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` • `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*

Feature	Description
`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."
- __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 [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
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
```

```
C:\ProgramData\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; al
iasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:

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

In [3]:

```
#selecting first 3000 records

project_data = project_data[:3000]
project_data.shape
```

Out[3]:

```
(3000, 17)
```

In [4]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.head(2)
```

```
Number of data points in train data (3000, 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']
```

Out[4]:

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

In [5]:

```
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[5]:

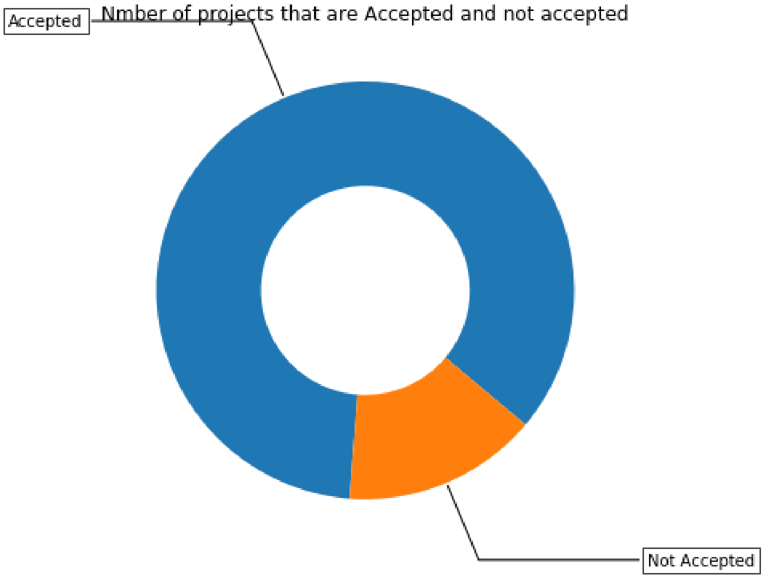
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 Data Analysis

In [6]:

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


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

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

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

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

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

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

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

plt.show()
```

```
Number of projects thar are approved for funding  2547 , ( 84.89999999999999 %)
Number of projects thar are not approved for funding  453 , ( 15.1 %)
```



1.2.1 Univariate Analysis: School State

In [7]:

```
# 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 = percentage (think about it)
temp.columns = ['state_code', 'num_proposals']
print(temp.head(1))

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

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

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

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

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

```
  state_code  num_proposals
0         AK           0.75
```

Out[7]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n\nscl = [[0.0, \'rg
b(242,240,247)\'],[0.2, \'rgb(218,218,235)\'],[0.4, \'rgb(188,189,220)\'],            [0.6, \'rgb(1
58,154,200)\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\ndata = [ dict(\n        ty
pe=\'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        colorbar = dict(title = "% of pro")\n    ) ]\n\nlayout = d
ict(\n        title = \'Project Proposals % of Acceptance Rate by US States\',\n        geo = dict(
\n            scope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n            show
akes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n        ),\n    )\n\nfig =
go.Figure(data=data, layout=layout)\noffline.iplot(fig, filename=\'us-map-heat-map\')\n'
```

In [8]:

```
# 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        0.50000
7          DC        0.62500
0          AK        0.75000
21         ME        0.75000
42         TN        0.77551
==================================================
States with highest % approvals
   state_code  num_proposals
41         SD            1.0
26         MT            1.0
28         ND            1.0
11         HI            1.0
50         WY            1.0
```

In [9]:

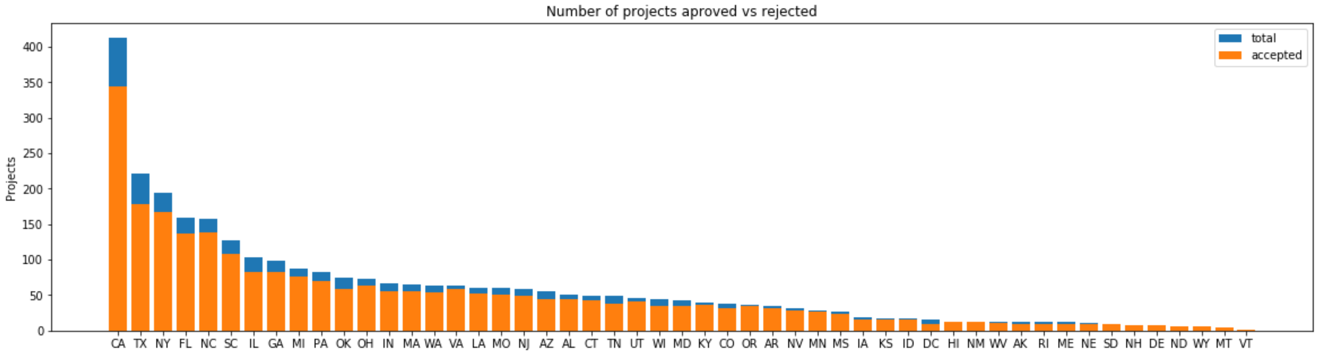
```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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 [10]:

```
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 [11]:

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



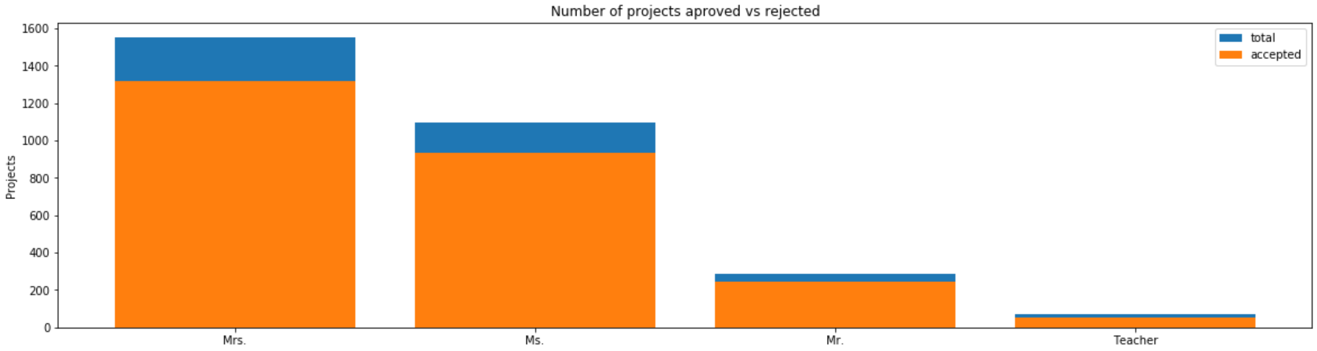
```
   school_state  project_is_approved  total       Avg
4            CA                  345    413  0.835351
43           TX                  179    221  0.809955
34           NY                  167    194  0.860825
9            FL                  137    159  0.861635
27           NC                  139    157  0.885350
==================================================
   school_state  project_is_approved  total  Avg
8            DE                    7      7  1.0
28           ND                    6      6  1.0
50           WY                    6      6  1.0
26           MT                    4      4  1.0
46           VT                    1      2  0.5
```

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

1.2.2 Univariate Analysis: teacher_prefix

In [12]:

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

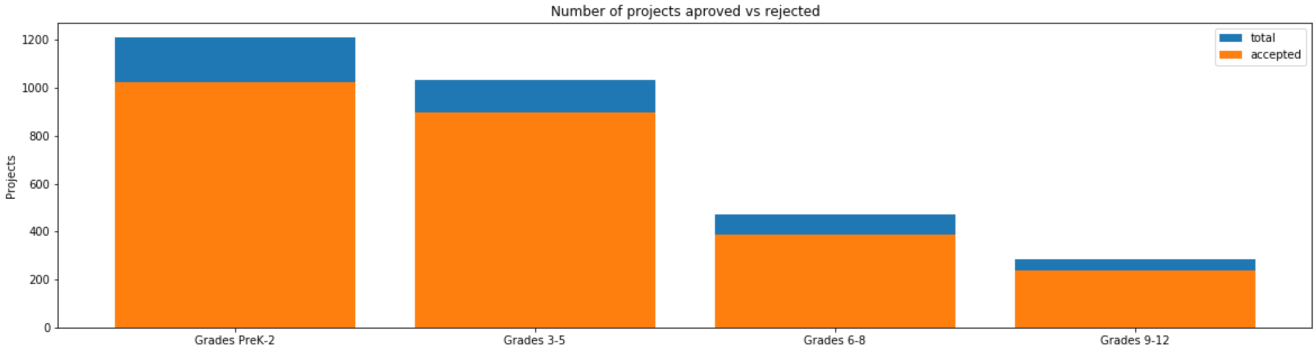


```
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 1317   1553  0.848036
2            Ms.                  933   1095  0.852055
0            Mr.                  246    284  0.866197
3        Teacher                   51     68  0.750000
==================================================
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 1317   1553  0.848036
2            Ms.                  933   1095  0.852055
0            Mr.                  246    284  0.866197
3        Teacher                   51     68  0.750000
```

1.2.3 Univariate Analysis: project_grade_category

In [13]:

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



```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 1025   1211  0.846408
0             Grades 3-5                  897   1032  0.869186
1             Grades 6-8                  388    472  0.822034
2            Grades 9-12                  237    285  0.831579
==================================================
```

```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 1025   1211  0.846408
0             Grades 3-5                  897   1032  0.869186
1             Grades 6-8                  388    472  0.822034
2            Grades 9-12                  237    285  0.831579
```

1.2.4 Univariate Analysis: project_subject_categories

In [14]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python:
https://stackoverflow.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 & 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 [15]:

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

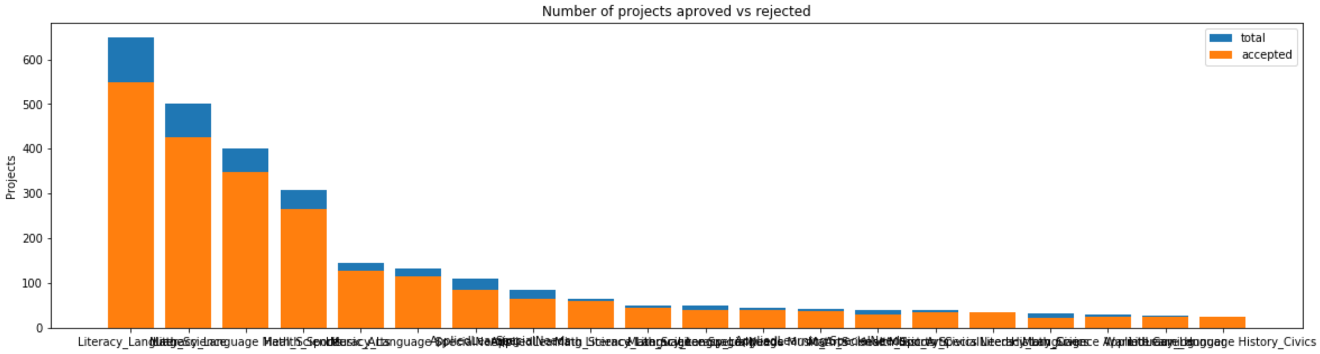
Out[15]:

Unnamed:	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate

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

In [16]:

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



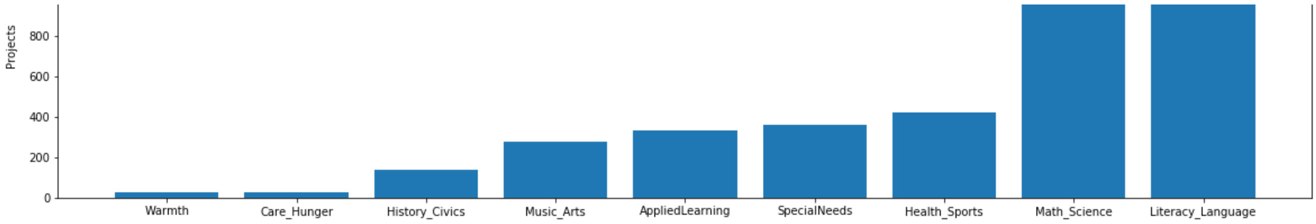
```
                    clean_categories  project_is_approved  total       Avg
21                 Literacy_Language                  548    649  0.844376
28                      Math_Science                  425    502  0.846614
24  Literacy_Language Math_Science                  348    401  0.867830
7                      Health_Sports                  266    308  0.863636
35                        Music_Arts                  128    146  0.876712
==================================================
                      clean_categories  project_is_approved  total       Avg
17  History_Civics Literacy_Language                   35     35  1.000000
14                      History_Civics                   23     32  0.718750
29       Math_Science AppliedLearning                   24     29  0.827586
43                 Warmth Care_Hunger                   26     27  0.962963
23  Literacy_Language History_Civics                   25     25  1.000000
```

In [17]:

```
# 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 [18]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))


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

plt.ylabel('Projects')
plt.title('% of projects aproved category wise')
plt.xticks(ind, list(sorted_cat_dict.keys()))
plt.show()
```



In [19]:

```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Warmth               :        29
Care_Hunger          :        29
History_Civics       :       137
Music_Arts           :       279
AppliedLearning      :       332
SpecialNeeds         :       362
Health_Sports        :       420
Math_Science         :      1143
Literacy_Language    :      1439
```

1.2.5 Univariate Analysis: project_subject_subcategories

In [20]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: 
https://stackoverflow.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 [21]:

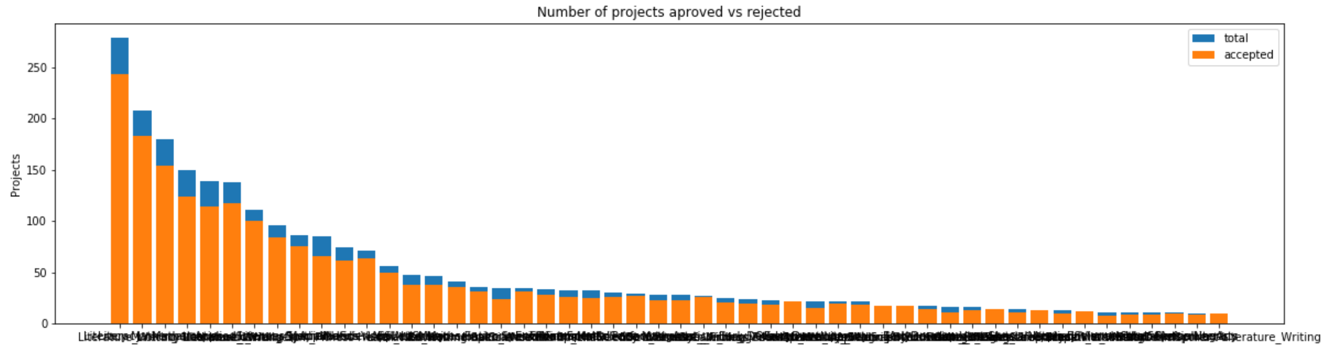
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[21]:

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

In [22]:

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



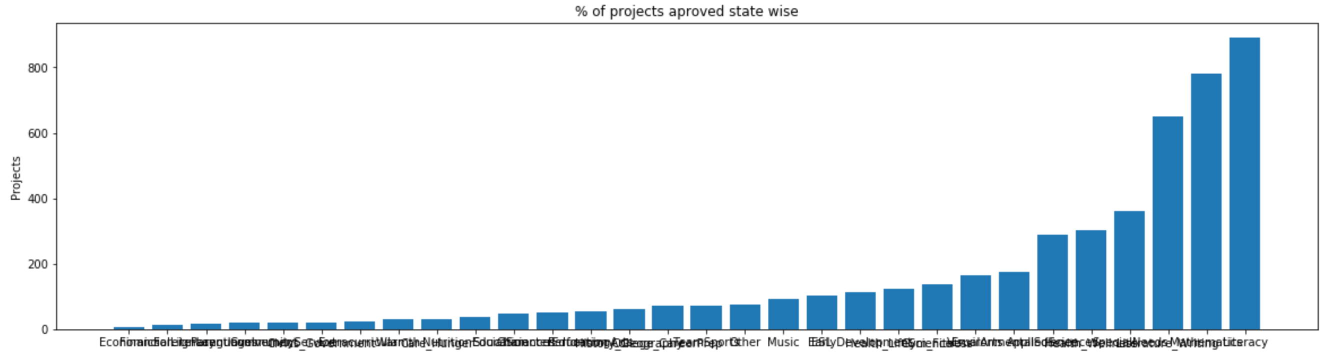
```
                clean_subcategories  project_is_approved  total       Avg
162                        Literacy                  243    279  0.870968
164            Literacy Mathematics                  183    208  0.879808
174  Literature_Writing Mathematics                  154    180  0.855556
180                     Mathematics                  124    150  0.826667
163     Literacy Literature_Writing                  114    139  0.820144
==================================================
                     clean_subcategories  project_is_approved  total       Avg
151            Health_Wellness TeamSports                    8     11  0.727273
97        EnvironmentalScience Literacy                    9     11  0.818182
199                  Other SpecialNeeds                   10     11  0.909091
204                      PerformingArts                    8     10  0.800000
79   EarlyDevelopment Literature_Writing                  10     10  1.000000
```

In [23]:

```
# 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 [24]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))


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

plt.ylabel('Projects')
plt.title('% of projects aproved state wise')
plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
plt.show()
```



In [25]:

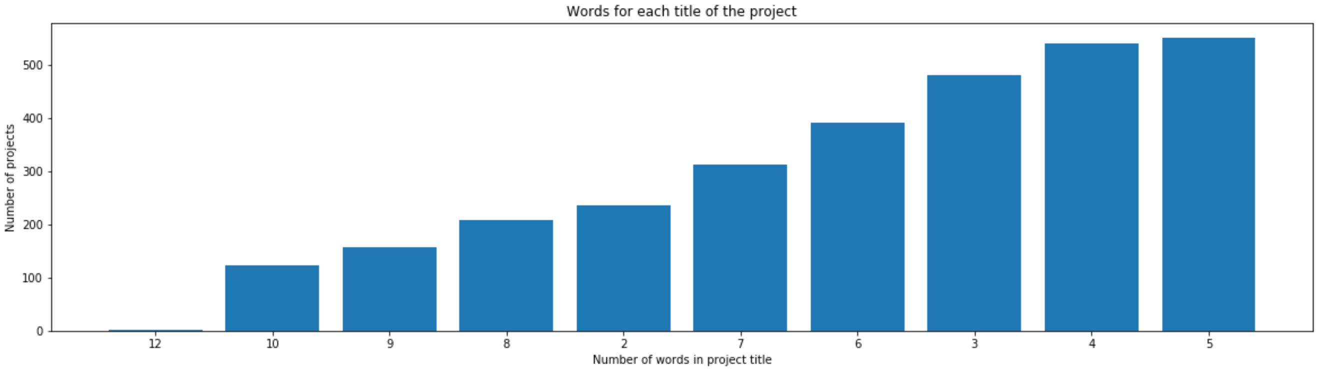
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20}:{:0}".format(i,j))
```

```
Economics           :6
FinancialLiteracy   :12
ForeignLanguages    :16
ParentInvolvement   :19
CommunityService    :19
Civics_Government   :21
Extracurricular     :24
Warmth              :29
Care_Hunger         :29
NutritionEducation  :36
SocialSciences      :49
CharacterEducation  :51
PerformingArts      :56
History_Geography   :61
College_CareerPrep  :71
TeamSports          :73
Other               :75
Music               :91
ESL                 :104
EarlyDevelopment    :113
Health_LifeScience  :125
Gym_Fitness         :139
VisualArts          :166
EnvironmentalScience:174
AppliedSciences     :289
Health_Wellness     :301
SpecialNeeds        :362
Literature_Writing  :650
Mathematics         :780
Literacy            :891
```

1.2.6 Univariate Analysis: Text features (Title)

In [26]:

```
#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()
```

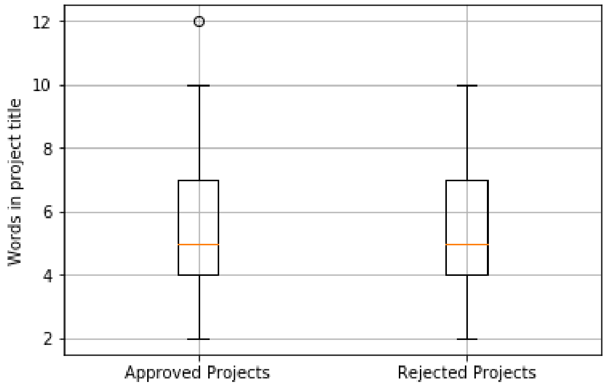


In [27]:

```
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
```

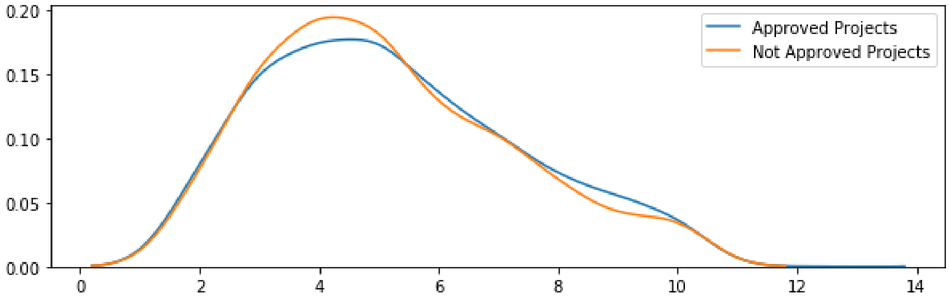
In [28]:

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



In [29]:

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



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

In [30]:

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

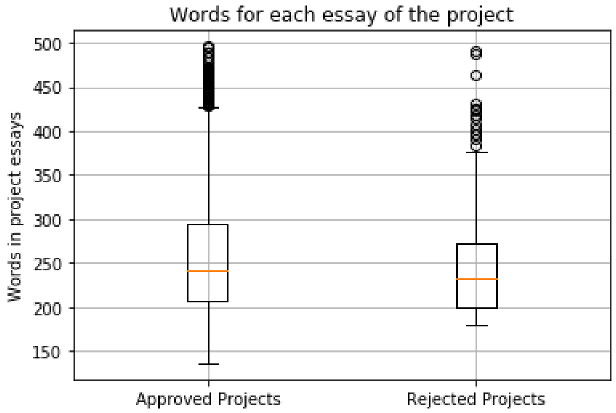
In [31]:

```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].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
```

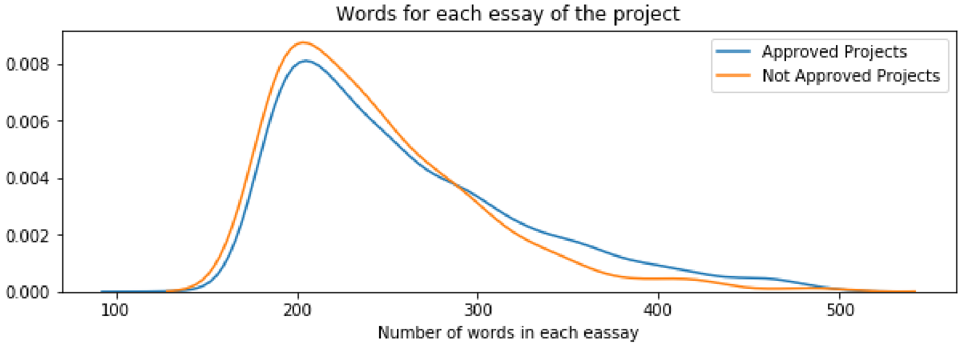
In [32]:

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



In [33]:

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



1.2.8 Univariate Analysis: Cost per project

In [34]:

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

Out[34]:

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

In [35]:

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

Out[35]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [36]:

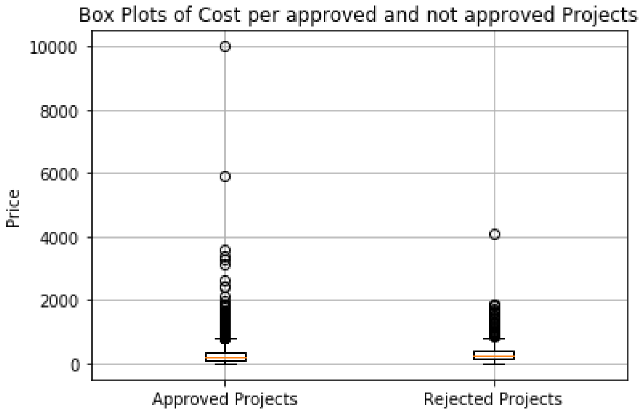
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [37]:

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

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

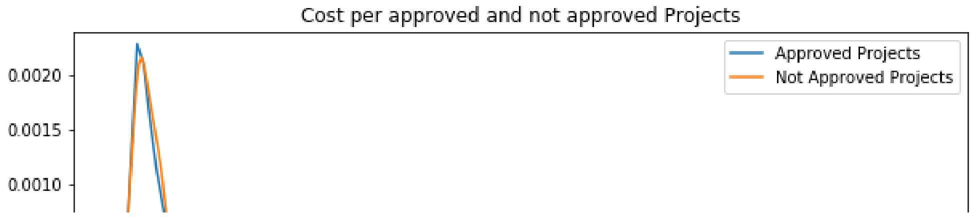
In [38]:

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



In [39]:

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



0.0005
0.0000
0
2000
4000
6000
8000
10000
Cost of a project

In [40]:

```
# 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      |        1.83       |          5.19         |
|     5      |       13.668      |         34.682        |
|     10     |       34.13       |         72.638        |
|     15     |       55.893      |         93.944        |
|     20     |       75.776      |        125.886        |
|     25     |       99.51       |         141.73        |
|     30     |       119.25      |        154.496        |
|     35     |       139.9       |         177.74        |
|     40     |       159.43      |        193.064        |
|     45     |      179.341      |        225.466        |
|     50     |       203.14      |         254.64        |
|     55     |      231.509      |        282.564        |
|     60     |       262.86      |        307.154        |
|     65     |      293.957      |         345.83        |
|     70     |      327.968      |        385.146        |
|     75     |      376.735      |         415.98        |
|     80     |      425.102      |        468.824        |
|     85     |      499.827      |        598.762        |
|     90     |      622.014      |        708.452        |
|     95     |      830.066      |        1009.878       |
|    100     |       9999.0      |        4102.47        |
+------------+-------------------+-----------------------+
```

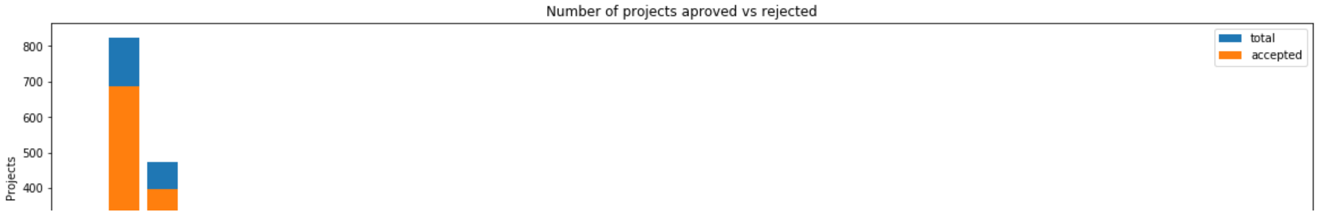
1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

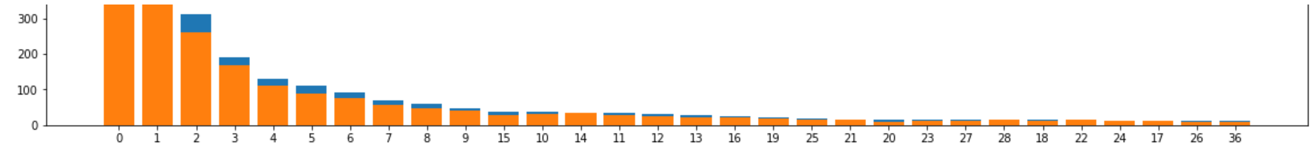
Please do this on your own based on the data analysis that was done in the above cells

In [41]:

```
# Checking the number of unique values in 'teacher_number_of_previously_posted_projects' column
# https://cmdlinetips.com/2018/01/how-to-get-unique-values-from-a-column-in-pandas-data-frame/
print(len(project_data['teacher_number_of_previously_posted_projects'].sort_values().unique()))

#reusing the function 'univariate_barplots' created at the beginning of the document
a=univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects',
'project_is_approved', 30)
```

```
135
```





```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                  687    824   
1                                             1                  397    473   
2                                             2                  261    312   
3                                             3                  168    191   
4                                             4                  111    130   

        Avg  
0  0.833738  
1  0.839323  
2  0.836538  
3  0.879581  
4  0.853846  
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved  total  \
22                                            22                   14     14   
24                                            24                   11     13   
17                                            17                   13     13   
26                                            26                   10     13   
36                                            36                   10     12   

         Avg  
22  1.000000  
24  0.846154  
17  1.000000  
26  0.769231  
36  0.833333  
```

Observations:

- There is no relation between number of projects posted by the teachers and their chances of approval.
- The above stats say that there are teachers who have been rejected even thought they have the highest number of submissions and viceversa

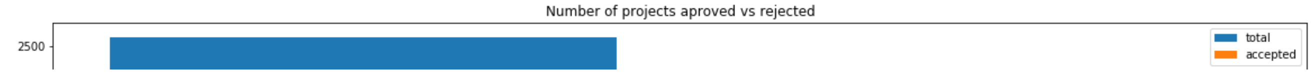
1.2.10 Univariate Analysis: project_resource_summary

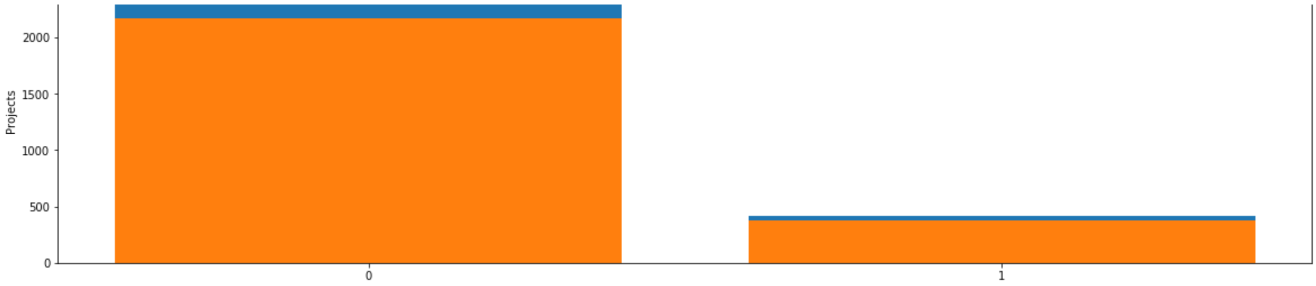
Please do this on your own based on the data analysis that was done in the above cells

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 [42]:

```
import re #importing regular expressions

#storing resource sumamries to a list
summaries = list(project_data['project_resource_summary'].values)

alphanum_flg = []
for i in summaries:
    if re.search(r'[0-9]',i): #Checking if the summary contains numbers
        alphanum_flg.append(int(1)) #assigning 1 for summaries containing numbers
    else:
        alphanum_flg.append(int(0)) #assigning 0 for summaries without numbers

project_data['alphanum_flg'] = alphanum_flg

#reusing the function 'univariate_barplots' created at the beginning of the document
univariate_barplots(project_data, 'alphanum_flg', 'project_is_approved', False)
```





```
   alphanum_flg  project_is_approved  total       Avg
0             0                 2174   2580  0.842636
1             1                  373    420  0.888095
==================================================
   alphanum_flg  project_is_approved  total       Avg
0             0                 2174   2580  0.842636
1             1                  373    420  0.888095
```

Observations:

The percentage of acceptance is 84 and 88 for summaries without numbers and with numbers respectively. Therefore there is slightly higher chance of approval, if there are numbers in the summaries.

1.3 Text preprocessing

1.3.1 Essay Text

In [43]:

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

Out[43]:

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

2 rows × 21 columns

In [44]:

```
# 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)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th 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 o
```

```
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide 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 hom
e 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 En
glish 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 ed
ucational 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 l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f 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 bea
utiful 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 t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents 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 hav
e 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 us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e 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 schools for a child who can't sit
still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with 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 each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade.  This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of 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
==================================================
```

In [45]:

```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 [46]:

```
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative,
and hard-working.  They are all unique - unique in their interests, their learning, their
abilities, and so much more.  What they all have in common is their desire to learn each day,
despite difficulties that they encounter.  \r\nOur classroom is amazing - because we understand
that everyone learns at their own pace.  As the teacher, I pride myself in making sure my students
are always engaged, motivated, and inspired to create their own learning! \r\nThis project is to h
elp my students choose seating that is more appropriate for them, developmentally.  Many students
tire of sitting in chairs during lessons, and having different seats available helps to keep them
engaged and learning.\r\nFlexible seating is important in our classroom, as many of our students s
truggle with attention, focus, and engagement.  We currently have stability balls for seating, as
well as regular chairs, but these stools will help students who have trouble with balance, or find
it difficult to sit on a stability ball for a long period of time.  We are excited to try these st
ools as a part of our engaging classroom community!nannan
==================================================
```

In [47]:

```
# \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)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative,
and hard-working.  They are all unique - unique in their interests, their learning, their
abilities, and so much more.  What they all have in common is their desire to learn each day,
despite difficulties that they encounter.    Our classroom is amazing - because we understand that
everyone learns at their own pace.  As the teacher, I pride myself in making sure my students are
always engaged, motivated, and inspired to create their own learning!   This project is to help my
students choose seating that is more appropriate for them, developmentally.  Many students tire of
sitting in chairs during lessons, and having different seats available helps to keep them engaged
and learning.  Flexible seating is important in our classroom, as many of our students struggle wi
th attention, focus, and engagement.  We currently have stability balls for seating, as well as re
gular chairs, but these stools will help students who have trouble with balance, or find it diffic
ult to sit on a stability ball for a long period of time.  We are excited to try these stools as a
part of our engaging classroom community!nannan
```

In [48]:

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

```
Describing my students is not an easy task Many would say that they are inspirational creative and
hard working They are all unique unique in their interests their learning their abilities and so m
uch more What they all have in common is their desire to learn each day despite difficulties that
they encounter Our classroom is amazing because we understand that everyone learns at their own pa
ce As the teacher I pride myself in making sure my students are always engaged motivated and inspi
red to create their own learning This project is to help my students choose seating that is more a
ppropriate for them developmentally Many students tire of sitting in chairs during lessons and hav
ing different seats available helps to keep them engaged and learning Flexible seating is
important in our classroom as many of our students struggle with attention focus and engagement We
currently have stability balls for seating as well as regular chairs but these stools will help st
udents who have trouble with balance or find it difficult to sit on a stability ball for a long pe
riod of time We are excited to try these stools as a part of our engaging classroom community nann
an
```

In [49]:

```
# 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', "do
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 [50]:

```
# 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%|████████████████████████████████████████████████████████████| 3000/3000
[00:03<00:00, 925.93it/s]
```

In [51]:

```
# after preprocesing
preprocessed_essays[2000]
```

Out[51]:

```
'describing students not easy task many would say inspirational creative hard working they unique
unique interests learning abilities much what common desire learn day despite difficulties
encounter our classroom amazing understand everyone learns pace as teacher i pride making sure stu
dents always engaged motivated inspired create learning this project help students choose seating
appropriate developmentally many students tire sitting chairs lessons different seats available he
lps keep engaged learning flexible seating important classroom many students struggle attention fo
cus engagement we currently stability balls seating well regular chairs stools help students
trouble balance find difficult sit stability ball long period time we excited try stools part enga
ging classroom community nannan'
```

1.3.2 Project title Text

In [52]:

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

Out[52]:

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

2 rows × 21 columns

In [53]:

```python
#Printing a few random review summaries

for i in range(1,3000,1000):
    sent = project_data['project_title'].values[i]
    print(sent,'--- Row No:',i)
    print("="*50)
```

```
Wanted: Projector for Hungry Learners --- Row No: 1
==================================================
Kindles for Kids! --- Row No: 1001
==================================================
Classroom Supplies --- Row No: 2001
==================================================
```

In [54]:

```python
# The above random records show that there are no URLs or HTML tags, but we will remove incase if 
there are any

from tqdm import tqdm #for status bar
from bs4 import BeautifulSoup #for html tags

preprocessed_title=[]

for title in tqdm(project_data['project_title'].values):
    # To remove urls - https://stackoverflow.com/a/40823105/4084039
    title = re.sub(r"http\S+", "", title)

    # To remove all HTML tags
    #https://stackoverflow.com/questions/16206380/python-beautifulsoup-how-to-remove-all-tags-from
-an-element
    title = BeautifulSoup(title, 'lxml').get_text()

    # To split contractions - refer decontracted function defined above
    title = decontracted(title)

    # To remove alphanumerics (words with numbers in them) - 
https://stackoverflow.com/a/18082370/4084039
    title = re.sub("\S*\d\S*", "", title).strip()

    # To remove special characters - https://stackoverflow.com/a/5843547/4084039
    title = re.sub('[^A-Za-z]+', ' ', title)

    # To remove stop words from the summaries and convert to lowercase
    title = ' '.join(e.lower() for e in title.split() if e.lower() not in stopwords)
    preprocessed_title.append(title.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 3000/3000 
[00:01<00:00, 1835.11it/s]
```

1. 4 Preparing data for models

In [55]:

```
project_data.columns
```

Out[55]:

```
Index(['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_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'alphanum_flg'],
      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 [56]:

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


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (3000, 9)
```

In [57]:

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


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

```
['Economics', 'FinancialLiteracy', 'ForeignLanguages', 'ParentInvolvement', 'CommunityService', 'C
ivics_Government', 'Extracurricular', 'Warmth', 'Care_Hunger', 'NutritionEducation',
'SocialSciences', 'CharacterEducation', 'PerformingArts', 'History_Geography',
'College_CareerPrep', 'TeamSports', 'Other', 'Music', 'ESL', 'EarlyDevelopment',
'Health_LifeScience', 'Gym_Fitness', 'VisualArts', 'EnvironmentalScience', 'AppliedSciences',
'Health_Wellness', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (3000, 30)
```

Please do the similar feature encoding with state, teacher_prefix and project_grade_category also

In [58]:

```python
# we use count vectorizer to convert the values into one hot encoded features

#https://cmdlinetips.com/2018/01/how-to-get-unique-values-from-a-column-in-pandas-data-frame/
#To get unique values from school_state column
school_state_lst=project_data['school_state'].unique()

vectorizer = CountVectorizer(vocabulary = school_state_lst, 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)
```

```
['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY', 'OK', 'MA', 'NV', 'OH', 'PA', 'A
L', 'LA', 'VA', 'AR', 'WA', 'WV', 'ID', 'TN', 'MS', 'CO', 'UT', 'IL', 'MI', 'HI', 'IA', 'RI', 'NJ',
'MO', 'DE', 'MN', 'ME', 'WY', 'ND', 'OR', 'AK', 'MD', 'WI', 'SD', 'NE', 'NM', 'DC', 'KS', 'MT', 'NH
', 'VT']
Shape of matrix after one hot encoding  (3000, 51)
```

In [59]:

```python
# we use count vectorizer to convert the values into one hot encoded features

#https://cmdlinetips.com/2018/01/how-to-get-unique-values-from-a-column-in-pandas-data-frame/
#https://stackoverflow.com/questions/48090658/sklearn-how-to-incorporate-missing-data-when-one-hot
-encoding

#replacing Nan values with 'Unknown'
project_data['teacher_prefix']=project_data['teacher_prefix'].replace(np.nan,'Unknown')

#fetching unique values
teacher_prefix_lst=project_data['teacher_prefix'].unique()

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

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

```
['Mrs.', 'Mr.', 'Ms.', 'Teacher']
Shape of matrix after one hot encoding  (3000, 4)
```

In [60]:

```python
# we use count vectorizer to convert the values into one hot encoded features

#https://cmdlinetips.com/2018/01/how-to-get-unique-values-from-a-column-in-pandas-data-frame/
#To get unique values from project_grade_category column
grade_cat_lst=project_data['project_grade_category'].unique()

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

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

```
['Grades PreK-2', 'Grades 6-8', 'Grades 3-5', 'Grades 9-12']
Shape of matrix after one hot encoding  (3000, 4)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [61]:

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

```
Shape of matrix after one hot encodig  (3000, 3327)
```

1.4.2.2 Bag of Words on `project_title`

In [62]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it

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

```
Shape of matrix after one hot encodig  (3000, 191)
```

1.4.2.3 TFIDF vectorizer

In [63]:

```
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  (3000, 3327)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [64]:

```
# Similarly you can vectorize for title also

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

```
Shape of matrix after one hot encodig  (3000, 191)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [65]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[65]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\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
word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\'
\'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the
coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus",
len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words tha
t are present in both glove vectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\n
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [66]:

```
# 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())

#https://stackoverflow.com/questions/59825/how-to-retrieve-an-element-from-a-set-without-removing-
it
print(list(glove_words)[0:10])
```

```
['authorized', 'chad', 'utencils', 'brads', 'francoise', 'sizzling', 'sensibility', 'draping', 'wa
tsi', 'eductional']
```

In [67]:

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

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

```
100%|████████████████████████████████████████████████████████████████████████████████| 3000/3000
[00:02<00:00, 1274.77it/s]
```

```
3000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [68]:

```
# Similarly you can vectorize for title also

# average Word2Vec
# compute average word2vec for each title
avg_w2v_title_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_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/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_title_vectors.append(vector)

print(len(avg_w2v_title_vectors))
print(len(avg_w2v_title_vectors[0]))
```

```
100%|███████████████████████████████████████████████████████████████████████████████| 3000/3000
[00:00<00:00, 33219.75it/s]
```

```
3000
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [69]:

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

In [70]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the 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%|██████████████████████████████████████████████████████████████████████████████| 3000/3000
[00:14<00:00, 209.64it/s]
```

```
3000
300
```

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

In [71]:

```
# Similarly you can vectorize for title also

tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_title)
# 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 [72]:

```
# average Word2Vec
# compute average word2vec for each project title.
tfidf_w2v_title_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_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 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_title_vectors.append(vector)
```

```
print(len(tfidf_w2v_title_vectors))
print(len(tfidf_w2v_title_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████| 3000/3000
[00:00<00:00, 13630.70it/s]
```

```
3000
300
```

1.4.3 Vectorizing Numerical features

In [73]:

```
# 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 mean and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

```
Mean : 302.93009, Standard deviation : 379.88810517689535
```

In [74]:

```
price_standardized
```

Out[74]:

```
array([[-3.90457316e-01],
       [-1.03453884e-02],
       [ 5.63112946e-01],
       ...,
       [-6.55324783e-01],
       [ 6.75540788e-01],
       [ 1.47333908e+01]])
```

1.4.3 Vectorizing No. of previously posted projects

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
import warnings
warnings.filterwarnings("ignore")

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

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

```
Mean : 10.508, Standard deviation : 25.69666051055638
Mean : 10.598, Standard deviation : 25.62666051855632
```

1.4.4 Merging all the above features

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

In [87]:

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

```
(3000, 9)
(3000, 30)
(3000, 3327)
(3000, 191)
(3000, 1)
(3000, 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((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[76]:

```
(3000, 3367)
```

Assignment 2: Apply TSNE

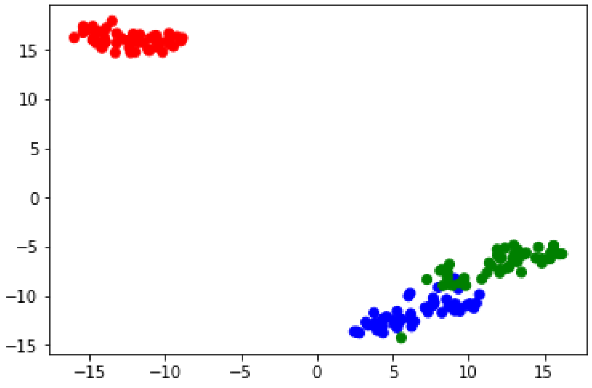
If you are using any code snippet from the internet, you have to provide the reference/citations, as we did in the above cells. Otherwise, it will be treated as plagiarism without citations.

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

In [77]:

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

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

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

X_embedding = tsne.fit_transform(x)
y=y.reshape(-1,1)
print(x.shape,y.shape)
print(type(X_embedding),type(y))

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

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

```
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
(150, 4) (150, 1)
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
```



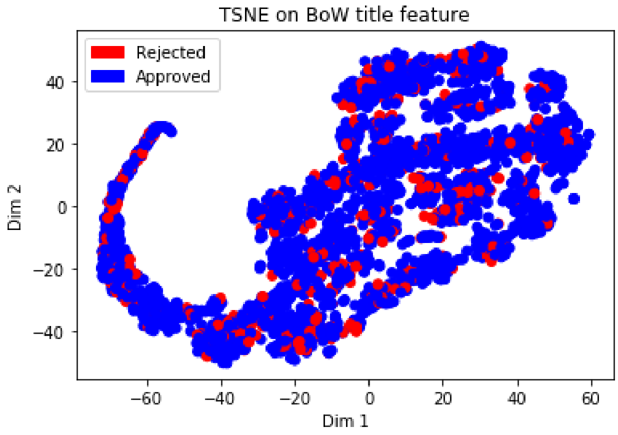
2.1.1 TSNE with `BOW` encoding of `project_title` feature

In [90]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

             school_state_one_hot, title_bow, price_standardized, prev_proj_standardized))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()
```

```
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(y.shape,type(y))

for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and n
umpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(la
mbda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on BoW title feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 291) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

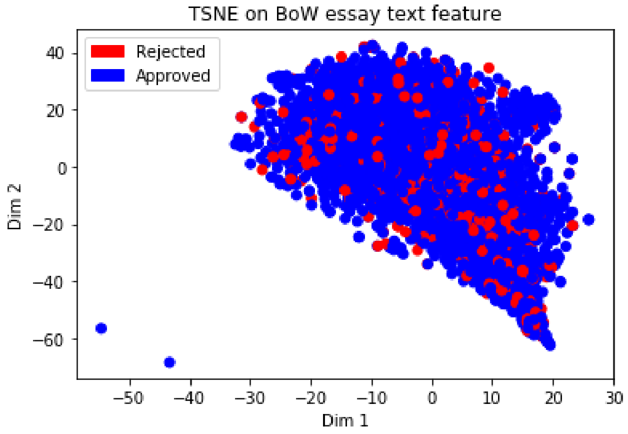
There is no clear boundary to seperate the approved and rejected points. No inference can be made

2.1.2 TSNE with `TFIDF` encoding of `essay_text` feature

In [91]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

             school_state_one_hot, text_bow, price_standardized, prev_proj_standardized))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(x.shape,type(X_embedding))
print(y.shape,type(y))

for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and numpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on BoW essay text feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 3427) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

There is no clear boundary to seperate the approved and rejected points. No inference can be made

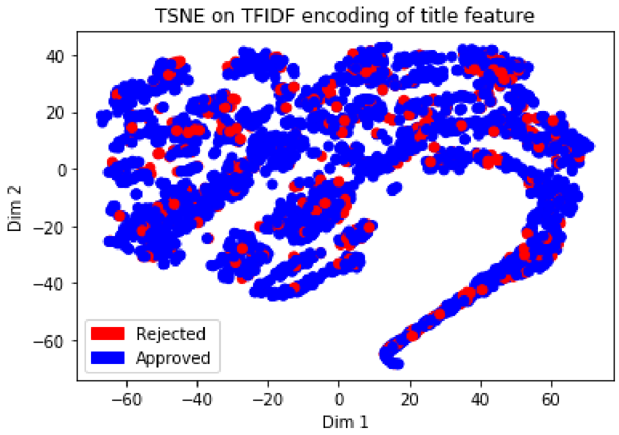
2.2.1 TSNE with `TFIDF` encoding of `project_title` feature

In [93]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

             school_state_one_hot, title_tfidf , price_standardized, prev_proj_standardized ))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(y.shape,type(y))
```

```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and n
umpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(la
mbda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on TFIDF encoding of title feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 291) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

Most of the data points are overlapping. No inference can be made

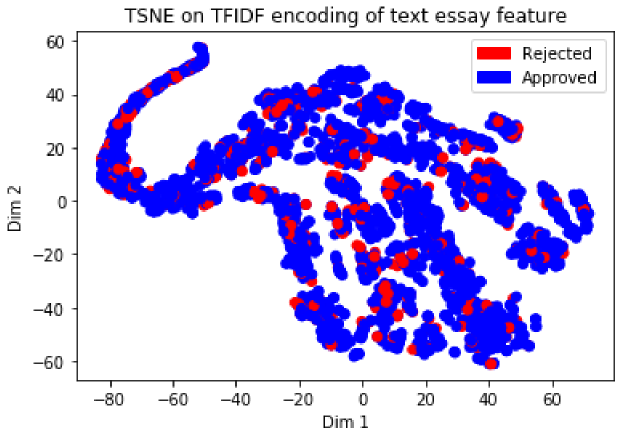
2.2.2 TSNE with `TFIDF` encoding of `text_essay` feature

In [96]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

             school_state_one_hot, text_tfidf, price_standardized, prev_proj_standardized))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(y.shape,type(y))
```

```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and n
umpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(la
mbda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on TFIDF encoding of essay text feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 3427) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

Most of the data points are overlapping. No inference can be made

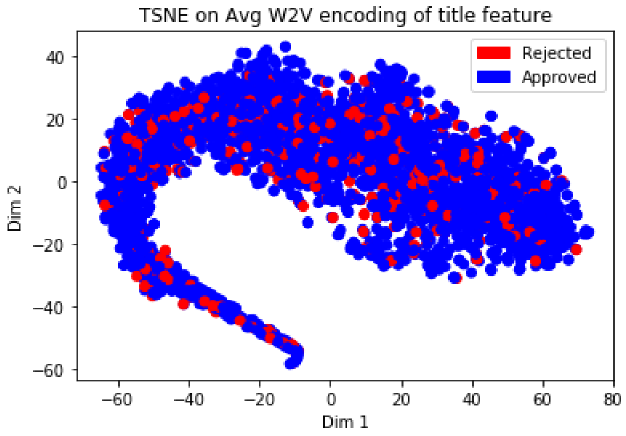
2.3.1 TSNE with `AVG W2V` encoding of `project_title` feature

In [95]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

              school_state_one_hot, avg_w2v_title_vectors, price_standardized,
prev_proj_standardized))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(y.shape,type(y))
```

```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and numpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on Avg W2V encoding of title feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 400) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

As no clustering of same colored data points are observed, it is difficult to come up with a threshold to differentiate both the groups. So, no inference can be made

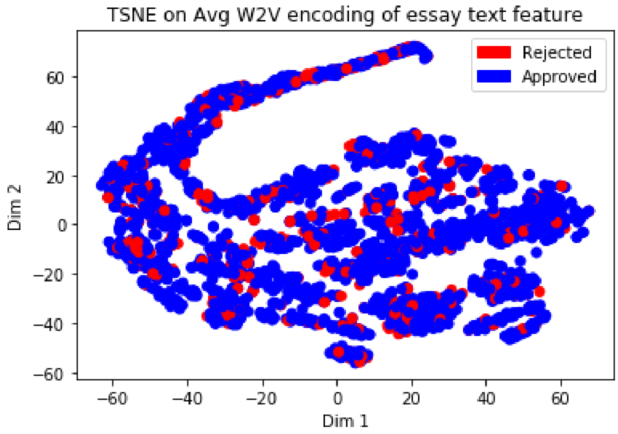
2.3.2 TSNE with `AVG W2V` encoding of `text_essay` feature

In [97]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

             school_state_one_hot, avg_w2v_vectors, price_standardized, prev_proj_standardized))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(y.shape,type(y))
```

```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and n
umpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(la
mbda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on Avg W2V encoding of essay text feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 400) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

As no clustering of same colored data points are observed, it is difficult to come up with a threshold to differentiate both the groups. So, no inference can be made

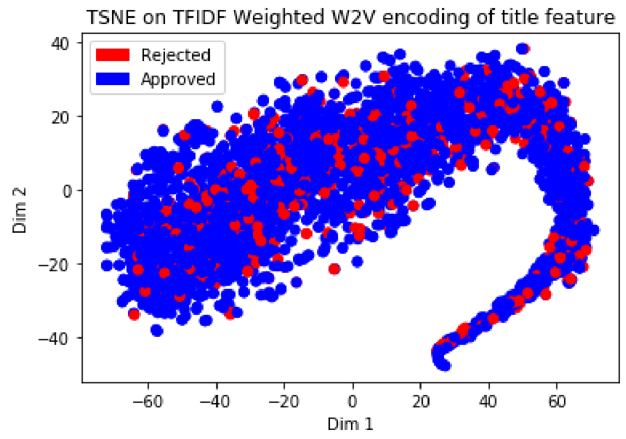
2.4.1 TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature

In [98]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

             school_state_one_hot, tfidf_w2v_title_vectors, price_standardized,
prev_proj_standardized))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(y.shape,type(y))
```

```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and numpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on TFIDF Weighted W2V encoding of title feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 400) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

Most of the data points are overlapping. No inference can be made

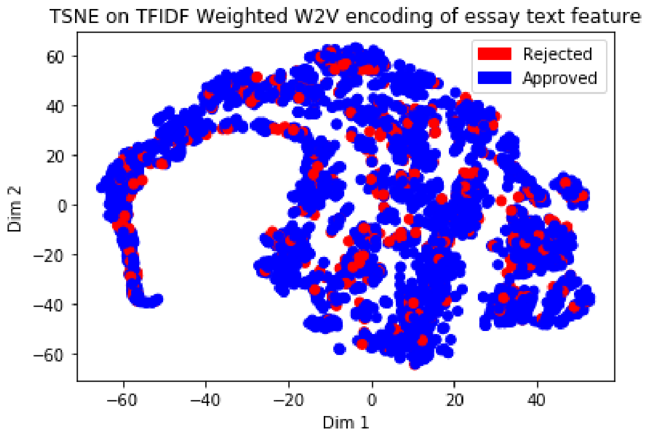
2.4.2 TSNE with `TFIDF Weighted W2V` encoding of `text_essay` feature

In [99]:

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

#https://www.digitalocean.com/community/tutorials/how-to-plot-data-in-python-3-using-matplotlib

from scipy.sparse import hstack
import matplotlib.patches as mpatches

x = hstack((categories_one_hot, sub_categories_one_hot, grade_cat_one_hot, teacher_prefix_one_hot,

             school_state_one_hot, tfidf_w2v_vectors, price_standardized, prev_proj_standardized))
y=project_data['project_is_approved']

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

X_embedding = tsne.fit_transform(x.todense()) #converting x to dense matrix as it is sparse
y=y.as_matrix()

print(x.shape,type(X_embedding))
print(y.shape,type(y))
```

```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1))) # -1 means that the dimension is unknown and numpy figures it out
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x: colors[x]))
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')

#https://stackoverflow.com/questions/39500265/manually-add-legend-items-python-matplotlib
#manually adding legends
red_patch = mpatches.Patch(color='red', label='Rejected')
blue_patch = mpatches.Patch(color='blue', label='Approved')
plt.title('TSNE on TFIDF Weighted W2V encoding of essay text feature')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```

```
(3000, 400) <class 'numpy.ndarray'>
(3000,) <class 'numpy.ndarray'>
```



Observation

Most of the data points are overlapping. No inference can be made

2.5 Summary

- There is no relation between the number of projects submitted by a teacher and their chances of approval
- The percentages of approval and rejection rate did not depend on the presence of numbers in the project summary
- TSNE with BoW, TFIDF, AvgW2V and TFIDF Weighted W2V encodings of project titles didnot produce any clear boundaries to seperate approved and rejected projects

```
In [ ]:
```