bouncing Our Wiggles and Worries Away!
**************************************************
Team Teaching with organization: Making room for individuals!!
```

In [379]:
```
#removing special chars, stop words etc
# Combining all the above statemennts
from tqdm import tqdm
processed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sen = decontracted(sentance)
    sen = sen.replace('\\r', ' ')
    sen = sen.replace('\\"', ' ')
    sen = sen.replace('\\n', ' ')
    sen = sen.replace('\"', ' ')
    sen = sen.replace('\\', ' ')
    sen = sen.replace('!!', ' ')
    sen = re.sub('[^A-Za-z0-9]+', ' ', sen)
    # https://gist.github.com/sebleier/554280
    sen = ' '.join(e for e in sen.split() if e not in stopwords)
    processed_title.append(sen.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████| 109248/109248 [00:05<00:00, 19505.42it/s]
```

In [380]:
```
#After Pre Processing on Titles; check some titles
print (processed_title[6000])
print ("*"*50)
print (processed_title[7892])
```

```
tech must haves spectacular student center
```

```
**************************************************
college signing day
```

1. 4 Preparing data for models

In [242]:
```
project_data.columns
```

Out[242]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_stat
e',
       'project_submitted_datetime', 'project_grade_category', '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_appr
oved',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      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/

In [267]:
```
# 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 ("*"*50)
print(vectorizer.get_feature_names())
print ("*"*50)
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 [265]:
```
# 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())

print ("*"*50)

sub_categories_one_hot = vectorizer.transform(project_data['clean_subca
tegories'].values)

print("Shape of matrix after one hot encodig ",sub_categories_one_hot.s
hape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolveme
nt', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutri
tionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingA
rts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPre
p', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopme
nt', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Healt
h_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing',
'Mathematics', 'Literacy']
**************************************************
Shape of matrix after one hot encodig  (109248, 30)
```

In [ ]:
```
# Please do the similar feature encoding with state, teacher_prefix and
 project_grade_category also
```

In [346]:
```
# we use count vectorizer to convert the values into one hot encoded fe
atures
# school_state
from sklearn.feature_extraction.text import CountVectorizer
print(project_data['school_state'].values)

print ("*"*100)
vectorizer_state = CountVectorizer(lowercase=False, binary=True)

vectorizer_state.fit(project_data['school_state'].values)

print(vectorizer_state.get_feature_names())

print ("*"*100)
```

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

print(state_one_hot.toarray()[0:5])

print ("*"*100)

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

```
['IN' 'FL' 'AZ' ..., 'NJ' 'NY' 'VA']
****************************************************************************
****************************
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'H
I', 'IA', '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']
****************************************************************************
****************************
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
  0 0 0 0 0 0 1 0 0 0 0 0 0 0]]
****************************************************************************
****************************
Shape of state_one_hot matrix after one hot encodig  (109248, 51)
```

In [361]:
```
# Vectorize project_grade_category
#Pre process project_grade_category: Remove the space and also "-"
#http://www.datasciencemadesimple.com/strip-space-column-pandas-dataframe-leading-trailing-2/
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(" ","")

project_data['project_grade_category'] = project_data['project_grade_category'].str.replace("-","to")

#https://www.geeksforgeeks.org/get-unique-values-from-a-column-in-pandas-dataframe/
print(project_data['project_grade_category'].unique())

print ("*"*100)
vectorizer_grade = CountVectorizer(lowercase=False, binary=True)

vectorizer_grade.fit(project_data['project_grade_category'].values)

print(vectorizer_grade.get_feature_names())

print ("*"*100)

grade_one_hot = vectorizer_grade.transform(project_data['project_grade_category'].values)

print(grade_one_hot.toarray()[0:5])

print ("*"*100)

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

```
['GradesPreKto2' 'Grades6to8' 'Grades3to5' 'Grades9to12']
****************************************************************************
****************************
['Grades3to5', 'Grades6to8', 'Grades9to12', 'GradesPreKto2']
****************************************************************************
****************************
[[0 0 0 1]
```

```
 [0 1 0 0]
 [0 1 0 0]
 [0 0 0 1]
 [0 0 0 1]]
**************************************************************************
****************************
Shape of grade_one_hot matrix after one hot encodig  (109248, 4)
```

In [252]:
```
#https://datascience.stackexchange.com/questions/12645/how-to-count-the
-number-of-missing-values-in-each-row-in-pandas-dataframe
project_data.isnull().sum()
```

Out[252]:
```
Unnamed: 0                                         0
id                                                 0
teacher_id                                         0
teacher_prefix                                     3
school_state                                       0
project_submitted_datetime                         0
project_grade_category                             0
project_title                                      0
project_essay_1                                    0
project_essay_2                                    0
project_essay_3                               105490
project_essay_4                               105490
project_resource_summary                           0
teacher_number_of_previously_posted_projects       0
project_is_approved                                0
clean_categories                                   0
clean_subcategories                                0
essay                                              0
price                                              0
quantity                                           0
dtype: int64
```

In [322]:
```
# Replace NaN values in teacher_refix with NotMentioned
project_data["teacher_prefix"]=project_data["teacher_prefix"].replace(t
o_replace ="NaN", value ="NotMentioned")
```

In [330]:
```
#check if nan vaules still exist for teacher_prefix
```

```
project_data["teacher_prefix"].isnull().sum()
```

Out[330]:
```
0
```

In [329]:
```
# we use count vectorizer to convert the values into one hot encoded fe
atures
# Vectorize teacher_prefix
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_tp = CountVectorizer(lowercase=False, binary=True)

print(project_data['teacher_prefix'].values)

print ("*"*100)

vectorizer_tp.fit(project_data['teacher_prefix'].values)

print(vectorizer_tp.get_feature_names())

print ("*"*100)

teacher_prefix_one_hot = vectorizer_tp.transform(project_data['teacher_
prefix'].values)

print(categories_one_hot.toarray()[0:1])

print ("*"*100)
print("Shape of teacher_prefix_one_hot matrix after one hot encodig ",t
eacher_prefix_one_hot.shape)
```

```
['Mrs.' 'Mr.' 'Ms.' ..., 'Mrs.' 'Mrs.' 'Ms.']
****************************************************************************
****************************
['Dr', 'Mr', 'Mrs', 'Ms', 'NotMentioned', 'Teacher']
****************************************************************************
****************************
[[0 0 0 0 0 0 0 1]]
****************************************************************************
****************************
Shape of teacher_prefix_one_hot matrix after one hot encodig  (109248,
```

6)

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [339]:
```
# We are considering only the words which appeared in at least 10 docum
ents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow_essay = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow_essay.shape)
print('*'*100)
print('Few random words from the essays')
print('*'*100)
print(vectorizer.get_feature_names()[1111:1120])
```

```
Shape of matrix after one hot encodig  (109248, 16623)
****************************************************************************
****************************
Few random words from the essays
****************************************************************************
****************************
['armstrong', 'army', 'aromatherapy', 'arose', 'around', 'arrange', 'ar
ranged', 'arrangement', 'arrangements']
```

1.4.2.2 Bag of Words on `project_title`

In [ ]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

In [382]:
```
vectorizer = CountVectorizer(min_df=10)
text_bow_title = vectorizer.fit_transform(processed_title)
print("Shape of matrix after one hot encodig ",text_bow_title.shape)
print("*"*100)
print("First 2 rows of sparse matrix ",text_bow_title[:2])
```

```
print("*"*100)
print("First 2 rows of dense matrix \n",text_bow_title.todense()[:2,169
9:])
print('*'*100)
print('Few random words from the titles')
print(vectorizer.get_feature_names()[1111:1120])
print("*"*100)
print("Type of text_bow ",type(text_bow_title))
```

```
Shape of matrix after one hot encodig  (109248, 3329)
****************************************************************************
****************************
First 2 rows of sparse matrix    (0, 1426)      1
  (0, 1700)     1
  (0, 977)      1
  (0, 2879)     1
  (0, 918)      1
  (1, 1452)     1
  (1, 2333)     1
  (1, 3178)     1
  (1, 1700)     1
****************************************************************************
****************************
First 2 rows of dense matrix
 [[0 1 0 ..., 0 0 0]
 [0 1 0 ..., 0 0 0]]
****************************************************************************
****************************
Few random words from the titles
['fidgets', 'fidgety', 'field', 'fifth', 'fight', 'fighting', 'filamen
t', 'file', 'fill']
****************************************************************************
****************************
Type of text_bow  <class 'scipy.sparse.csr.csr_matrix'>
```

1.4.2.3 TFIDF vectorizer

In [283]:
```
#For Pre Processed Essay
```

```python
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 [ ]:

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

In [383]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_title = TfidfVectorizer(min_df=10)
text_tfidf_title = vectorizer_title.fit_transform(processed_title)
print("Shape of matrix after one hot encodig ",text_tfidf_title.shape)
print("*"*100)
print("First 2 rows of sparse matrix ",text_tfidf_title[:2])
print("*"*100)
print("Type of text_bow ",type(text_tfidf_title))
print("*"*100)
print('Few random feature names of tfidf words in project title')
print('='*100)
print(vectorizer_title.get_feature_names()[1111:1120])
print("*"*100)
print(text_tfidf_title.toarray()[0:2,1700:1705])
```

```
Shape of matrix after one hot encodig  (109248, 3329)
************************************************************************
*****************************
First 2 rows of sparse matrix    (0, 918)       0.5360933294
  (0, 2879)     0.445820557858
  (0, 977)      0.460574924461
  (0, 1700)     0.341317060614
  (0, 1426)     0.430373530152
  (1, 1700)     0.35288507621
  (1, 3178)     0.516002690242
  (1, 2333)     0.565363534765
  (1, 1452)     0.538123982191
```

```
**************************************************************************
****************************
Type of text_bow  <class 'scipy.sparse.csr.csr_matrix'>
**************************************************************************
****************************
Few random feature names of tfidf words in project title
==========================================================================
==============================
['fidgets', 'fidgety', 'field', 'fifth', 'fight', 'fighting', 'filamen
t', 'file', 'fill']
**************************************************************************
****************************
[[ 0.34131706  0.          0.          0.          0.        ]
 [ 0.35288508  0.          0.          0.          0.        ]]
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [285]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

Loading Glove Model
1917495it [06:32, 4879.69it/s]
```

```
Done. 1917495  words loaded!

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

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

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

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

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


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

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


'''
```

Out[285]:
```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230
349/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove
```

```
Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}
\n    for line in tqdm(f):\n        splitLine = line.split()\n        w
ord = splitLine[0]\n        embedding = np.array([float(val) for val in
splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",le
n(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 preproce
d_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in th
e coupus", len(words))\nwords = set(words)\nprint("the unique words in
the coupus", len(words))\n\ninter_words = set(model.keys()).intersectio
n(words)\nprint("The number of words that are present in both glove vec
tors and our coupus",       len(inter_words),"(",np.round(len(inter_wor
ds)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(mo
del.keys())\nfor i in words:\n    if i in words_glove:\n        words_c
ourpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n
\n\n# stronging variables into pickle files python: http://www.jessicay
ung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimpo
rt pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump
(words_courpus, f)\n\n\n'
```

In [290]:
```
# 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 [291]:
```
# 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/re
view
    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 [01:07<00:00, 1610.90it/s]
```

```
109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [ ]:
```
# Similarly you can vectorize for title also
```

In [384]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_title = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm(processed_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/re
view
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title.append(vector)

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

```
100%|
```

```
100%|| 109248/109248 [00:02<00:00, 42154.88it/s]
```

```
109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [294]:
```
# 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 v
alue
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model
.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [295]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is store
d 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 sentenc
e/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 t
he tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentenc
e.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 [06:24<00:00, 284.32it/s]
```

```
109248
300
```

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

In [ ]:
```
# Similarly you can vectorize for title also
```

In [385]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(processed_title)
# we are converting a dictionary with word as a key, and the idf as a v
alue
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model
.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [387]:
```
# average Word2Vec
# compute average word2vec for each Project Title
tfidf_weighted_w2v_vectors_title = []; # the avg-w2v for each sentence/
review is stored in this
for sentence in tqdm(processed_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentenc
e/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 t
he tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentenc
e.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_weighted_w2v_vectors_title.append(vector)
print(len(tfidf_weighted_w2v_vectors_title))
print(len(tfidf_weighted_w2v_vectors_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████| 109248/109248 [00:06<00:00, 17632.13it/s]
```

```
109248
300
```

1.4.3 Vectorizing Numerical features

In [367]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
print("*"*100)
print(price_standardized.shape)
```

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

```
**************************************************************************
****************************
(109248, 1)
```

In [368]:
```
price_standardized
```

Out[368]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

In [370]:
```
# Reshape your data either using array.reshape(-1, 1)

quantity_scalar = StandardScaler()
quantity_scalar.fit(project_data['quantity'].values.reshape(-1,1)) # fi
nding the mean and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(p
rice_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
quantity_standardized = quantity_scalar.transform(project_data['quantit
y'].values.reshape(-1, 1))
print("*"*100)
print(quantity_standardized.shape)
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
**************************************************************************
****************************
(109248, 1)
```

In [372]:
```
# Reshape your data either using array.reshape(-1, 1)

teacher_number_of_previously_posted_projects_scalar = StandardScaler()
teacher_number_of_previously_posted_projects_scalar.fit(project_data['t
eacher_number_of_previously_posted_projects'].values.reshape(-1,1)) # f
inding the mean and standard deviation of this data
```

```
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(p
rice_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized = teacher_num
ber_of_previously_posted_projects_scalar.transform(project_data['teache
r_number_of_previously_posted_projects'].values.reshape(-1, 1))
print("*"*100)
print(teacher_number_of_previously_posted_projects_standardized.shape)
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
****************************************************************************
****************************
(109248, 1)
```

1.4.4 Merging all the above features

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

In [388]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow_title.shape)
print(price_standardized.shape)
print(teacher_prefix_one_hot.shape)
print(grade_one_hot.shape)
print(state_one_hot.shape)
print(quantity_standardized.shape)
print(teacher_number_of_previously_posted_projects_standardized.shape)
print("Title-BOW :",(text_bow_title.shape))
print("*"*100)
print("Title-TFIDF :",(text_tfidf_title.shape))
print("*"*100)
print("Title-AVG W2V :{},{}".format(len(avg_w2v_vectors_title), len(avg
_w2v_vectors_title[0])))
print("*"*100)
print("Title-TFIDF W2V : {},{}".format(len(tfidf_weighted_w2v_vectors_t
itle), len(tfidf_weighted_w2v_vectors_title[0])))
```

```
(109248, 9)
(109248, 30)
(109248, 3329)
(109248, 1)
(109248, 6)
(109248, 4)
(109248, 51)
(109248, 1)
(109248, 1)
Title-BOW : (109248, 3329)
**********************************************************************
****************************
Title-TFIDF : (109248, 3329)
**********************************************************************
****************************
Title-AVG W2V :109248,300
**********************************************************************
****************************
Title-TFIDF W2V : 109248,300
```

In [411]:

```
# 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_bow = hstack((categories_one_hot, sub_categories_one_hot, text_bow_title, price_standardized,teacher_prefix_one_hot,grade_one_hot,state_one_hot,quantity_standardized,teacher_number_of_previously_posted_projects_standardized))
X_bow.shape
print(type(X_bow))

X_bow = X_bow.toarray()
```

```
<class 'scipy.sparse.coo.coo_matrix'>
```

In [412]:

```
# 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 an
d a dense matirx :)
X_tfidf = hstack((categories_one_hot, sub_categories_one_hot, text_tfid
f_title, price_standardized,teacher_prefix_one_hot,grade_one_hot,state_
one_hot,quantity_standardized,teacher_number_of_previously_posted_proje
cts_standardized))
X_tfidf.shape
X_tfidf=X_tfidf.toarray()
```

In [418]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/40840
39
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix an
d a dense matirx :)
X_avg_w2v = hstack((categories_one_hot, sub_categories_one_hot, avg_w2v
_vectors_title, price_standardized,teacher_prefix_one_hot,grade_one_hot
,state_one_hot,quantity_standardized,teacher_number_of_previously_poste
d_projects_standardized))
X_avg_w2v.shape
X_avg_w2v=X_avg_w2v.toarray()
```

In [419]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/40840
39
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix an
d a dense matirx :)
X_weighted_w2v = hstack((categories_one_hot, sub_categories_one_hot, tf
idf_weighted_w2v_vectors_title, price_standardized,teacher_prefix_one_h
ot,grade_one_hot,state_one_hot,quantity_standardized,teacher_number_of_
previously_posted_projects_standardized))
X_weighted_w2v.shape
X_weighted_w2v=X_weighted_w2v.toarray()
```

In [428]:
```
# ALL FEATURES
# merge two sparse matrices: https://stackoverflow.com/a/19710648/40840
39
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix an
```

```
d a dense matirx :)
X_All_Feat = hstack((categories_one_hot, sub_categories_one_hot, text_t
fidf_title, price_standardized,teacher_prefix_one_hot,grade_one_hot,sta
te_one_hot,quantity_standardized,teacher_number_of_previously_posted_pr
ojects_standardized))
X_All_Feat.shape
X_All_Feat=X_All_Feat.toarray()
```

Assignment 2: Apply TSNE

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

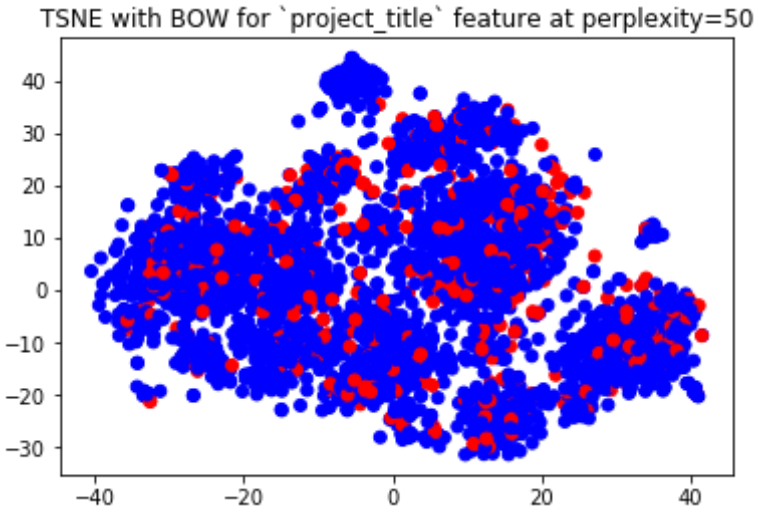
1. In the above cells we have plotted and analyzed many features. Please observe the plots and write the observations in markdown cells below every plot.
2. EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects
3. Build the data matrix using these features
  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

7. Note 2: Consider only 5k to 6k data points to avoid memory issues. If you run into memory error issues, reduce the number of data points but clearly state the number of datat-poins you are using

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

In [422]:
```
# please write all of the code with proper documentation and proper tit
les 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
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = X_bow[0:3000]
y = project_data['project_is_approved'][0:3000]

tsne = TSNE(n_components=2, perplexity=50, 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 int
o dense matrix

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

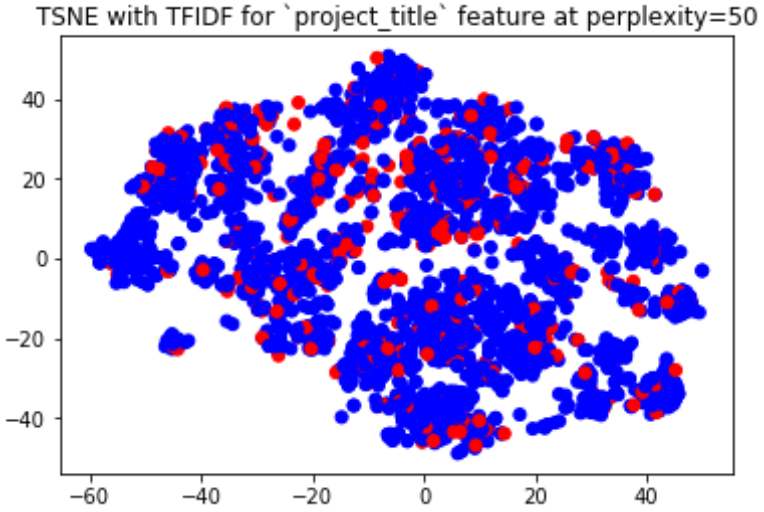
```
plt.title("TSNE with BOW for `project_title` feature at perplexity=50")
plt.show()
```



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

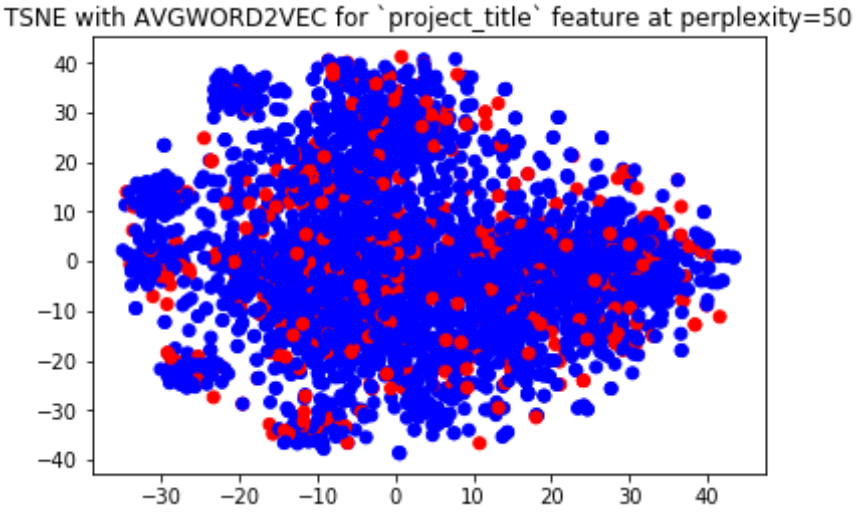
In [423]:
```
# please write all of the code with proper documentation and proper tit
les for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to
 the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = X_tfidf[0:3000]
```

```
y = project_data['project_is_approved'][0:3000]

tsne = TSNE(n_components=2, perplexity=50, 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 int
o dense matrix

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimen
sion_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=f
or_tsne_df['Score'].apply(lambda x: colors[x]))
plt.title("TSNE with TFIDF for `project_title` feature at perplexity=5
0")
plt.show()
```



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

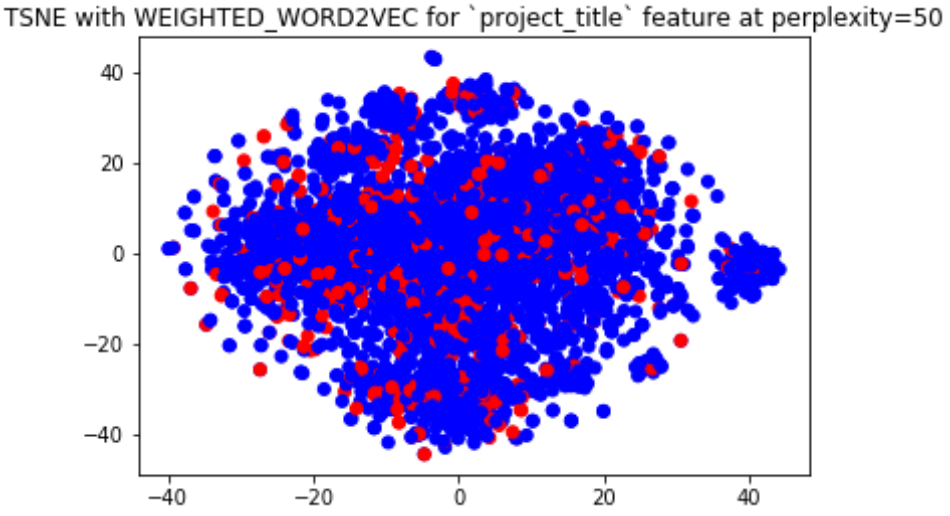
In [424]:
```
# please write all of the code with proper documentation and proper tit
les for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to
 the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = X_avg_w2v[0:3000]
y = project_data['project_is_approved'][0:3000]

tsne = TSNE(n_components=2, perplexity=50, 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 int
o dense matrix

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimen
sion_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=f
or_tsne_df['Score'].apply(lambda x: colors[x]))
plt.title("TSNE with AVGWORD2VEC for `project_title` feature at perplex
ity=50")
plt.show()
```



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

In [425]:
```
# please write all of the code with proper documentation and proper tit
les 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
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
```

```
x = X_weighted_w2v[0:3000]
y = project_data['project_is_approved'][0:3000]

tsne = TSNE(n_components=2, perplexity=50, 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 int
o dense matrix

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimen
sion_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=f
or_tsne_df['Score'].apply(lambda x: colors[x]))
plt.title("TSNE with WEIGHTED_WORD2VEC for `project_title` feature at p
erplexity=50")
plt.show()
```



TSNE with BOW , TFIDF , AVG W2V , TFIDF Weighted

W2V

2.5 Summary

TSNE with : Bag of Words, TFIDF, Avg Word2Vec, TFIDF Weighted Word2Vec does not produce any clusters of similar data points. Hence, it is not possible to conclude anything