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 project. **Example:** `p036502`
`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 following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following 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, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy`

Feature	Description
	• Literature & Writing, Social Sciences
project_resource_summary	An explanation of the resources needed for the project. **Example:** • My students need hands on 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 was submitted. **Example:** 2016-04-28 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project. **Example:** bdf8baa8fedef6bfeec7ae4ff1c15c56
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 previously submitted by the same teacher. **Example:** 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."

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 [85]:

```
%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 import metrics
from sklearn.metrics import confusion_matrix
from nltk.stem.porter import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords
import re
import gensim
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
import plotly.offline as offline
import plotly.graph_objs as go
plotly.offline.init_notebook_mode()
from collections import Counter
sns.set(style="whitegrid")
```

Reading Data

In [5]:

```
project_data = pd.read_csv('C:/Users/User/Downloads/train_data.csv')
resource_data = pd.read_csv('C:/Users/User/Downloads/resources.csv')
```

In [9]:

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

```
Number of data points in train data (109248, 17)
**************************************************
The attributes of data: ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' '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']
```

In [10]:

```
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[10]:

	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 [11]:

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

```
Number of projects that are approved for funding  92706 , ( 84.85830404217927 %)
Number of projects that are not approved for funding  16542 , ( 15.141695957820739 %)
```

Observation:

1. 85.85% of projects approved for funding .
2. 15.14% of projects not approved for funding.

1.2.1 Univariate Analysis: School State

In [12]:

```
# 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()
temp.columns = ['state_code', 'num_proposals']
```

In [13]:

```
# 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 [18]:

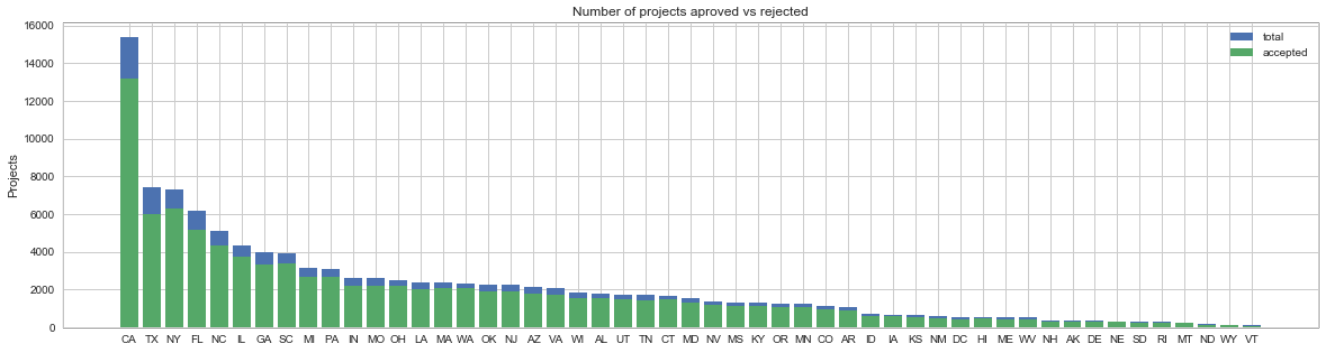
```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

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

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

In [19]:

```
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 [20]:

```
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
```

Observation:

1) 85.81% of project approved from the projects given by "CA school_state" .The projects given by "CA school_state" are more than other school_states.

2) 80.00% of project approved from the projects given by "VT school_ state".The projects given by "vt school_state" are less than other school_states.

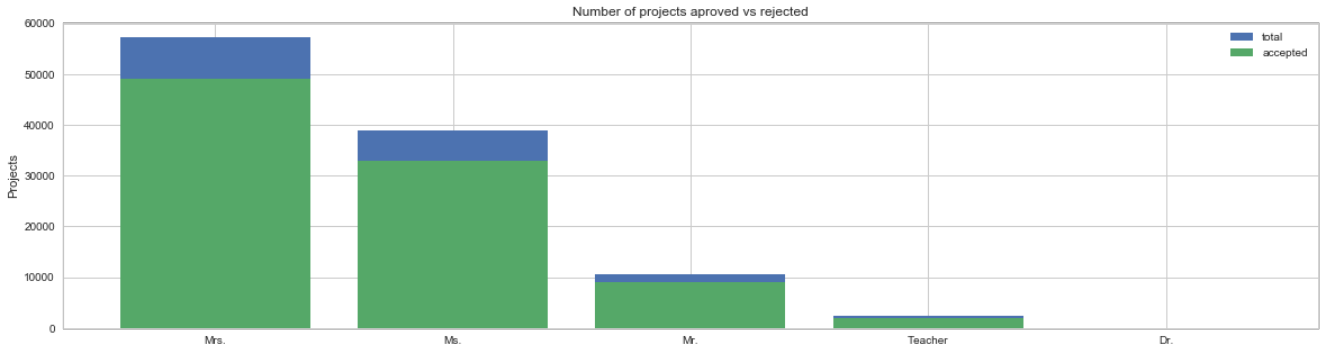
3) The project approved mostly depends upon the number of projects assigned.

4) Every state has greater than 80% success rate in approval.

1.2.2 Univariate Analysis: teacher_prefix

In [21]:

```
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
```

Obervation:

1) 85.55% of project approved from the projects given by {'Mrs'} .The projects given by {'Mrs'} are more than other teachers.

2) 69.23% of project approved from the projects given by {'Dr'}.The projects given by{'Dr'} are less than other teachers.

3) The project approved mostly depends upon the number of projects assigned.

1.2.3 Univariate Analysis: project_grade_category

In [22]:

```
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
```

Observation:

1) 84.87% of projects approved from Grades PreK-2.The number of projects are more compare to other Grades.

2) 83.76% of projects approved from Grades 9-12.The number of projects are less compare to other Grades.

3) The project approved mostly depends upon the number of projects assigned.

1.2.4 Univariate Analysis: project_subject_categories

In [23]:

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

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

In [24]:

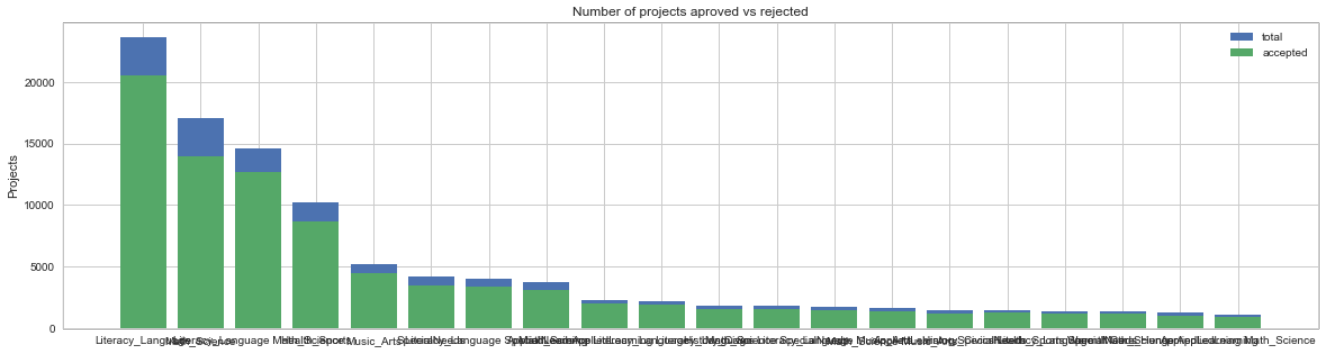
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[24]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [25]:

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




```
                 clean_categories  project_is_approved  total       Avg
24                Literacy_Language                20520  23655  0.867470
32                     Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12725  14636  0.869432
8                     Health_Sports                 8640  10177  0.848973
40                       Music_Arts                 4429   5180  0.855019
==================================================
                    clean_categories  project_is_approved  total       Avg
19  History_Civics Literacy_Language                 1271   1421  0.894441
14         Health_Sports SpecialNeeds                 1215   1391  0.873472
50                 Warmth Care_Hunger                 1212   1309  0.925898
33       Math_Science AppliedLearning                 1019   1220  0.835246
4        AppliedLearning Math_Science                  855   1052  0.812738
```

Observation:

1) 86.74% of projects approved from the projects of Literacy_language categories.The projects given from Literacy_language subject are more compare to other categories.

2) 86.94% of projects approved from the projects on combination both Literacy_language & math_science categories .

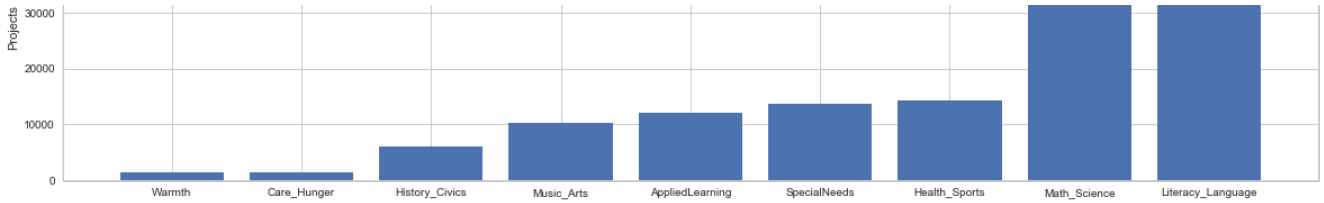
3) 89.44% of projects approved from the projects on combination both History_Civics & Literacy_language categories .

In [26]:

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

In [27]:

```
# 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 [28]:

```
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
```

Observation:

1) The highest number from Literacy_Language categories .

2) The least number from warmth categories.

1.2.5 Univariate Analysis: project_subject_subcategories

In [29]:

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

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

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

In [30]:

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

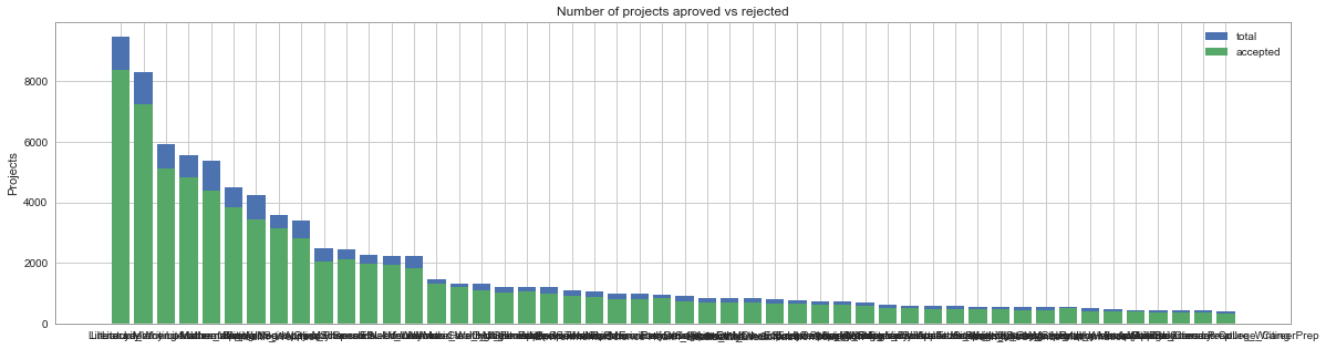
Out[30]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [31]:

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




```
                clean_subcategories  project_is_approved  total       Avg
317                         Literacy                 8371   9486  0.882458
319              Literacy Mathematics                7260   8325  0.872072
331  Literature_Writing Mathematics                  5140   5923  0.867803
318      Literacy Literature_Writing                 4823   5571  0.865733
342                      Mathematics                 4385   5379  0.815207
==================================================
                    clean_subcategories  project_is_approved  total       Avg
196       EnvironmentalScience Literacy                  389    444  0.876126
127                                 ESL                 349    421  0.828979
79                   College_CareerPrep                 343    421  0.814727
17   AppliedSciences Literature_Writing                 361    420  0.859524
3    AppliedSciences College_CareerPrep                 330    405  0.814815
```

Observation:

1) 88.24% of projects approved from the projects of Literacy subcategories.The projects given from Literacy_language are more compare to other subcategories.

2) 87.61% of projects approved from the projects on combination of EnivronmentalScience Literacy subcategories .

3) 81.52% of projects approved from the projects on Mathematics subcategories .The projects given from Mathematics are more compare to other subcategories.

4) 81.48% of projects approved from the projects on combination of AppliedSciences College_CareerPrep subcategories .

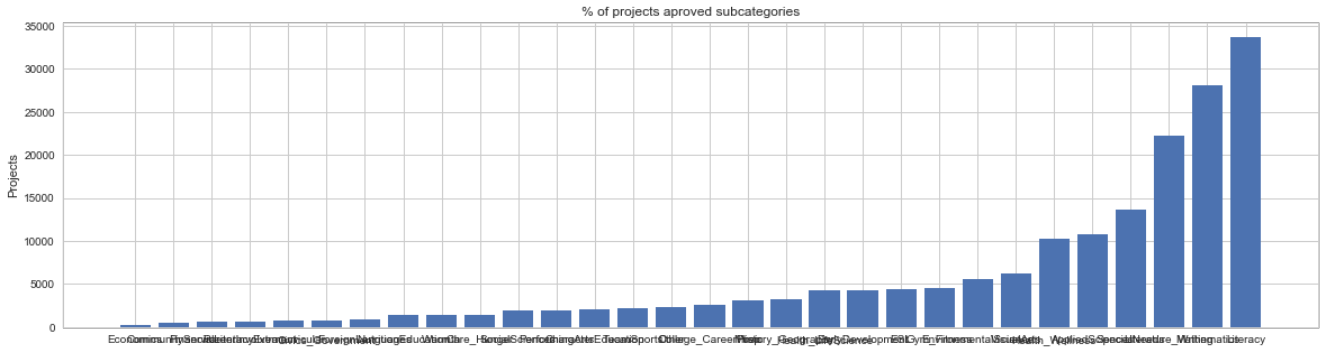
In [32]:

```
# 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 [33]:

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



In [34]:

```
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
```

Observation:

1) From the plot its not clear to identify the subcategories.The sorted_subcategories dic will give the list of subcategories.

2) The highest number from Literacy_Language subcategories .

3) The least number from Economics subcategories.

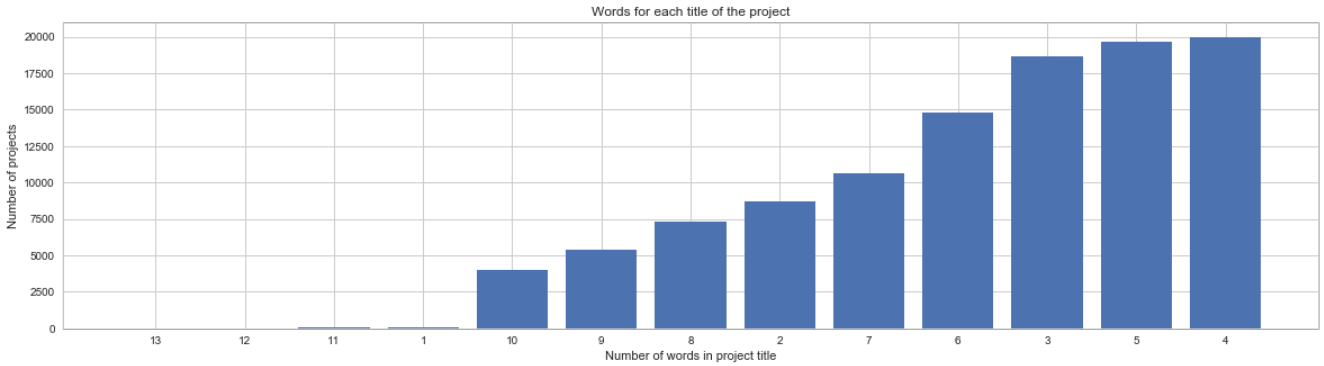
1.2.6 Univariate Analysis: Text features (Title)

Observation:

The number of words in project title are less as 4 in the number of projects.As the words increased as 10 in the number of words in project title there are less number of project .

In [35]:

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


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

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

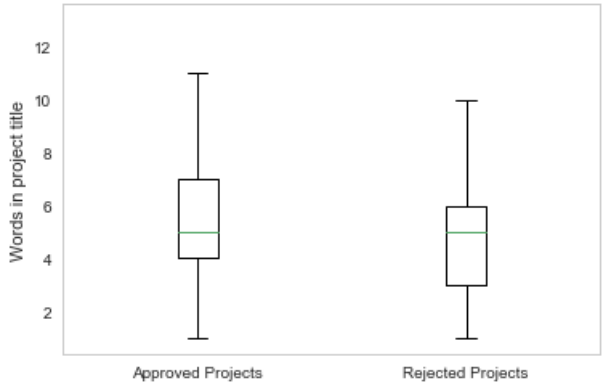


In [36]:

```
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
sns.set(style="whitegrid")
```

In [37]:

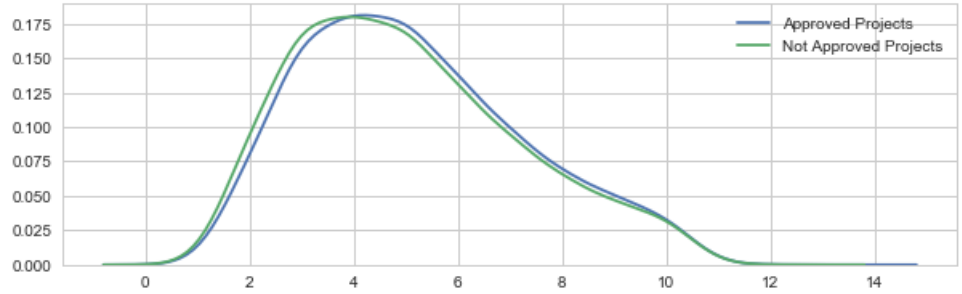
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



In [38]:

```
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()
```



Observation:

The number of words in project title are slightly more in approved projects than not approved projects.

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

In [39]:

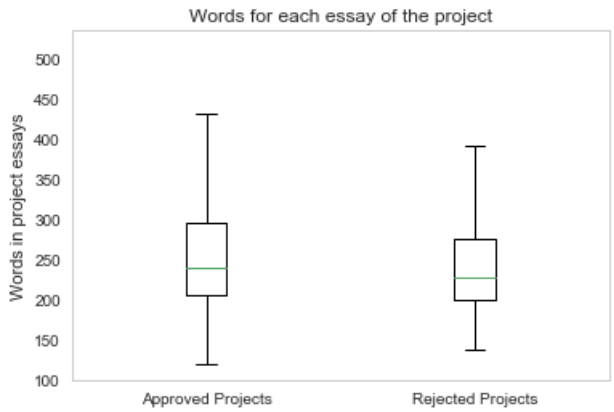
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [40]:

```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.split().app
ly(len)
approved_word_count = approved_word_count.values

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

In [41]:

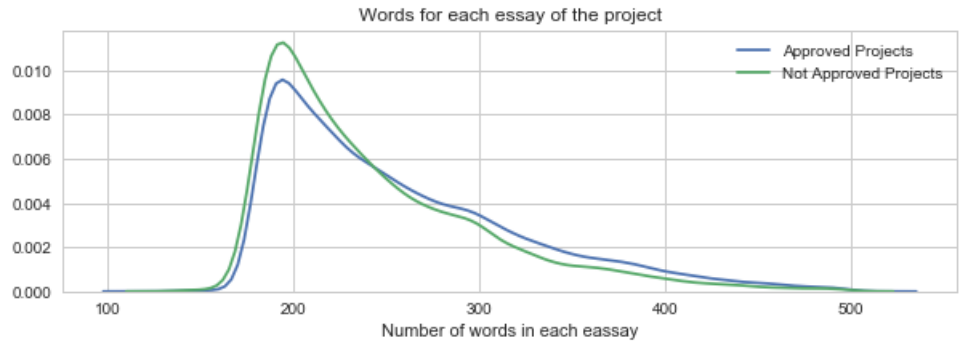
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()
```



In [42]:

```
plt.figure(figsize=(10,3))
```

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



Observation:

The number of words for each essay of project are slightly more in approved projects than not approved projects.

1.2.8 Univariate Analysis: Cost per project

In [44]:

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

Out[44]:

	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 [45]:

```
# 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[45]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [46]:

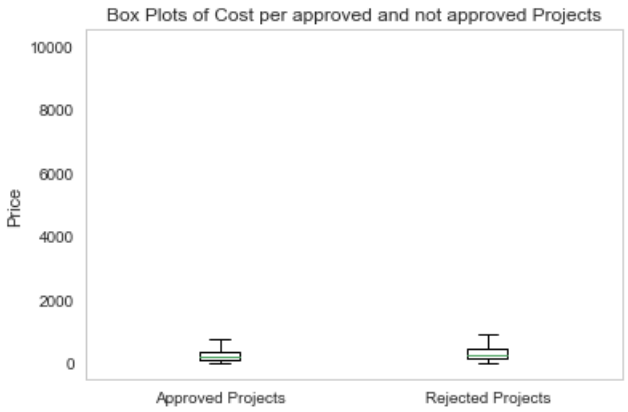
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [47]:

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

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

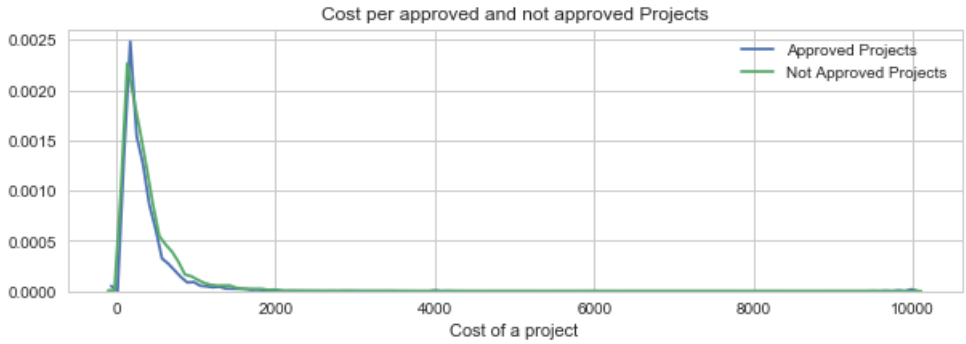
In [48]:

```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Price')
plt.grid()
plt.show()
```



In [49]:

```
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()
```



Observation:

From the box plot and Pdf its not clear about the cost per approved and not approved projects .

In [50]:

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

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

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

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

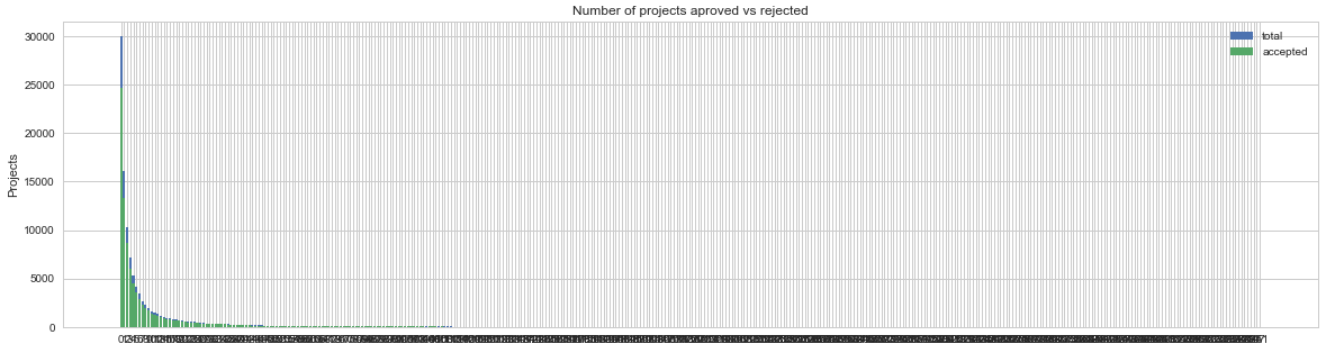
Observation:

From the prettytable its clearly explains about the approved projects and not approved projects at different percentiles.

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [51]:

```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects',
'project_is_approved', False)
```




```
   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  \
242                                           242                   1      1
268                                           270                   1      1
234                                           234                   1      1
335                                           347                   1      1
373                                           451                   1      1

     Avg
242  1.0
268  1.0
```

```
206   1.0
234   1.0
335   1.0
373   1.0
```

In [52]:

```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039
temp = pd.DataFrame(project_data.groupby("teacher_id")
['teacher_number_of_previously_posted_projects'].apply(np.mean)).reset_index()
temp.columns = ['teacher_id', 'num_proposals']
```

In [53]:

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

```
teacher with highest number of previously posted projects
                            teacher_id  num_proposals
44833  9f49ba20aa1c28eb95dbad8b8edd2b69     314.846154
44270  9d7051e2611cebdb758f1c7bd09360ac     326.760000
809    02bccf5c109ace4f3dcbce819a46daa1     347.166667
45044  a006826c170f91f85ff80dc5a132fade     351.666667
70484  fa2f220b537e8653fb48878ebb38044d     390.636364
==================================================
teacher with lowest number of previously posted projects
                            teacher_id  num_proposals
67122  ee29d8a44c30611083dd64dffc99dc8a            0.0
46144  a3b4cbd5f496fee9d6484c4f063b0974            0.0
20740  4a4d9bdc3cea355d25991f61fc72717d            0.0
46147  a3b8b68c44bad1004cf51a2e4bb41193            0.0
20738  4a4d14f3d9984e630caf1ad8c4d51bbf            0.0
```

In [54]:

```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='teacher_number_of_previously_posted_projects', 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 previously posted projects by teacher')
    plt.xticks(ind, list(data[xtick].values))
    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()
```

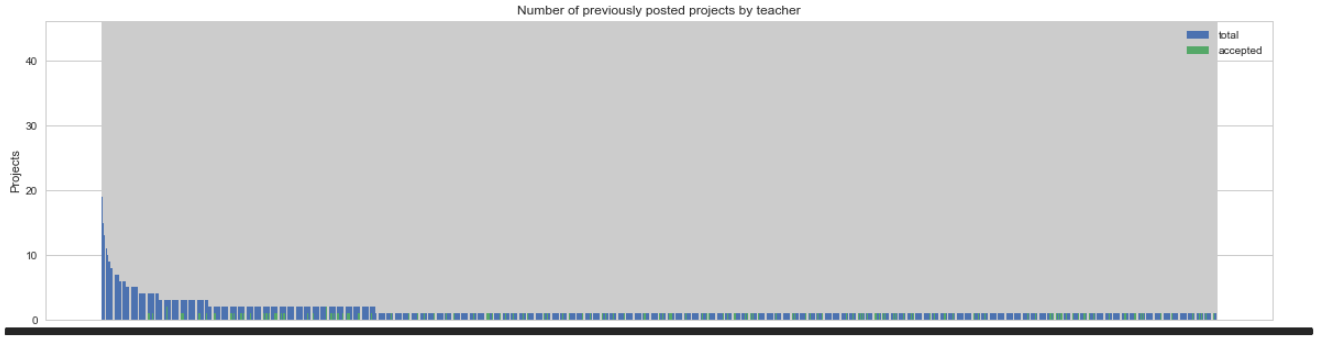
In [55]:

```
def univariate_barplots(data, col1, col2="teacher_number_of_previously_posted_projects", 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 [86]:

```
univariate_barplots(project_data, 'teacher_id', 'teacher_number_of_previously_posted_projects', False)
```




```
                            teacher_id  \
70484  fa2f220b537e8653fb48878ebb38044d
8702   1f64dcec848be8e95c4482cc845706b2
62925  df8a4b7ad173b57f7ac52e447cc24043
34570  7b17c95da53e3d1f011f84232ad01238
49149  ae67d8bbc64ec3bf7fd2db1297721160

       teacher_number_of_previously_posted_projects  total         Avg
70484                                             0     44  390.636364
8702                                              0     42  275.309524
62925                                             0     42   96.761905
34570                                             0     34   28.000000
49149                                             0     33  118.030303
==================================================
                            teacher_id  \
27489  61fdee5b0c34ea70671f52b3c3a01b71
27490  61ff263134490840c2a7f3b11e21cf0e
27491  61ff51943e34dba224a319b49a4d44fa
27492  61ff58caffe3bfef6e929db19e808aeb
72167  ffff8e040521f62207881376ecc964d5

       teacher_number_of_previously_posted_projects  total   Avg
27489                                             0      1   0.0
27490                                             0      1   0.0
27491                                             0      1   3.0
27492                                             0      1   0.0
72167                                             0      1  14.0
```

Observation:

we are getting the teacher number of previously posted projects with teacher id and the total & average.

"0" number of previously posted projects by most of the teachers.

1. 3 Preparing data for models

we are going to consider these features to build the data matrix.

```
      - school_state           : categorical data(one hot encoding)
      - clean_categories       : categorical data(one hot encoding)
      - clean_subcategories    : categorical data(one hot encoding)
      - project_grade_category : categorical data(one hot encoding)
      - teacher_prefix         : categorical data(one hot encoding)
```

```
- project_title          : text data(BOW,TFIDF,AVG W2V,TFIDF W2V)


- teacher_number_of_previously_posted_projects : numerical
- price                                        : numerical
```

1.3.1 Vectorizing Categorical data

school_state : categorical data(one hot encoding)

In [56]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer( lowercase=False, binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


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

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

clean_categories : categorical data(one hot encoding)

In [58]:

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


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

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

clean_subcategories : categorical data(one hot encoding)

In [59]:

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


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

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
```

```
Shape of matrix after one hot encoding  (109248, 30)
```

project_grade_category : categorical data(one hot encoding)

In [60]:

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


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

```
['12', 'Grades', 'PreK']
Shape of matrix after one hot encoding  (109248, 3)
```

teacher_prefix : categorical data(one hot encoding)

In [61]:

```
# we use count vectorizer to convert the values into one hot encoded features
from dask import dataframe as df
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer( lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values.astype('U'))


print(vectorizer.get_feature_names())

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

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher', 'nan']
Shape of matrix after one hot encoding  (109248, 6)
```

1.3.2 Vectorizing Text data

project_title : text data(BOW,TFIDF,AVG W2V,TFIDF W2V)

In [62]:

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

Out[62]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

	Unnamed:						

In [64]:

```
# printing some random title.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [65]:

```
# 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 [66]:

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

```
We Need To Move It While We Input It!
==================================================
```

In [67]:

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

```
We Need To Move It While We Input It!
```

In [68]:

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

```
print(sent)
```

```
We Need To Move It While We Input It
```

In [69]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', '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', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', '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', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [70]:

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

```
100%|██████████████████████████████████| 109248/109248 [00:10<00:00, 10713.13it/s]
```

In [71]:

```
# after preprocesing
preprocessed_titles[20000]
```

Out[71]:

```
'we need to move it while we input it'
```

1.3.2.1 Bag of words

In [72]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
```

```
project_title_bow = vectorizer.fit_transform(project_data["project_title"])
print("Shape of matrix after one hot encoding ",project_title_bow.shape)
```

```
Shape of matrix after one hot encoding  (109248, 3349)
```

1.3.2.2 TFIDF vectorizer

In [73]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
project_title_tfidf = vectorizer.fit_transform(project_data["project_title"])
print("Shape of matrix after one hot encoding ",project_title_tfidf.shape)
```

```
Shape of matrix after one hot encoding  (109248, 3349)
```

1.3.2.3 Using Pretrained Models: Avg W2V

In [74]:

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

```
Loading Glove Model
```

```
1917494it [14:48, 2156.95it/s]
```

```
Done. 1917494  words loaded!
```

In [75]:

```
words = []

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

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

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

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

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

```
all the words in the coupus 473570
```

all the words in the coupus 473570
the unique words in the coupus 16903
The number of words that are present in both glove vectors and our coupus 15917 (94.167 %)
word 2 vec length 15917

In [76]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-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 [77]:

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

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

```
100%|██████████████████████████████| 109248/109248 [00:04<00:00, 23609.63it/s]
```

```
109248
300
```

1.3.2.7 Using Pretrained Models: TFIDF weighted W2V

In [78]:

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

In [79]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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 [00:13<00:00, 8238.44it/s]
```

```
109248
300
```

1.3.3 Vectorizing numerical features

1.3.3.1 price:numerical

In [80]:

```
# 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))
```

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

In [81]:

```
price_standardized
```

Out[81]:

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

1.3.3.2 Teacher_number_of_previously_posted_projects : numerical

In [86]:

```
# 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

# previously_posted_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)

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

```
# Now standardize the data with above maen and variance.
previously_posted_standardized =
previously_posted_scalar.transform(project_data['teacher_number_of_previously_posted_projects'].va
lues.reshape(-1, 1))
```

```
Mean : 11.153165275336848, Standard deviation : 367.49634838483496
```

In [83]:

```
previously_posted_standardized
```

Out[83]:

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

1.3.4 Merging all the above features

- we need to merge all the numerical vectors i.e categorical, text_BOW, numerical vectors

In [73]:

```
print(school_state_one_hot.shape)
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(project_title_bow.shape)
print(price_standardized.shape)
print(previously_posted_standardized.shape)
```

```
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 3)
(109248, 6)
(109248, 3349)
(109248, 1)
(109248, 1)
```

In [74]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((school_state_one_hot,categories_one_hot,
sub_categories_one_hot,project_grade_category_one_hot,teacher_prefix_one_hot, project_title_bow, p
rice_standardized.shape,previously_posted_standardized))
X.shape
```

Out[74]:

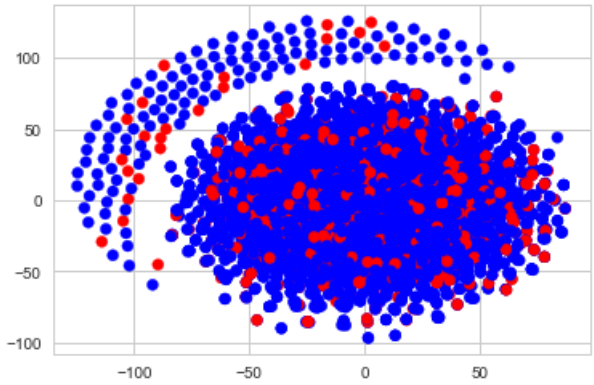
```
(109248, 3450)
```

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

In [74]:

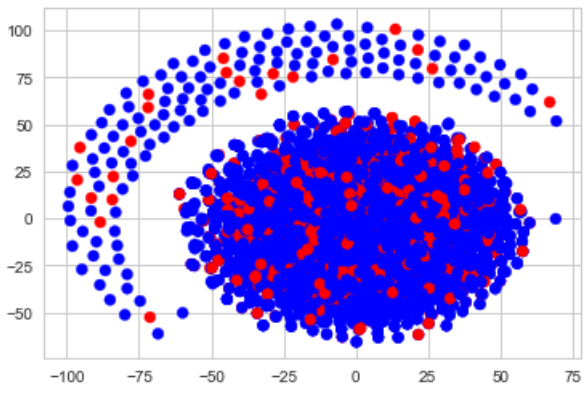
```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
```

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=30,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



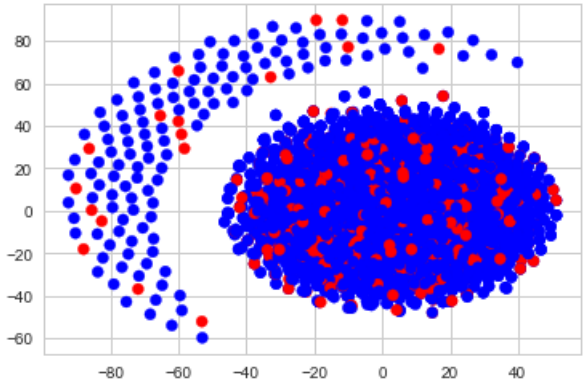
In [113]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=100,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



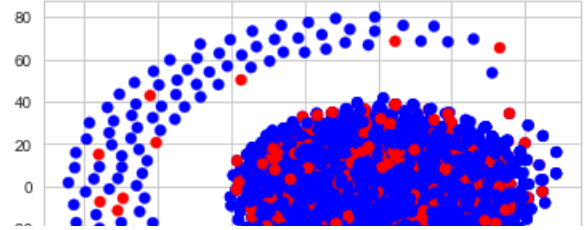
In [115]:

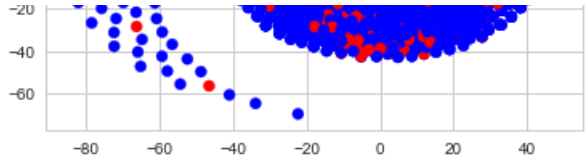
```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=175,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



In [114]:

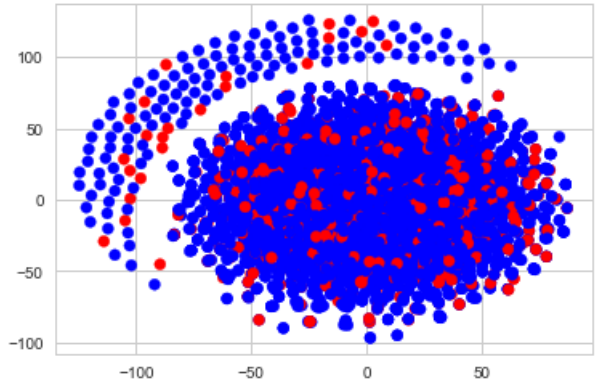
```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=250,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```





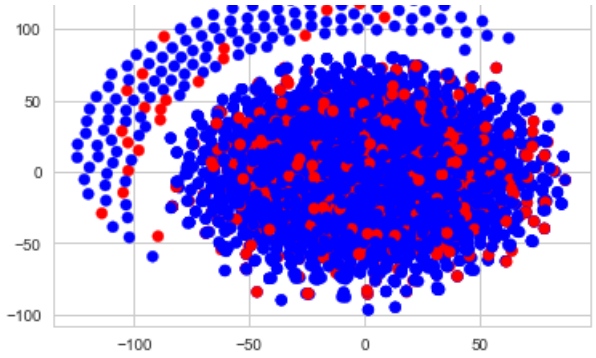
In [123]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



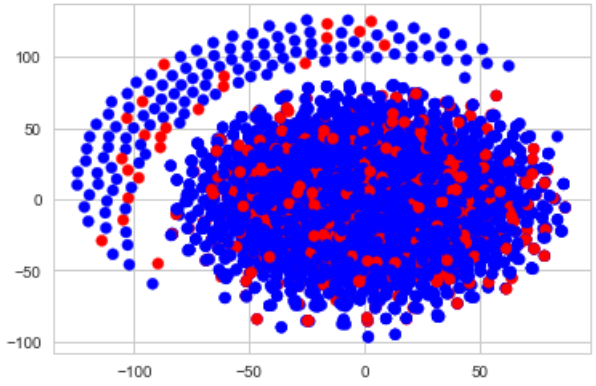
In [124]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tsne = TSNE(n_components=2, perplexity=200, learning_rate=200)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



In [125]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tsne = TSNE(n_components=2, perplexity=500, learning_rate=200)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



- we need to merge all the numerical vectors i.e categorical, project_title_tfidf, numerical vectors

In [75]:

```
print(school_state_one_hot.shape)
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(project_title_tfidf.shape)
print(previously_posted_standardized.shape)
```

```
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 3)
(109248, 6)
(109248, 3349)
(109248, 1)
```

(109248, 1)

In [76]:

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

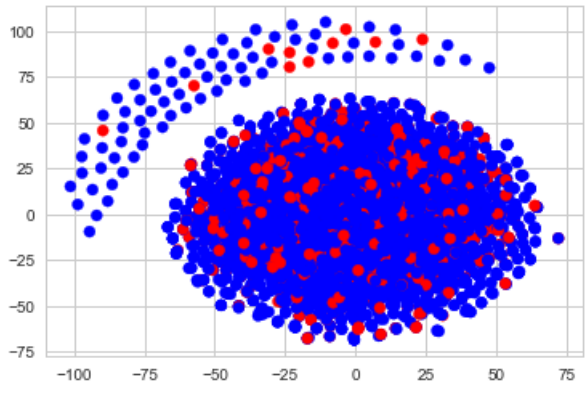
Out[76]:

(109248, 3449)

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

In [118]:

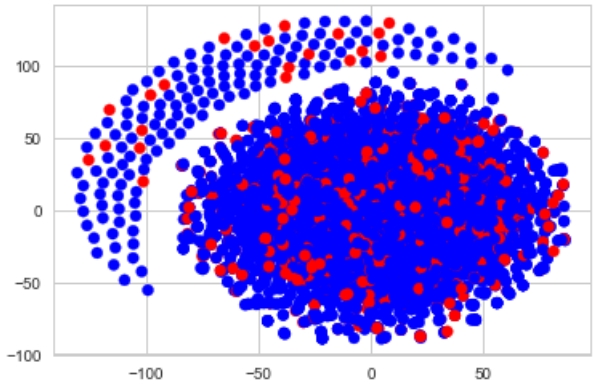
```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=100,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



In [77]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]
```

```
standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



- we need to merge all the numerical vectors i.e categorical, avg_w2v, numerical vectors

In [78]:

```
print(school_state_one_hot.shape)
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(avg_w2v_vectors)
print(previously_posted_standardized.shape)
```

```
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 3)
(109248, 6)
```

```
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`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
(109248, 1)
```

In [79]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((school_state_one_hot,categories_one_hot,
sub_categories_one_hot,project_grade_category_one_hot,teacher_prefix_one_hot,avg_w2v_vectors,
previously_posted_standardized))
X.shape
```

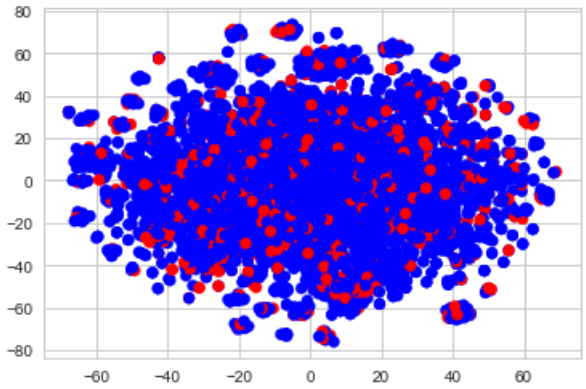
Out[79]:

```
(109248, 400)
```

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

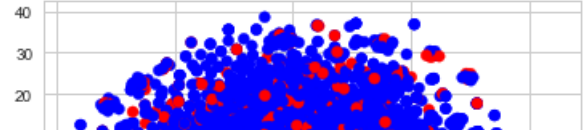
In [80]:

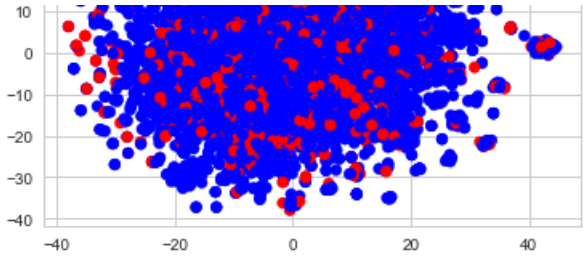
```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



In [81]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=100,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```





- we need to merge all the numerical vectors i.e categorical,TFIDF_w2v, numerical vectors

In [84]:

```
print(school_state_one_hot.shape)
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(tfidf_w2v_vectors)
print(previously_posted_standardized.shape)
```

```
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 3)
(109248, 6)
```

```
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`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
(109248, 1)
```

In [83]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((school_state_one_hot,categories_one_hot,
sub_categories_one_hot,project_grade_category_one_hot,teacher_prefix_one_hot,tfidf_w2v_vectors,
previously_posted_standardized))
X.shape
```

Out[83]:

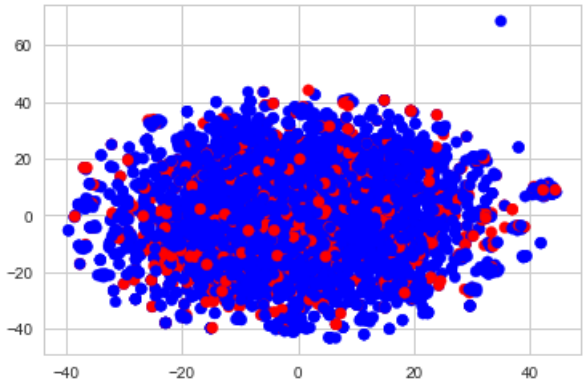
```
(109248, 400)
```

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

In [84]:

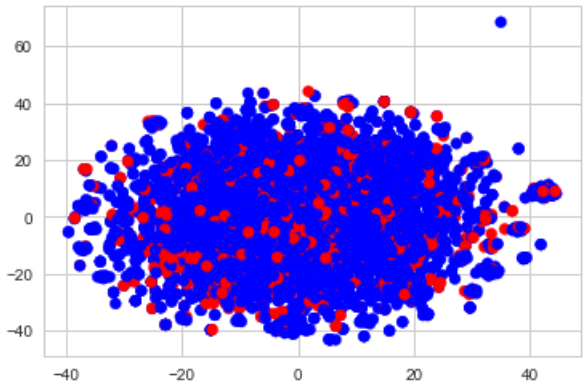
```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]
```

```
standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



In [85]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=100,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



- we need to merge all the numerical vectors i.e categorical,project_title_bow, project_title_tfidf,avg_w2v,tfidf_w2v, numerical vectors

In [86]:

```
print(school_state_one_hot.shape)
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(project_title_bow.shape)
print(project_title_tfidf.shape)
print(avg_w2v_vectors)
print(tfidf_w2v_vectors)
print(previously_posted_standardized.shape)
```

```
(109248, 51)
(109248, 9)
(109248, 30)
(109248, 3)
(109248, 6)
(109248, 3349)
(109248, 3349)
```

```
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`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
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`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
(109248, 1)
```

In [79]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((school_state_one_hot,categories_one_hot,
sub_categories_one_hot,project_grade_category_one_hot,teacher_prefix_one_hot,project_title_bow,pro
ject_title_tfidf,avg_w2v_vectors,tfidf_w2v_vectors,
previously_posted_standardized))
X.shape
```

Out[79]:

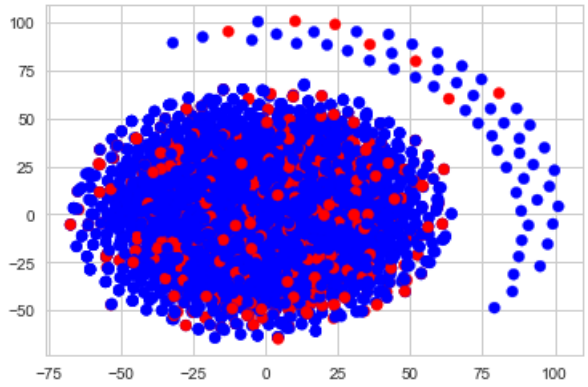
```
(109248, 7398)
```

2.5 TSNE with `BOW,TFIDF,W2V,TFIDF_W2V` encoding of `project_title` feature

In [82]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns
```

```
x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```



In [81]:

```
# https://github.com/Tejas163/Data-Science/blob/master/Project-1-Amazon%20Fine%20Food%20Reviews.py
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

x_tsne = X.tocsr()[0:5000]
y_tsne = project_data['project_is_approved'].values[0:5000]

standard_data=StandardScaler(with_mean=False).fit_transform(x_tsne)
tmodel=TSNE(n_components=2,random_state=0,perplexity=100,n_iter=1000)
tsne_data=tmodel.fit_transform(standard_data.toarray())
tsne_data = np.vstack((tsne_data.T, y_tsne)).T

for_tsne_df = pd.DataFrame(data=tsne_data, columns=(["Dimension_x", "Dimension_y", "label"]))
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['label'].apply(la
mbda x: colors[x]))
plt.show()
```