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	
`project_id`	A unique identifier for the proposed proje
`project_title`	Title • `Art Wi` •
`project_grade_category`	Grade level of students for which the project is targe • • • •

Feature	
project_subject_categories	One or more (comma-separated) subject categorie following e • • • • • I • • • •
school_state	• • Literacy & Langua State where school is located (Tw (https://en.wikipedia.org/wiki/List_of_U.S._state_abbr
project_subject_subcategories	One or more (comma-separated) subject subc • • Literature & Writin
project_resource_summary	An explanation of the resources needed f • My students need hands on literacy m se
project_essay_1	
project_essay_2	S
project_essay_3	
project_essay_4	
project_submitted_datetime	Datetime when project application was submitted. E
teacher_id	A unique identifier for the teacher of the pro bdf8baa8fedef6
teacher_prefix	Teacher's title. One of the follo • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submi

* 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 [43]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [44]:
```
#project_data = pd.read_csv('train_data.csv', nrows=5000)
project_data = pd.read_csv('train_data.csv')

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

In [45]:
```
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(3)
```

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

Out[45]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	20

In [46]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

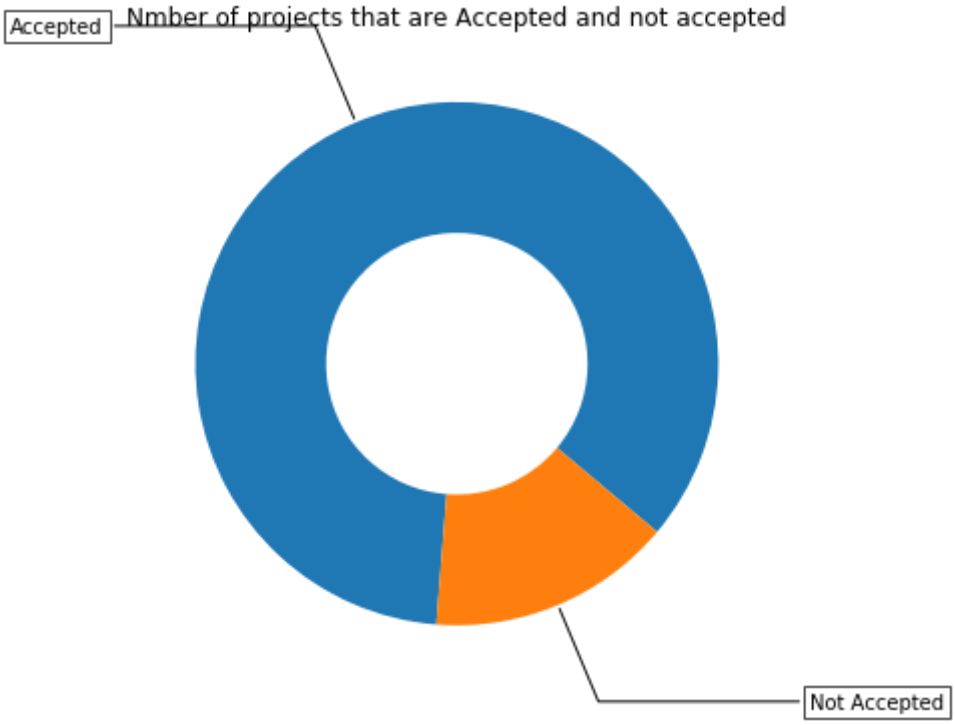
Out[46]:

	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

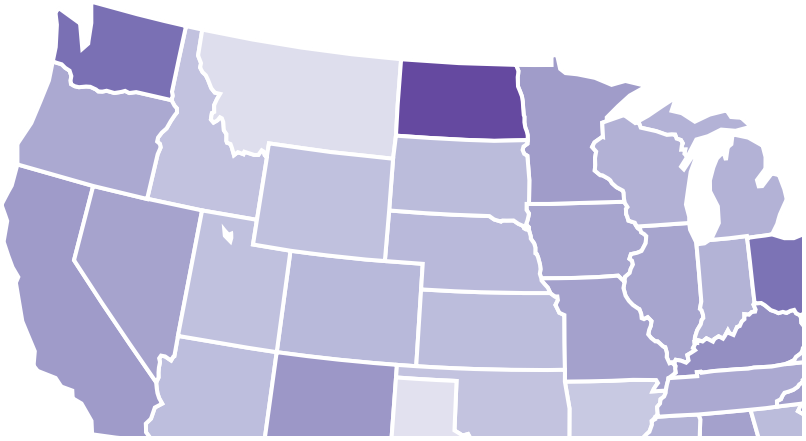
In [47]:
```
E IF YOU TAKE IT FROM ANOTHER WEBSITE.
y/pie_and_polar_charts/pie_and_donut_labels.html#sphx-glr-gallery-pie-and-pc


project_is_approved'].value_counts()
are approved for funding ", y_value_counts[1], ", (", (y_value_counts[1]/(y_
are not approved for funding ", y_value_counts[0], ", (", (y_value_counts[0]

(6, 6), subplot_kw=dict(aspect="equal"))
pted"]

lue_counts[0]]

dgeprops=dict(width=0.5), startangle=-40)

are,pad=0.3", fc="w", ec="k", lw=0.72)
coords='data', arrowprops=dict(arrowstyle="-"),
er=0, va="center")


/2. + p.theta1


"right", 1: "left"}[int(np.sign(x))]
gleA=0,angleB={}".format(ang)
onnectionstyle": connectionstyle})
x, y), xytext=(1.35*np.sign(x), 1.4*y),
nment=horizontalalignment, **kw)

 that are Accepted and not accepted")
```

```
Number of projects thar are approved for funding  92706 , ( 84.8583040421
7927 %)
Number of projects thar are not approved for funding  16542 , ( 15.141695
957820739 %)
```



1.2.1 Univariate Analysis: School State

In [48]:
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/1938559

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approv
# if you have data which contain only 0 and 1, then the mean = percentage (
temp.columns = ['state_code', 'num_proposals']
#print(temp)

# How to plot US state heatmap: https://datascience.stackexchange.com/a/962

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,1
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(

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')
```



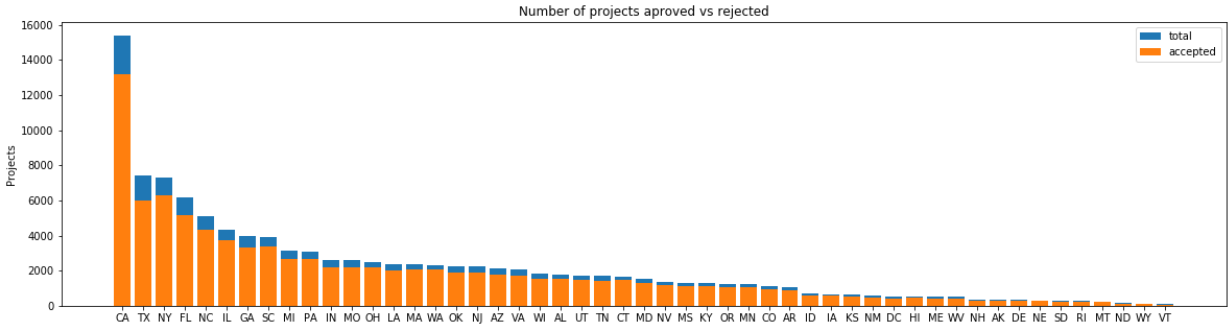
In [49]:
```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2lett
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
print('='*50)
print("States with highest % approvals")
print(temp.tail(5))
```

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

In [50]:
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_ar
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 [51]:
```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq

    # Pandas dataframe grouby count: https://stackoverflow.com/a/19385591/4
    temp['total'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'tot
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'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 [52]:
```
univariate_barplots(project_data, 'school_state', 'project_is_approved', Fa
```



	school_state	project_is_approved	total	Avg
4	CA	13205	15388	0.858136
43	TX	6014	7396	0.813142
34	NY	6291	7318	0.859661
9	FL	5144	6185	0.831690
27	NC	4353	5091	0.855038

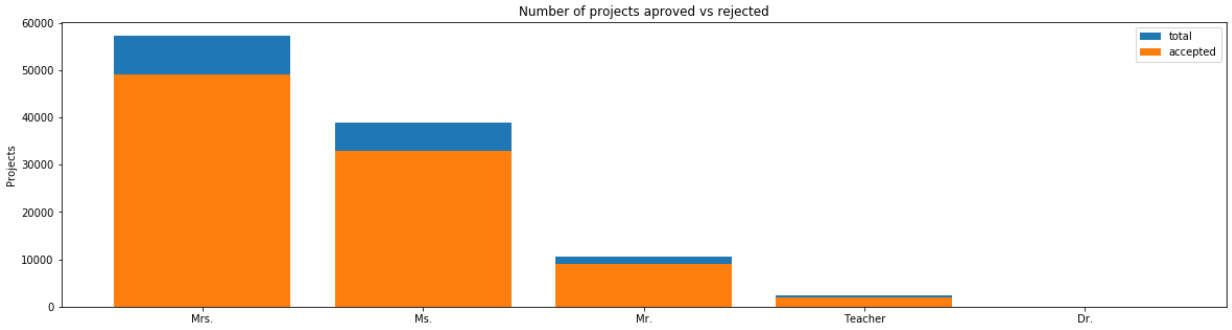
==========

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

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

1.2.2 Univariate Analysis: teacher_prefix

In [53]:
```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' ,
```



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

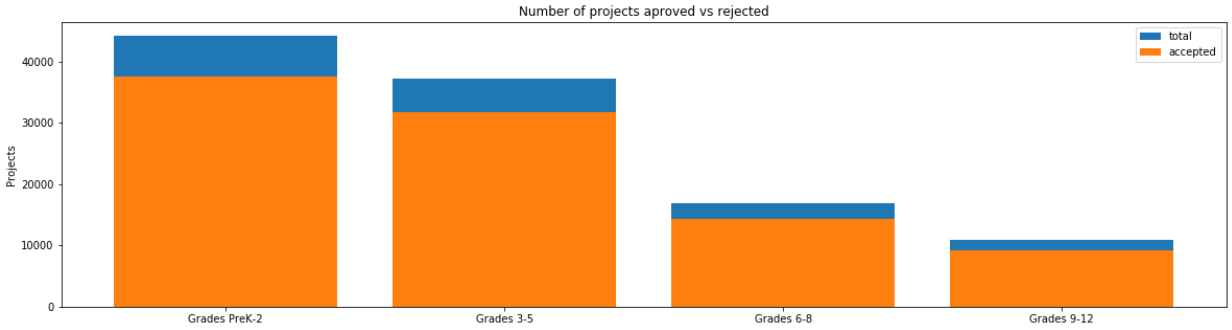
==========

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

Summary: Dr. title has least approval rate. While Mrs. has highest.

1.2.3 Univariate Analysis: project_grade_category

In [54]:
```
univariate_barplots(project_data, 'project_grade_category', 'project_is_app
```




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

In [55]:
```
### Different grades have similar approval rates
```

1.2.4 Univariate Analysis: project_subject_categories

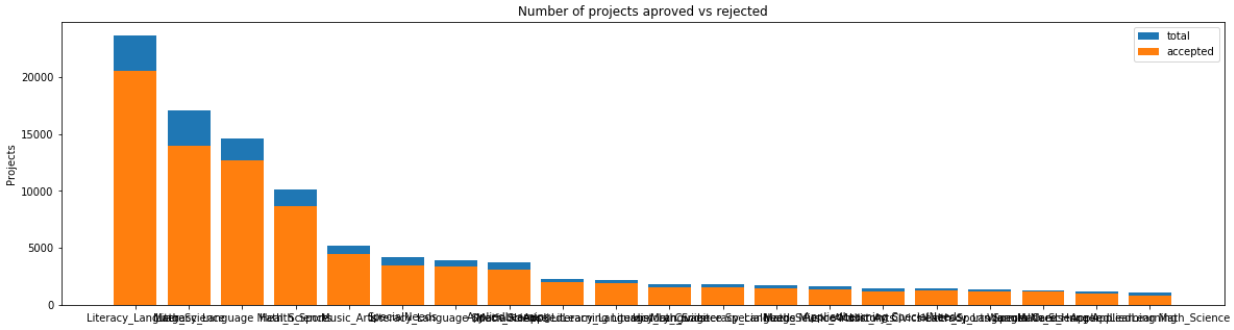
In [56]:
```
ɔgories = list(project_data['project_subject_categories'].values)
emove special characters from list of strings python: https://stackoverflow.

ttps://www.geeksforgeeks.org/removing-stop-words-nltk-python/
ttps://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fro
ttps://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string
_list = []
 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",
    if 'The' in j.split(): # this will split each of the catogory based on
        j=j.replace('The','') # if we have the words "The" we are going to
    j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empt
    temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trai
    temp = temp.replace('&','_') # we are replacing the & value into


cat_list.append(temp.strip())
```

In [57]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[57]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [58]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved'
```



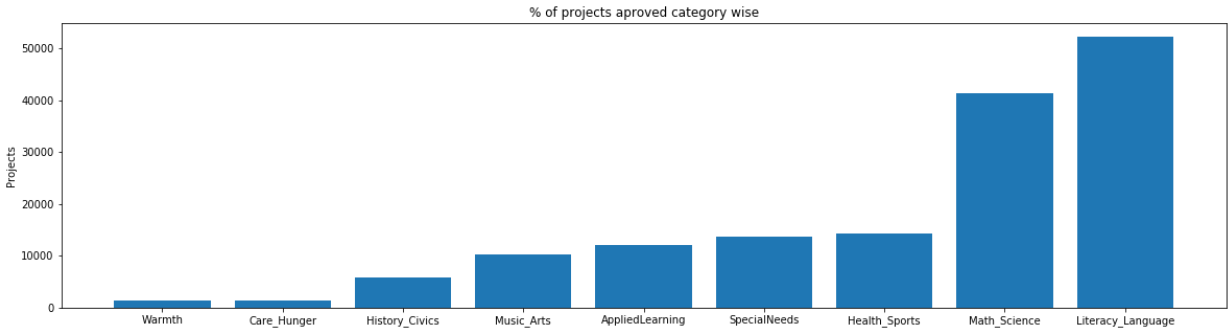

```
                clean_categories  project_is_approved  total       Avg
24              Literacy_Language                20520  23655  0.867470
32                   Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12725  14636  0.869432
8                   Health_Sports                 8640  10177  0.848973
40                     Music_Arts                 4429   5180  0.855019
==================================================
                   clean_categories  project_is_approved  total       Av
g
19  History_Civics Literacy_Language                 1271   1421  0.89444
1
14       Health_Sports SpecialNeeds                 1215   1391  0.87347
2
50               Warmth Care_Hunger                 1212   1309  0.92589
8
33      Math_Science AppliedLearning                1019   1220  0.83524
6
4       AppliedLearning Math_Science                 855   1052  0.81273
8
```

Literacy language has highest approval totals. Need to futher break down clean_categories

In [59]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

In [60]:
```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
print(cat_dict)
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()
```

```
{'Literacy_Language': 52239, 'History_Civics': 5914, 'Health_Sports': 142
23, 'Math_Science': 41421, 'SpecialNeeds': 13642, 'AppliedLearning': 1213
5, 'Music_Arts': 10293, 'Warmth': 1388, 'Care_Hunger': 1388}
```



In [61]:
```
### Literracy_Language has highest total approved projects. 2nd is Math Sci
```

In [62]:
```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

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

1.2.5 Univariate Analysis: project_subject_subcategories

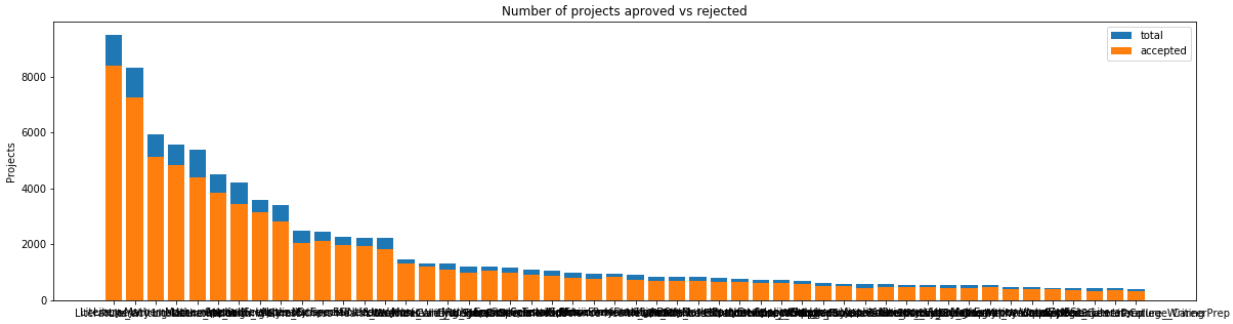
In [63]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverf

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

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

In [64]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[64]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [65]:
```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approv
```



	clean_subcategories	project_is_approved	total	Avg
317	Literacy	8371	9486	0.882458
319	Literacy Mathematics	7260	8325	0.872072
331	Literature_Writing Mathematics	5140	5923	0.867803
318	Literacy Literature_Writing	4823	5571	0.865733
342	Mathematics	4385	5379	0.815207

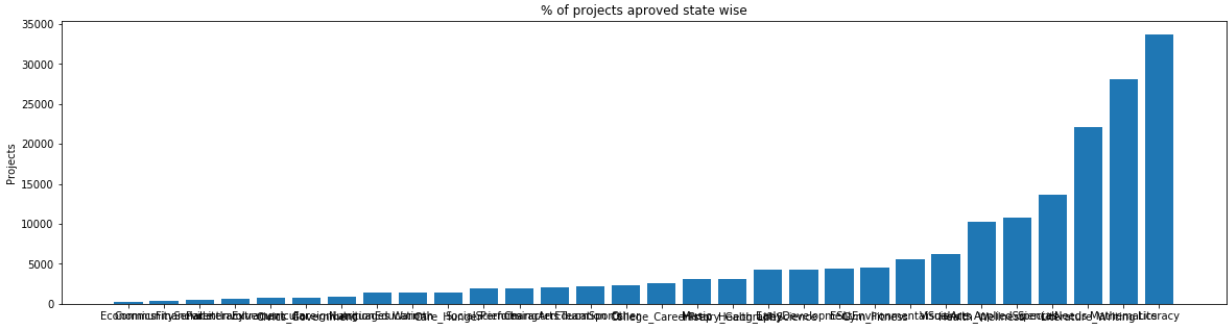
==========

	clean_subcategories	project_is_approved	total	Avg
196	EnvironmentalScience Literacy	389	444	0.876126
127	ESL	349	421	0.828979
79	College_CareerPrep	343	421	0.814727
17	AppliedSciences Literature_Writing	361	420	0.859524
3	AppliedSciences College_CareerPrep	330	405	0.814815

need to further break down subcategories

In [66]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [67]:
```
# 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()
```



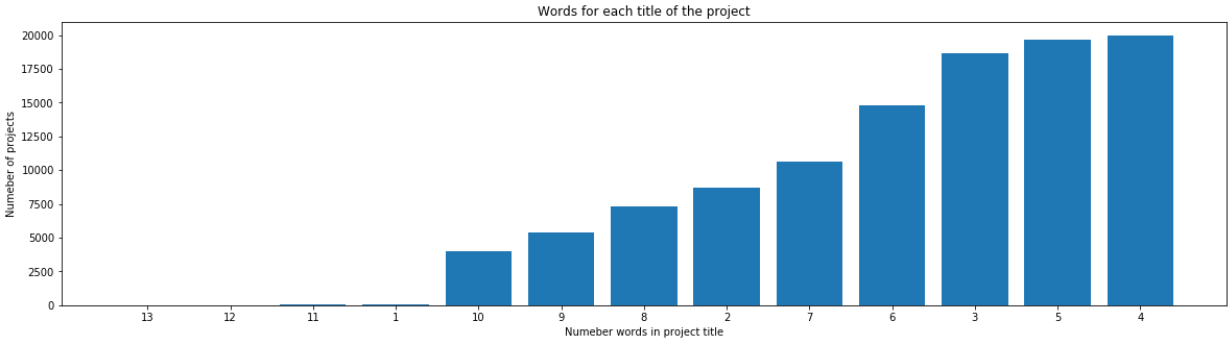
summary: Literacy and mathematics have hightes approval projects.

In [68]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
NutritionEducation   :      1355
Warmth               :      1388
Care_Hunger          :      1388
SocialSciences       :      1920
PerformingArts       :      1961
CharacterEducation   :      2065
TeamSports           :      2192
Other                :      2372
College_CareerPrep   :      2568
Music                :      3145
History_Geography    :      3171
Health_LifeScience   :      4235
EarlyDevelopment     :      4254
ESL                  :      4367
Gym_Fitness          :      4509
EnvironmentalScience :      5591
VisualArts           :      6278
Health_Wellness      :     10234
AppliedSciences      :     10816
SpecialNeeds         :     13642
Literature_Writing   :     22179
Mathematics          :     28074
Literacy             :     33700
```

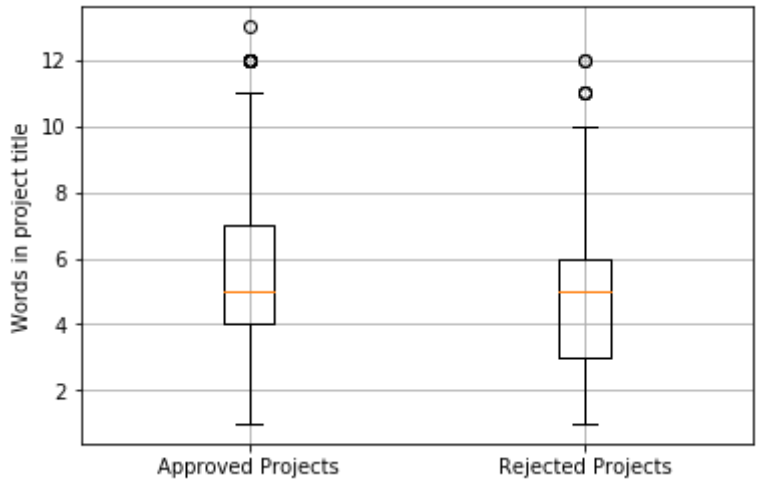
1.2.6 Univariate Analysis: Text features (Title)

In [69]:
```
#How to calculate number of words in a string in DataFrame: https://stackov
word_count = project_data['project_title'].str.split().apply(len).value_cou
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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

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

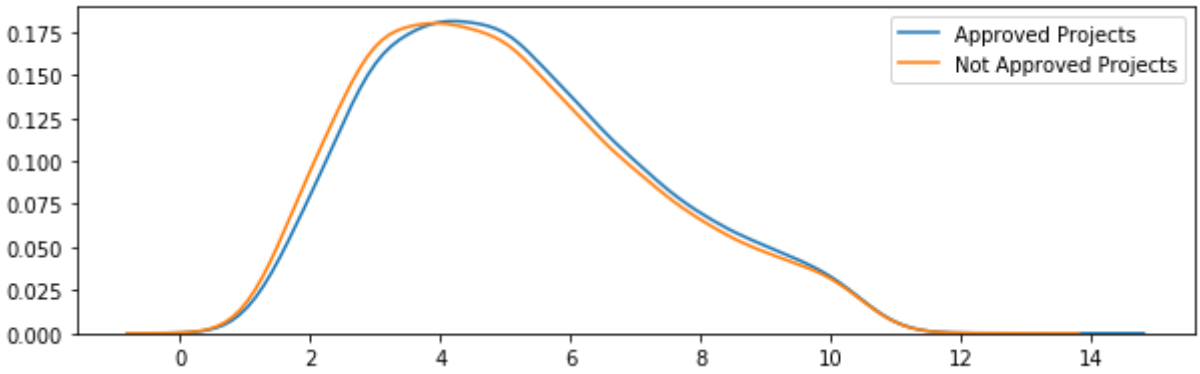


In [70]:
```
approved_title_word_count = project_data[project_data['project_is_approved'
approved_title_word_count = approved_title_word_count.values

rejected_title_word_count = project_data[project_data['project_is_approved'
rejected_title_word_count = rejected_title_word_count.values
```

In [71]:
```
# 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 [72]:
```
plt.figure(figsize=(10,3))
sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)
sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6
plt.legend()
plt.show()
```



Summary: Title with 4 and 5 words have hightest approval projects count

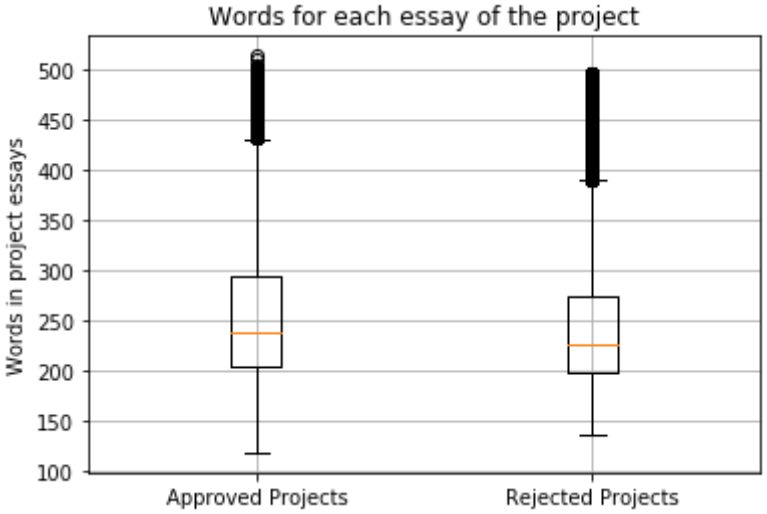
1.2.7 Univariate Analysis: Text features (Project Essay's)

In [73]:
```
# merge two column text dataframe:
print(type(project_data["project_essay_1"][0]))
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)
```

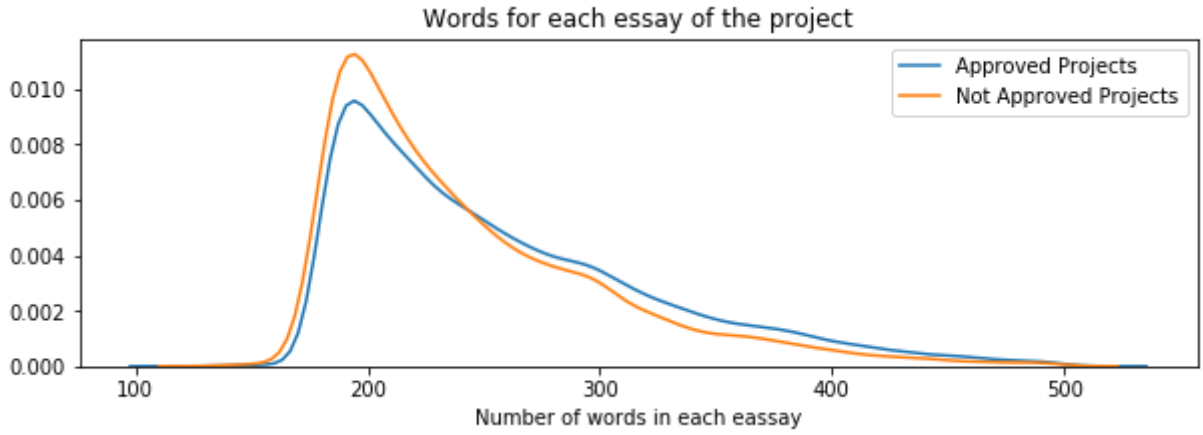
```
<class 'str'>
```

In [74]:
```
approved_word_count = project_data[project_data['project_is_approved']==1][
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0][
rejected_word_count = rejected_word_count.values
```

In [75]:
```
# 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 [171]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects"
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```



summary: Essay with around 200 words has hightes approval projects

1.2.8 Univariate Analysis: Cost per project

In [38]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[38]:

	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 [174]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-in
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'su
price_data.head(2)
```

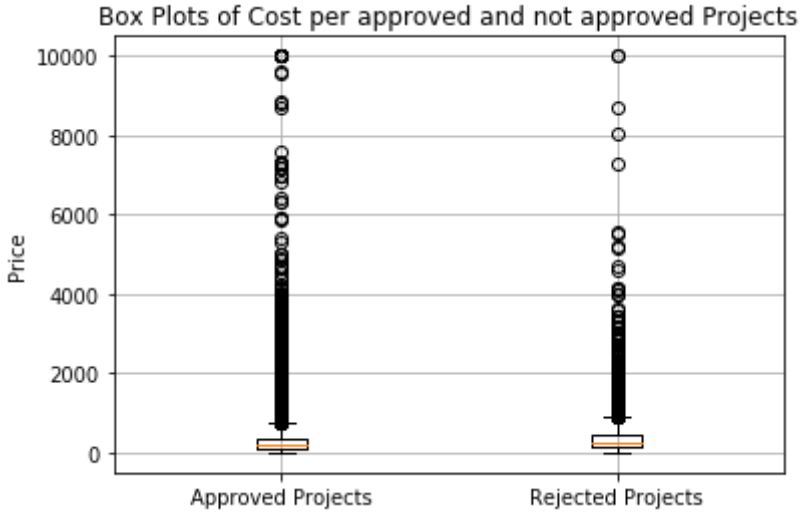
Out[174]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

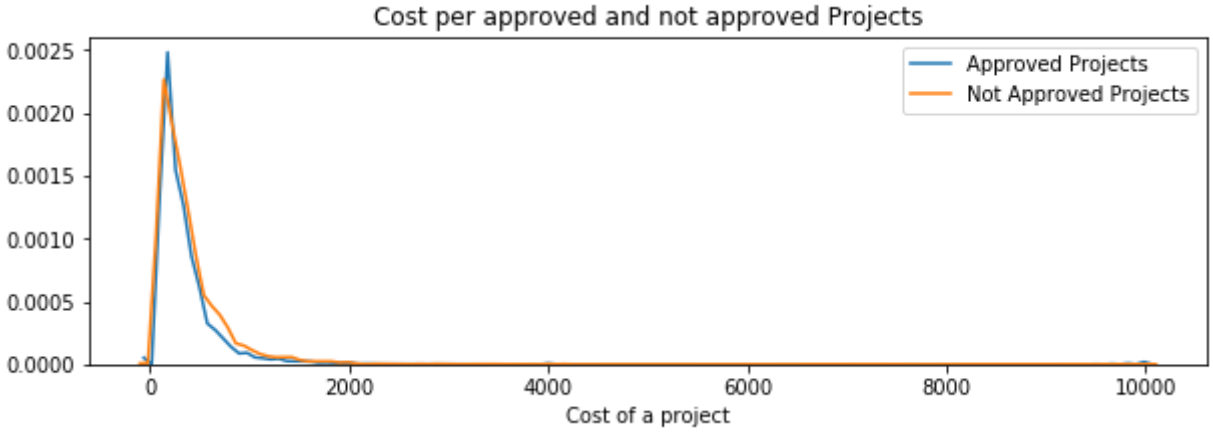
In [175]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [176]:
```
approved_price = project_data[project_data['project_is_approved']==1]['pric

rejected_price = project_data[project_data['project_is_approved']==0]['pric
```

In [177]:
```
# 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 [178]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_price, hist=False, label="Approved Projects")
sns.distplot(rejected_price, hist=False, label="Not Approved Projects")
plt.title('Cost per approved and not approved Projects')
plt.xlabel('Cost of a project')
plt.legend()
plt.show()
```



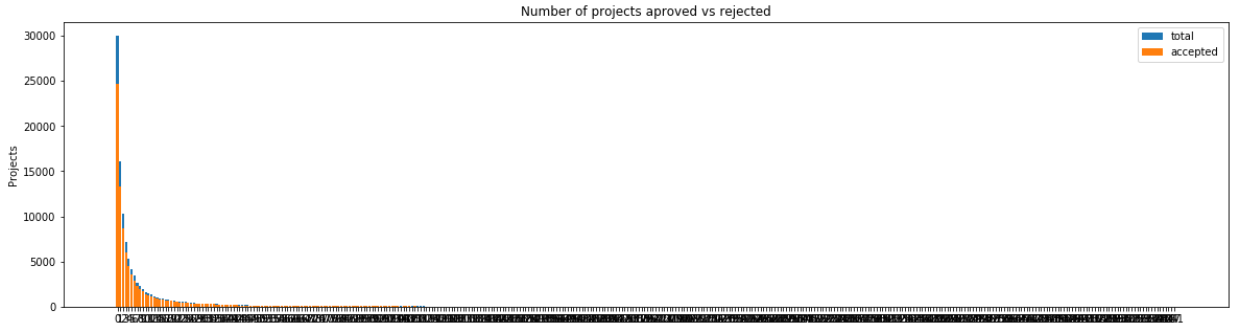
summary: Cost of a project does not seem to be a key factor for approval or not

In [45]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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.
print(x)
```

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

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

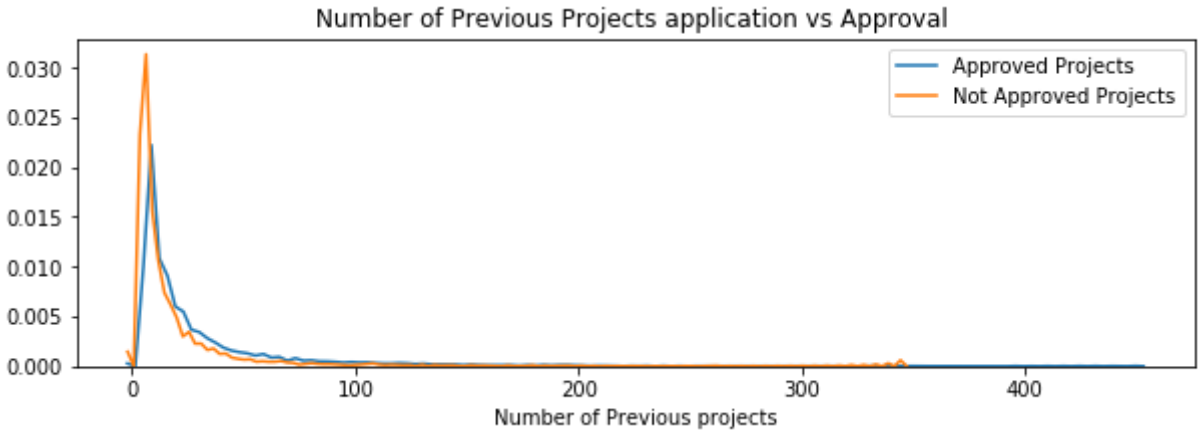
In [46]:
```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_proj
```




```
   teacher_number_of_previously_posted_projects  project_is_approved  tot
al  \
0                                             0                24652  300
14
1                                             1                13329  160
58
2                                             2                 8705  103
50
3                                             3                 5997   71
10
4                                             4                 4452   52
66

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
     teacher_number_of_previously_posted_projects  project_is_approved  t
otal  \
242                                           242                   1
1
268                                           270                   1
1
234                                           234                   1
1
335                                           347                   1
1
373                                           451                   1
1

     Avg
242  1.0
268  1.0
234  1.0
335  1.0
373  1.0
```

In [47]:
```
approved_pre_projects = project_data[project_data['project_is_approved']==1

rejected_pre_projects = project_data[project_data['project_is_approved']==0
```

In [50]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_pre_projects, hist=False, label="Approved Projects")
sns.distplot(rejected_pre_projects, hist=False, label="Not Approved Project
plt.title('Number of Previous Projects application vs Approval ')
plt.xlabel('Number of Previous projects')
plt.legend()
plt.show()
```



summary: Previous no projects submitted has hights total approvals. Previous 2-4 submittions has highest approval rate

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

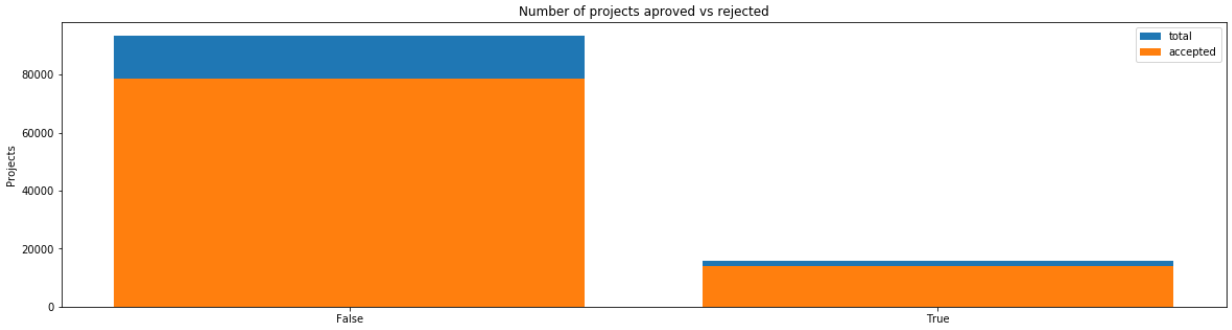
Check if the `presence of the numerical digits` in the `project_resource_summary` effects the acceptance of the project or not. If you observe that `presence of the numerical digits` is helpful in the classification, please include it for further process or you can ignore it.

In [68]:
```
#print(project_data['project_resource_summary'].values[0:50])
# check if string has number https://stackoverflow.com/questions/19859282/c

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

In [76]:
```
from tqdm import tqdm
with_dig = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_resource_summary'].values):
    dig = hasNumbers(sentance)
    with_dig.append(dig)

project_data['project_resource_summary_with_dig'] = with_dig
```

```
100%|██████████| 109248/109248 [00:00<00:00, 233017.24it/s]
```

In [77]:
```
univariate_barplots(project_data, 'project_resource_summary_with_dig', 'prc
```




```
   project_resource_summary_with_dig  project_is_approved  total       Av
g
0                              False               78616  93492  0.84088
5
1                               True               14090  15756  0.89426
3
==================================================
   project_resource_summary_with_dig  project_is_approved  total       Av
g
0                              False               78616  93492  0.84088
5
1                               True               14090  15756  0.89426
3
```

Summary: Containing digits in project resrouce summary has higher approval rates 89% than those not 84%.

1.3 Text preprocessing

1.3.1 Essay Text

In [93]:
```
project_data.head(2)
```

Out[93]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [94]:
```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their sec
ond or third languages. We are a melting pot of refugees, immigrants, and
native-born Americans bringing the gift of language to our school. \r\n\r
\n We have over 24 languages represented in our English Learner program w
ith students at every level of mastery.  We also have over 40 countries r
epresented with the families within our school.  Each student brings a we
alth of knowledge and experiences to us that open our eyes to new culture
s, beliefs, and respect.\"The limits of your language are the limits of y
our world.\"-Ludwig Wittgenstein  Our English learner's have a strong sup
port system at home that begs for more resources.  Many times our parents
are learning to read and speak English along side of their children.  Som
etimes this creates barriers for parents to be able to help their child l
earn phonetics, letter recognition, and other reading skills.\r\n\r\nBy p
roviding these dvd's and players, students are able to continue their mas
tery of the English language even if no one at home is able to assist.  A
ll families with students within the Level 1 proficiency status, will be
a offered to be a part of this program.  These educational videos will be
specially chosen by the English Learner Teacher and will be sent home reg
ularly to watch.  The videos are to help the child develop early reading
skills.\r\n\r\nParents that do not have access to a dvd player will have
the opportunity to check out a dvd player to use for the year.  The plan
is to use these videos and educational dvd's for the years to come for ot
her EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this yea
r all love learning, at least most of the time. At our school, 97.3% of t
he students receive free or reduced price lunch. Of the 560 students, 97.
3% are minority students. \r\nThe school has a vibrant community that lov
es to get together and celebrate. Around Halloween there is a whole schoo
l parade to show off the beautiful costumes that students wear. On Cinco
de Mayo we put on a big festival with crafts made by the students, dance
s, and games. At the end of the year the school hosts a carnival to celeb
rate the hard work put in during the school year, with a dunk tank being
the most popular activity.My students will use these five brightly colore
d Hokki stools in place of regular, stationary, 4-legged chairs. As I wil
l only have a total of ten in the classroom and not enough for each stude
nt to have an individual one, they will be used in a variety of ways. Dur
ing 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 s
mall group tables during math and reading times. The rest of the day they
will be used by the students who need the highest amount of movement in t
heir 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 a
```

re sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as 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 the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in 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 and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical 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 to make our new school year a very successful one. Thank you!nannan

==========

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

```
==================================================
The mediocre teacher tells. The good teacher explains. The superior teach
er demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy s
chool has 803 students which is makeup is 97.6% African-American, making
up the largest segment of the student body. A typical school in Dallas is
made up of 23.2% African-American students. Most of the students are on f
ree or reduced lunch. We aren't receiving doctors, lawyers, or engineers
children from rich backgrounds or neighborhoods. As an educator I am insp
iring minds of young children and we focus not only on academics but one
smart, effective, efficient, and disciplined students with good characte
r.In our classroom we can utilize the Bluetooth for swift transitions dur
ing class. I use a speaker which doesn't amplify the sound enough to rece
ive the message. Due to the volume of my speaker my students can't hear v
ideos or books clearly and it isn't making the lessons as meaningful. But
with the bluetooth speaker my students will be able to hear and I can sto
p, pause and replay it at any time.\r\nThe cart will allow me to have mor
e room for storage of things that are needed for the day and has an extra
part to it I can use.  The table top chart has all of the letter, words a
nd pictures for students to learn about different letters and it is more
accessible.nannan
==================================================
```

In [95]:
```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 [96]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. Th
ey are eager beavers and always strive to work their hardest working past
their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students r
eceive free or reduced price lunch.  Despite their disabilities and limit
ations, my students love coming to school and come eager to learn and exp
lore.Have you ever felt like you had ants in your pants and you needed to
groove and move as you were in a meeting? This is how my kids feel all th
e time. The want to be able to move as they learn or so they say.Wobble c
hairs are the answer and I love then because they develop their core, whi
ch enhances gross motor and in Turn fine motor skills. \r\nThey also want
to learn through games, my kids do not want to sit and do worksheets. The
y want to learn to count by jumping and playing. Physical engagement is t
he key to our success. The number toss and color and shape mats can make
that happen. My students will forget they are doing work and just have th
e fun a 6 year old deserves.nannan
==================================================
```

In [97]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-li
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. Th
ey are eager beavers and always strive to work their hardest working past
their limitations.     The materials we have are the ones I seek out for
my students. I teach in a Title I school where most of the students recei
ve free or reduced price lunch.  Despite their disabilities and limitatio
ns, my students love coming to school and come eager to learn and explor
e.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the
time. The want to be able to move as they learn or so they say.Wobble cha
irs are the answer and I love then because they develop their core, which
enhances gross motor and in Turn fine motor skills.   They also want to l
earn through games, my kids do not want to sit and do worksheets. They wa
nt to learn to count by jumping and playing. Physical engagement is the k
ey to our success. The number toss and color and shape mats can make that
happen. My students will forget they are doing work and just have the fun
a 6 year old deserves.nannan
```

In [98]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and
language delays cognitive delays gross fine motor delays to autism They a
re eager beavers and always strive to work their hardest working past the
ir limitations The materials we have are the ones I seek out for my stude
nts I teach in a Title I school where most of the students receive free o
r reduced price lunch Despite their disabilities and limitations my stude
nts love coming to school and come eager to learn and explore Have you ev
er felt like you had ants in your pants and you needed to groove and move
as you were in a meeting This is how my kids feel all the time The want t
o be able to move as they learn or so they say Wobble chairs are the answ
er and I love then because they develop their core which enhances gross m
otor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count
by jumping and playing Physical engagement is the key to our success The
number toss and color and shape mats can make that happen My students wil
l forget they are doing work and just have the fun a 6 year old deserves
nannan
```

In [99]:
```
# 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', 'y
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', '
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have'
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'be
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on'
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', '
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "sho
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "sh
            'won', "won't", 'wouldn', "wouldn't"]
```

In [100]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [01:26<00:00, 1269.07it/s]
```

In [101]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[101]:
```
'my kindergarten students varied disabilities ranging speech language del
ays cognitive delays gross fine motor delays autism they eager beavers al
ways strive work hardest working past limitations the materials ones i se
ek students i teach title i school students receive free reduced price lu
nch despite disabilities limitations students love coming school come eag
er learn explore have ever felt like ants pants needed groove move meetin
g this kids feel time the want able move learn say wobble chairs answer i
love develop core enhances gross motor turn fine motor skills they also w
ant learn games kids not want sit worksheets they want learn count jumpin
g playing physical engagement key success the number toss color shape mat
s make happen my students forget work fun 6 year old deserves nannan'
```

1.3.2 Project title Text

In [102]:
```
# similarly you can preprocess the titles also

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

```
100%|██████████| 109248/109248 [00:04<00:00, 24796.99it/s]
```

In [103]:
```
preprocessed_titles[2000]
```

Out[103]:
```
'steady stools active learning'
```

1. 4 Preparing data for models

In [104]:
```
project_data.columns
```

Out[104]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_stat
e',
       'project_submitted_datetime', 'project_grade_category', 'project_t
itle',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approv
ed',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

1.4.1 Vectorizing Categorical data

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

In [105]:
```
# we use count vectorizer to convert the values into one hot encoded featur
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lower
vectorizer.fit(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())

categories_one_hot = vectorizer.transform(project_data['clean_categories'].

categories_one_hot = categories_one_hot[0:5000]
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (5000, 9)
```

In [106]:
```
# we use count vectorizer to convert the values into one hot encoded featur
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), l
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot = vectorizer.transform(project_data['clean_subcatego
sub_categories_one_hot = sub_categories_one_hot[0:5000]


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutritio
nEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness',
'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics',
'Literacy']
Shape of matrix after one hot encodig  (5000, 30)
```

In [107]:
```
# Please do the similar feature encoding with state, teacher_prefix and pro

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


state_one_hot = vectorizer.transform(project_data['school_state'].values)
state_one_hot = state_one_hot[0:5000]


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

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI',
'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN',
'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH',
'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA',
'WI', 'WV', 'WY']
Shape of matrix after one hot encodig  (5000, 51)
```

In [108]:
```
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].apply(lambda x: np.str_(x)))
print(vectorizer.get_feature_names())


tp_one_hot = vectorizer.transform(project_data['teacher_prefix'].apply(lamb
tp_one_hot = tp_one_hot[0:5000]


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

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher', 'nan']
Shape of matrix after one hot encodig  (5000, 6)
```

In [109]:
```
from collections import Counter
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.splitlines())



grade_list = dict(my_counter)
print(grade_list)


# If not generating the above list and put into vocabulary, the vector will
# This is because of space and new lines. Otherwise no need for vocabulary

vectorizer = CountVectorizer(vocabulary=list(grade_list.keys()),lowercase=F



vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())

pg_one_hot = vectorizer.transform(project_data['project_grade_category'].va
pg_one_hot = pg_one_hot[0:5000]


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

```
{'Grades PreK-2': 44225, 'Grades 6-8': 16923, 'Grades 3-5': 37137, 'Grade
s 9-12': 10963}
['Grades PreK-2', 'Grades 6-8', 'Grades 3-5', 'Grades 9-12']
Shape of matrix after one hot encodig  (5000, 4)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [110]:
```
# We are considering only the words which appeared in at least 10 documents
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)[0:5000]
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (5000, 16623)
```

In [111]:
```
text_bow[50,20]
```

Out[111]:
```
0
```

1.4.2.2 Bag of Words on `project_title`

In [112]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

In [113]:
```
# Similarly you can vectorize for title also

vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)[0:5000]
print("Shape of matrix after one hot encodig ",title_bow.shape)
```

```
Shape of matrix after one hot encodig  (5000, 3329)
```

1.4.2.3 TFIDF vectorizer

In [114]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)[0:5000]
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (5000, 16623)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [115]:
```
# 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_titles)[0:5000]
print("Shape of matrix after one hot encodig ",title_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (5000, 3329)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [186]:

```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/408
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 c
      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/

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

```
521it [00:00, 5204.78it/s]

Loading Glove Model

1917495it [05:55, 5394.96it/s]

Done. 1917495  words loaded!

---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
<ipython-input-186-0c5ccbbe33a9> in <module>
     26 '''
     27 words = []
---> 28 for i in preproced_texts:
     29     words.extend(i.split(' '))
     30

NameError: name 'preproced_texts' is not defined
```

In [187]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [188]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in t
for sentence in tqdm(preprocessed_essays[0:5000]): # for each review/senten
    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%|██████████| 5000/5000 [00:03<00:00, 1665.98it/s]

5000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [189]:
```
# Similarly you can vectorize for title also
avg_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stor
for sentence in tqdm(preprocessed_titles[0:5000]): # for each review/senten
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles.append(vector)

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

```
100%|██████████| 5000/5000 [00:00<00:00, 11589.50it/s]

5000
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [190]:
```
# 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 [192]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(preprocessed_essays[0:5000]): # for each review/senten
    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/re
    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 t
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.sp
            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%|██████████| 5000/5000 [00:23<00:00, 215.37it/s]

5000
300
```

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

In [195]:
```
# Similarly you can vectorize for title also

tfidf_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is st
for sentence in tqdm(preprocessed_titles[0:5000]): # for each review/senten
    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/re
    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 t
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.sp
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles.append(vector)

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

```
100%|██████████| 5000/5000 [00:00<00:00, 7899.68it/s]

5000
300
```

1.4.3 Vectorizing Numerical features

In [214]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generate
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
# 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

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.re
```

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

In [224]:
```
price_standardized = price_standardized[0:5000]
price_standardized
```

Out[224]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.49749975],
       [-0.34707649],
       [-0.70245417]])
```

In [223]:
```

previous_scalar = StandardScaler()
previous_scalar.fit(project_data['teacher_number_of_previously_posted_proje

# finding the mean and standard deviation of this data
print(f"Mean : {previous_scalar.mean_[0]}, Standard deviation : {np.sqrt(pr

# Now standardize the data with above maen and variance.
previous_standardized = previous_scalar.transform(project_data['teacher_num
previous_standardized =previous_standardized[0:5000]
```

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

1.4.4 Merging all the above features

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

In [225]:
```
print(state_one_hot.shape)

print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(tp_one_hot.shape)
print(pg_one_hot.shape)



#title
print(title_bow.shape)
print(title_tfidf.shape)
print(len(avg_w2v_vectors_titles))
print(len(tfidf_w2v_vectors_titles))



print(price_standardized.shape)
print(previous_standardized.shape)
```

```
(5000, 51)
(5000, 9)
(5000, 30)
(5000, 6)
(5000, 51)
(5000, 3329)
(5000, 3329)
5000
5000
(5000, 1)
(5000, 1)
```

In [228]:
```
# 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
X = hstack((state_one_hot,categories_one_hot, sub_categories_one_hot, tp_on
            avg_w2v_vectors_titles, tfidf_w2v_vectors_titles, price_standar
X.shape
```

Out[228]: (5000, 7407)

Assignment 2: Apply TSNE

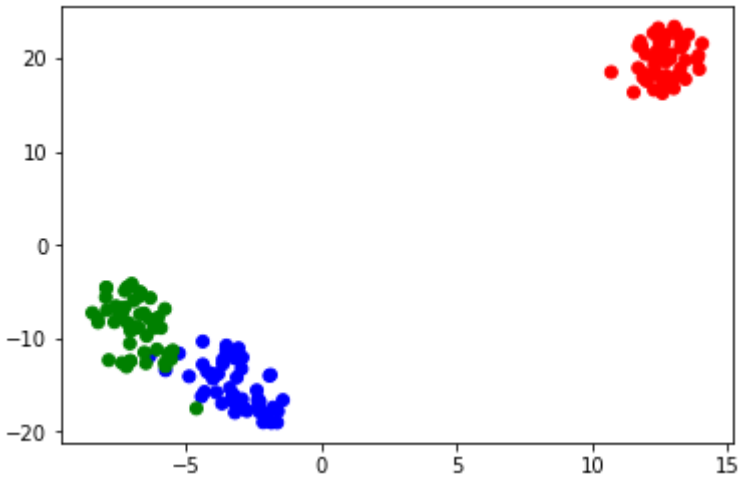
If you are using any code snippet from the internet, you have to provide the reference/citations, as we did in the above cells. Otherwise, it will be treated as plagiarism without citations.

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 [257]:
```
# this is the example code for TSNE
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

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

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_tra

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

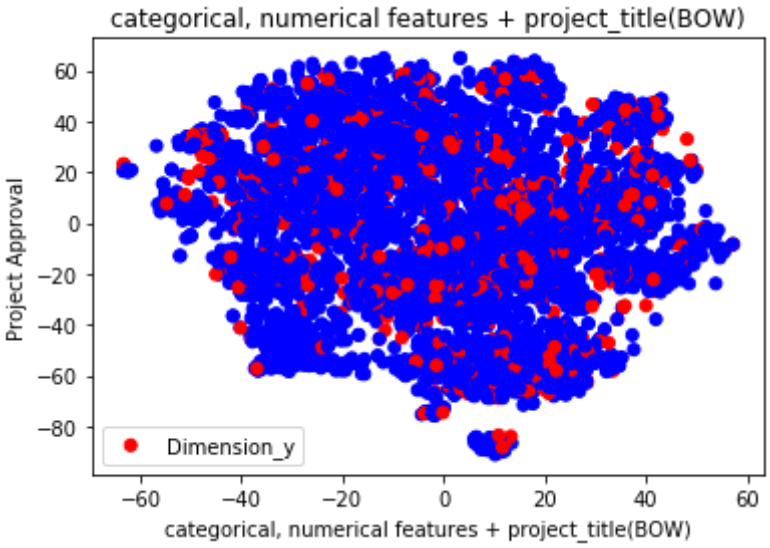


In [258]:
```
type(y)
```

Out[258]:
```
numpy.ndarray
```

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

In [261]:
```
# please write all of the code with proper documentation and proper titles
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

x = hstack((state_one_hot,categories_one_hot, sub_categories_one_hot, tp_on
            price_standardized, previous_standardized))


y = project_data['project_is_approved'][0:5000]

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

X_embedding = tsne.fit_transform(x.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_tra


for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_t

plt.title('categorical, numerical features + project_title(BOW)')
plt.xlabel('categorical, numerical features + project_title(BOW)')
plt.ylabel('Project Approval')

plt.legend()
plt.show()
```

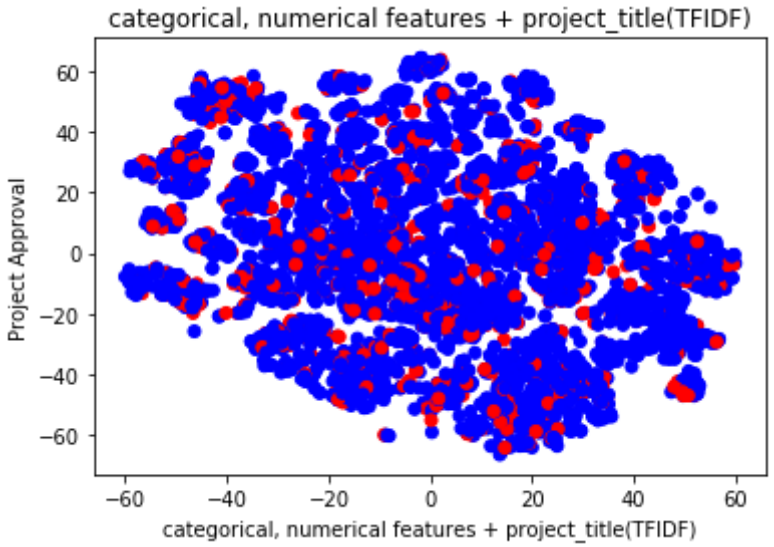


In [254]:
```
y = project_data['project_is_approved'][0:5000]
type(y)
```

Out[254]:
```
pandas.core.series.Series
```

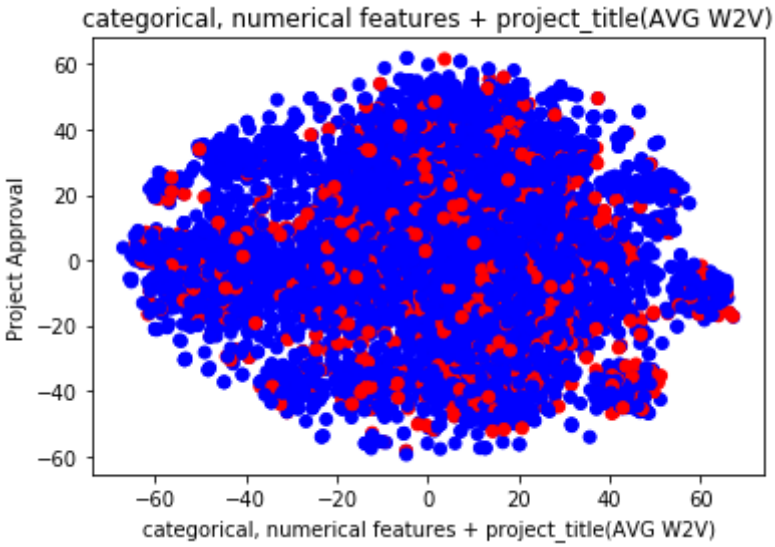
2.2 TSNE with `TFIDF` encoding of `project_title` feature

In [267]:
```
# please write all the code with proper documentation, and proper titles fo
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label



x = hstack((state_one_hot,categories_one_hot, sub_categories_one_hot, tp_on
            price_standardized, previous_standardized))


y = project_data['project_is_approved'][0:5000]

tsne = TSNE(n_components=2, perplexity=50, learning_rate=200)

X_embedding = tsne.fit_transform(x.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_tra


for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_t

plt.title('categorical, numerical features + project_title(TFIDF)')
plt.xlabel('categorical, numerical features + project_title(TFIDF)')
plt.ylabel('Project Approval')

plt.show()
```



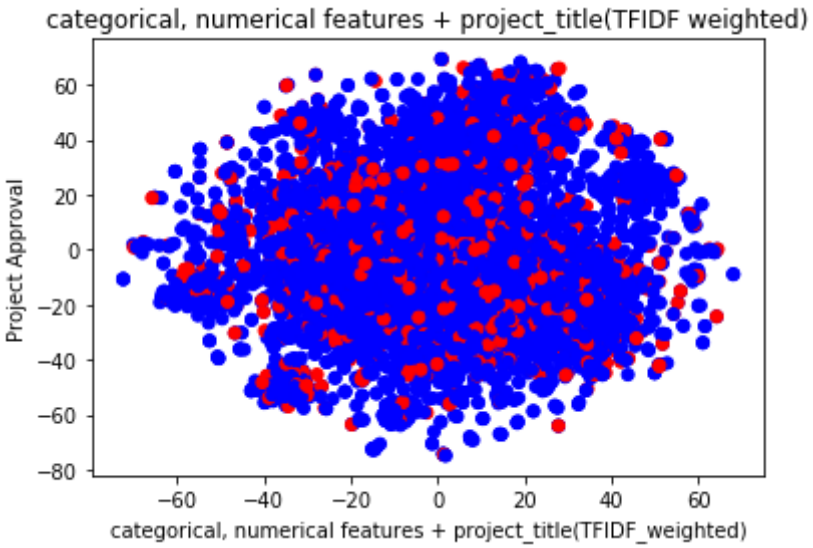
2.3 TSNE with `AVG W2V` encoding of `project_title` feature

In [264]:
```
# please write all the code with proper documentation, and proper titles fo
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label



x = hstack((state_one_hot,categories_one_hot, sub_categories_one_hot, tp_on
            price_standardized, previous_standardized))


y = project_data['project_is_approved'][0:5000]

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

X_embedding = tsne.fit_transform(x.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_tra


for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_t

plt.title('categorical, numerical features + project_title(AVG W2V)')
plt.xlabel('categorical, numerical features + project_title(AVG W2V)')
plt.ylabel('Project Approval')

plt.show()
```



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

In [266]:
```
# please write all the code with proper documentation, and proper titles fo
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label


x = hstack((state_one_hot,categories_one_hot, sub_categories_one_hot, tp_on
            price_standardized, previous_standardized))


y = project_data['project_is_approved'][0:5000]

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

X_embedding = tsne.fit_transform(x.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_tra


for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_t

plt.title('categorical, numerical features + project_title(TFIDF weighted)'
plt.xlabel('categorical, numerical features + project_title(TFIDF_weighted)
plt.ylabel('Project Approval')

plt.show()
```



2.5 Summary

In [0]:
```
# Write few sentences about the results that you obtained and the observati

## There are no obvious clusterings of groups based on TSNE visualizations.
## Even only 5000 rows used in X and Y, it takes very long time to plot eac
```