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`
	One or more (comma-separated) subject subcategories for the project. **Examples:**

Feature	Description
	• `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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 [2]:

```
%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

warnings.filterwarnings("ignore")
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 [3]:

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

In [6]:

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

Out[6]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [7]:

```
print("Number of data points in train data", project_data.shape)
print('-'*100)
print("The attributes of data :", project_data.columns.values)
```

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

In [8]:

```
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[8]:

	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

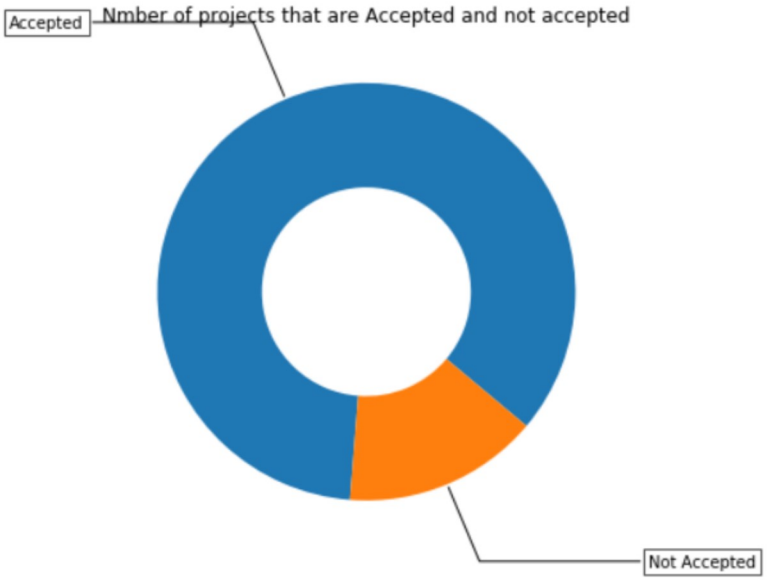
1.2 Data Analysis

In [9]:

```
# 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-pie-and-polar-charts-pie-and-donut-labels-py


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

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

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

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

```
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                horizontalalignment=horizontalalignment, **kw)

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

plt.show()
```

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



Observation(s) :

1. Majority of the projects that are submitted by the teachers are approved by DonorsChoose

1.2.1 Univariate Analysis: School State

In [4]:

```
# 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(10))
# 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',
```

```
            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')
```

Observation(s):

1. Delaware state has the highest percentage of approvals for the project.
2. Vermont state has the lowest percentage of apporovals for the project.

In [11]:

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

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

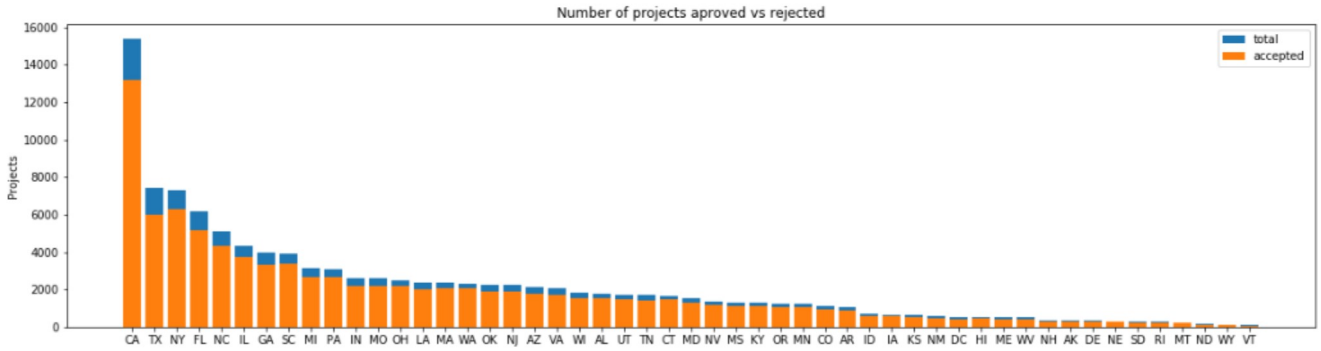
```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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 [13]:

```
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 [14]:

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



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

==========

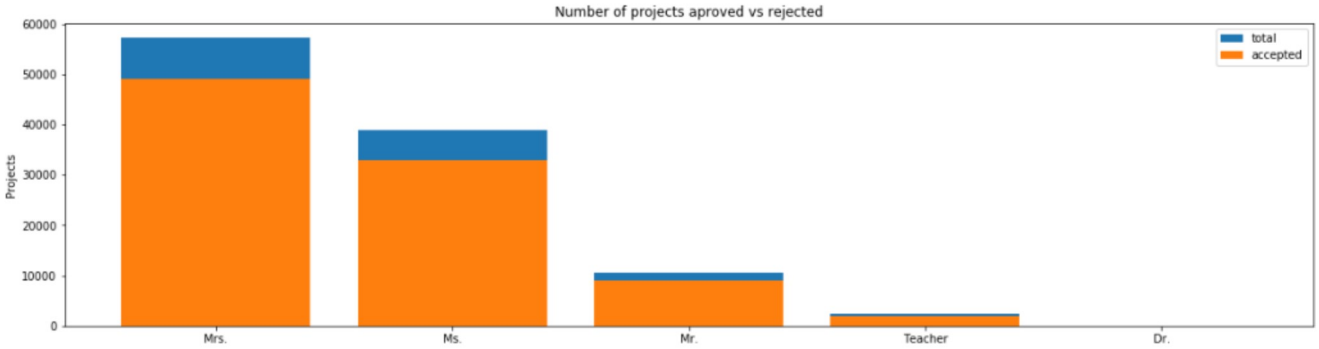
	school_state	project_is_approved	total	Avg
39	RI	243	285	0.852632
26	MT	200	245	0.816327
28	ND	127	143	0.888112
50	WY	82	98	0.836735
46	VT	64	80	0.800000

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

1.2.2 Univariate Analysis: teacher_prefix

In [15]:

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




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

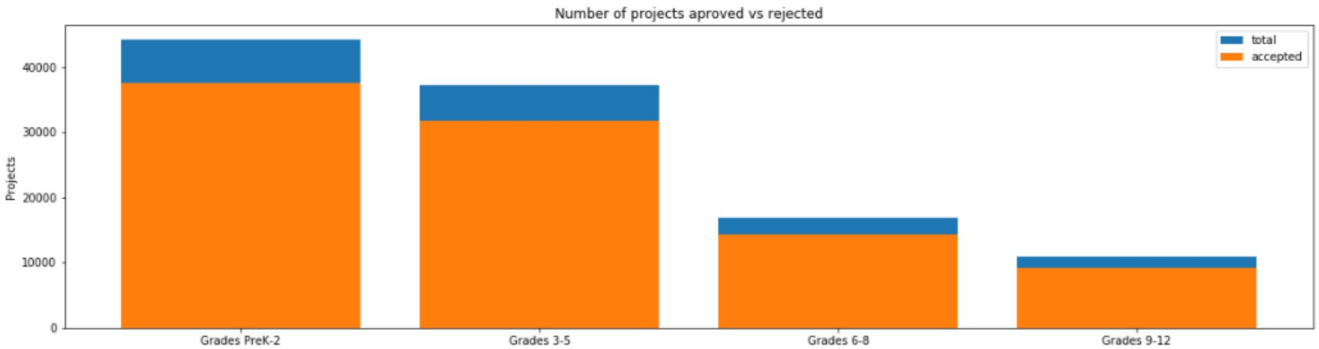
Observation(s):

1. Project approval rate is more for teacher's with prefix Mrs. and Ms.

1.2.3 Univariate Analysis: project_grade_category

In [16]:

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




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

```
1              Grades 6-8                 14258  16923  0.842522
2             Grades 9-12                  9183  10963  0.837636
```

Observation(s):

1. Approval rate is high for the projects which are are aimed for lower grades students

1.2.4 Univariate Analysis: project_subject_categories

In [17]:

```
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 [18]:

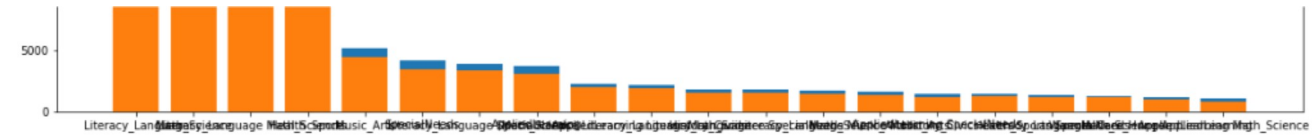
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[18]:

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

In [19]:

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



	clean_categories	project_is_approved	total	Avg
24	Literacy_Language	20520	23655	0.867470
32	Math_Science	13991	17072	0.819529
28	Literacy_Language Math_Science	12725	14636	0.869432
8	Health_Sports	8640	10177	0.848973
40	Music_Arts	4429	5180	0.855019

==========

	clean_categories	project_is_approved	total	Avg
19	History_Civics Literacy_Language	1271	1421	0.894441
14	Health_Sports SpecialNeeds	1215	1391	0.873472
50	Warmth Care_Hunger	1212	1309	0.925898
33	Math_Science AppliedLearning	1019	1220	0.835246
4	AppliedLearning Math_Science	855	1052	0.812738

Observation(s):

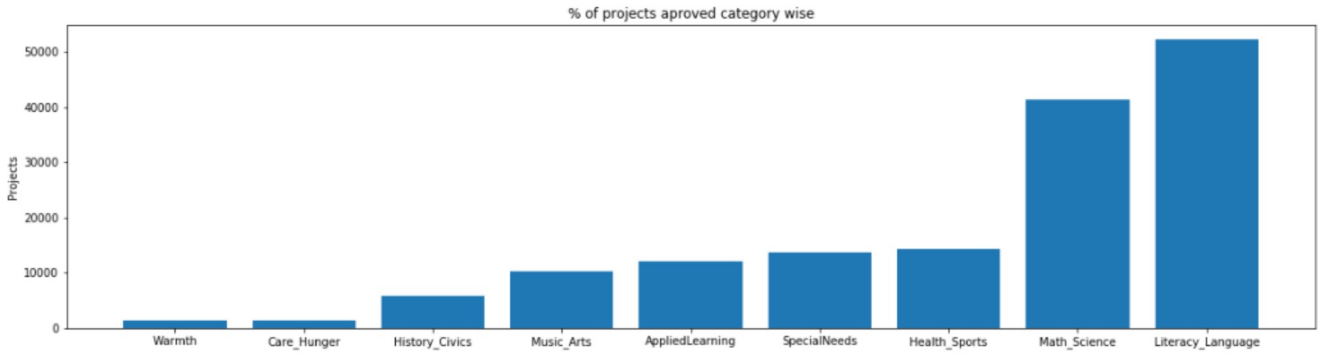
1. Approval rate is high for the projects which are related to Literacy and language or math and science

In [20]:

```
# 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 [21]:

```
# 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()
```



Obserevation(s):

1. Projects approval rate is high for the projects which are focussed on Literacy and language or Math and science

In [22]:

```
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 [23]:

```
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 [24]:

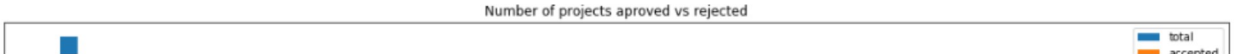
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

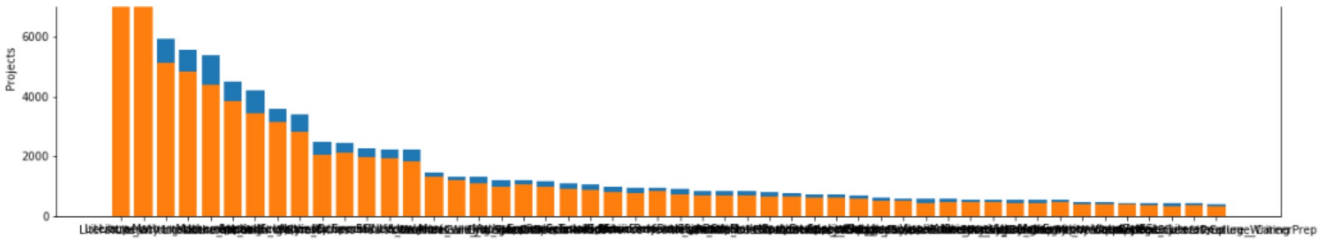
Out[24]:

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

In [25]:

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





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

==========

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

Observation(s):

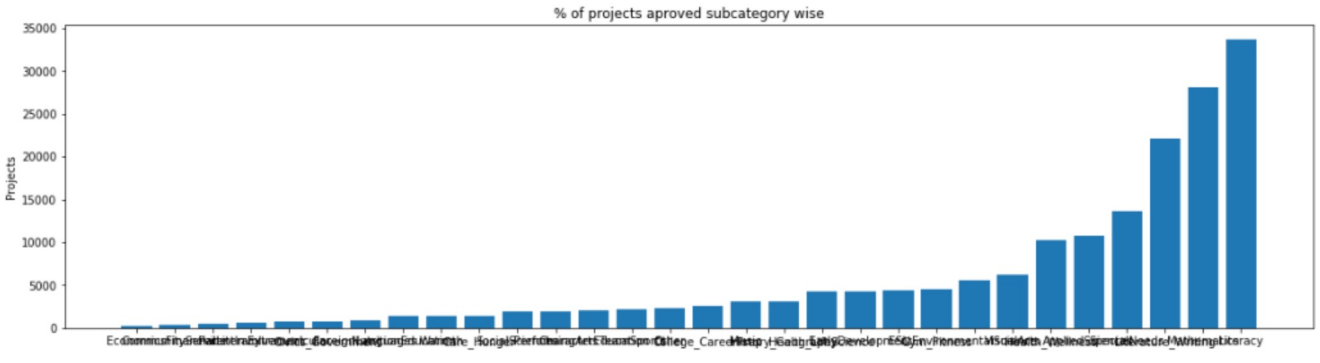
1. Project approval rate is high for the project if their subcategories falls in literacy or literacy including mathematics

In [26]:

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

In [27]:

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



Observation(s):

1. Project approval rate is high for the project if their subcategories falls in literacy or literacy including mathematics

In [28]:

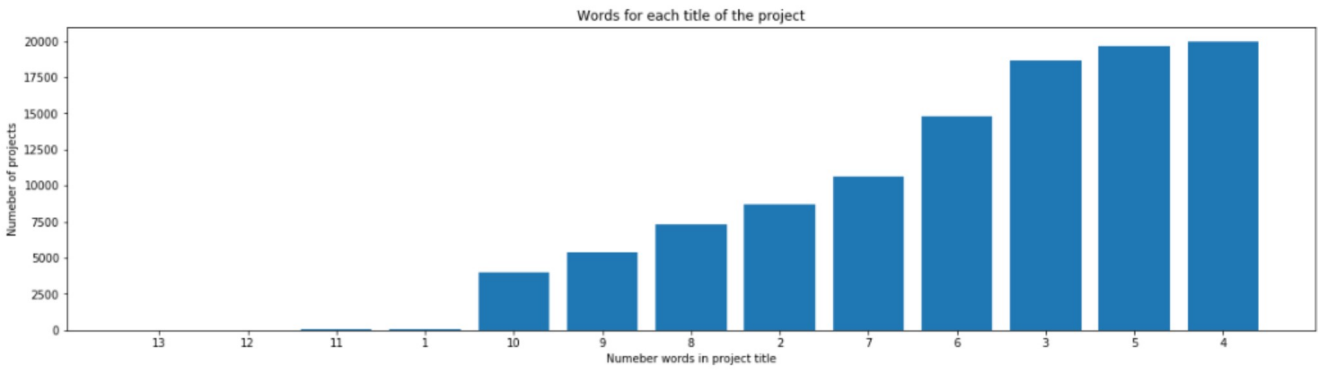
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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 [29]:

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


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

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



Observation(s):

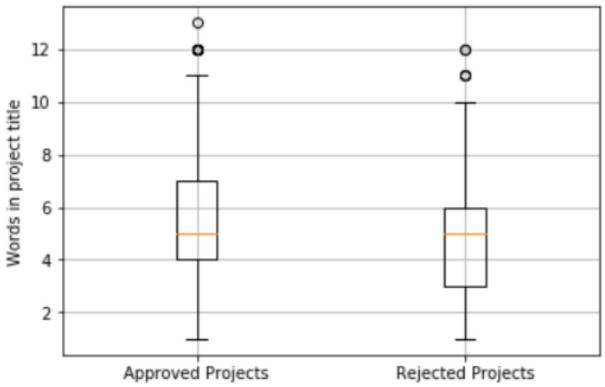
1. Most of the projects has 4 or 5 number of words in their title

In [30]:

```
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 [31]:

```
# 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()
```

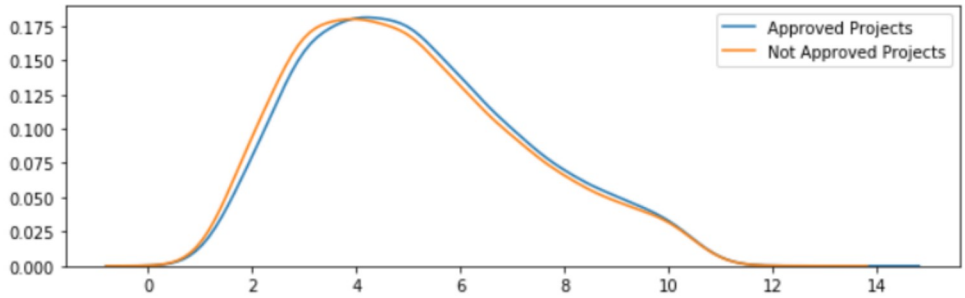


Observation(s):

1. No of words in the title of the project does not affect its approval in a significant way

In [32]:

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



Observation(s):

1. No of words in the title of the project does not affect its approval in a significant way

In [33]:

```
# 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 [34]:

```
project_data.tail(2)
```

Out[34]:

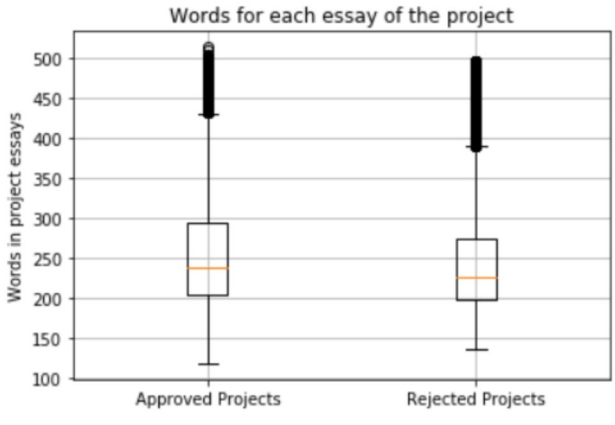
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
109246	164599	p206114	6d5675dbfafa1371f0e2f6f1b716fe2d	Mrs.	NY	2016-07-29 17:53:15
109247	128381	p191189	ca25d5573f2bd2660f7850a886395927	Ms.	VA	2016-06-29 09:17:01

In [35]:

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

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

In [36]:

```
# 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()
```

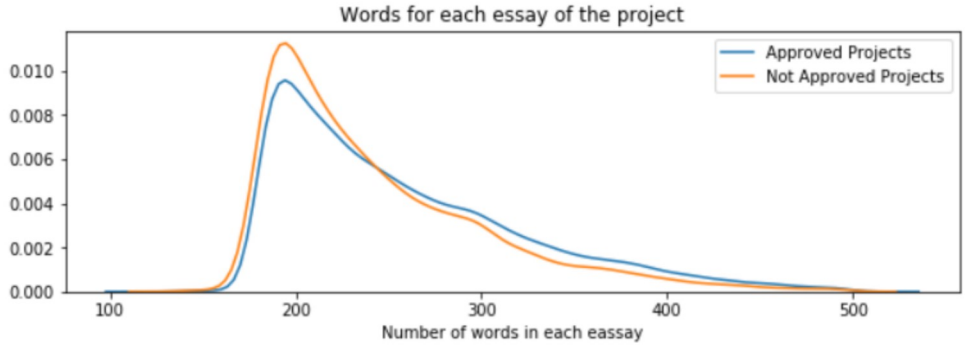


Observation(s):

1. Approved projects has slightly more number of words in their essays

In [37]:

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



Observation(s):

1. Approved projects has slightly more number of words in their essays

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 [39]:

```
# 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()
print(type(price_data))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [40]:

```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
project_data.head(2)
```

Out[40]:

	Unnamed:	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro

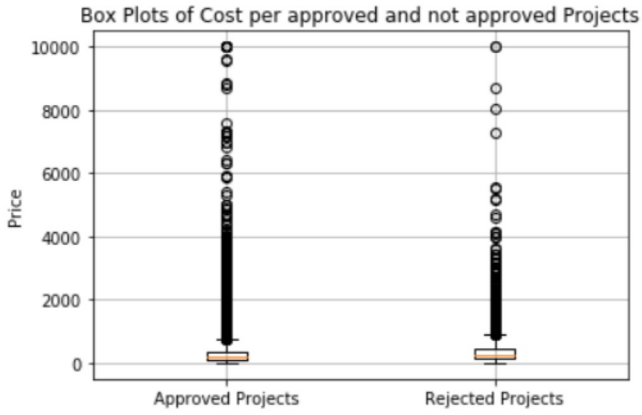
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [41]:

```
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 [42]:

```
# 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()
```

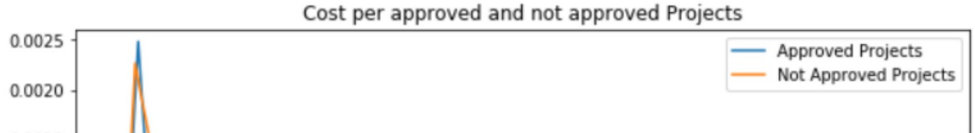


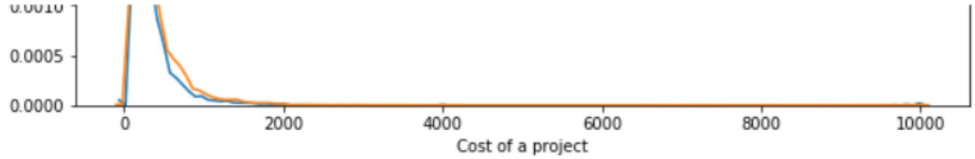
Observation(s):

1. Above graph depicts that approval for projects is almost similar irrespective of their cost and no bigger difference is seen.

In [43]:

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





Observation(s) :

1. Above graph depicts that approval for projects is almost similar irrespective of their cost and no bigger difference is seen.

In [44]:

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

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

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

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

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

Observation(s):

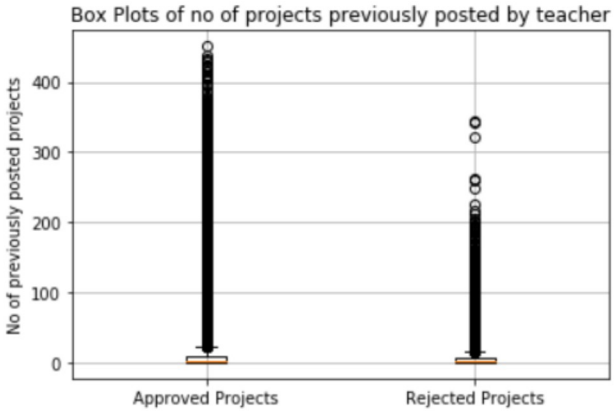
1. Approval rate of lower cost projects are higher.

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [45]:

```
approved_project = project_data[project_data['project_is_approved']==1]
['teacher_number_of_previously_posted_projects'].values

rejected_project = project_data[project_data['project_is_approved']==0]
['teacher_number_of_previously_posted_projects'].values

plt.boxplot([approved_project, rejected_project])
plt.title('Box Plots of no of projects previously posted by teacher')
```

```
plt.grid()
plt.show()
```

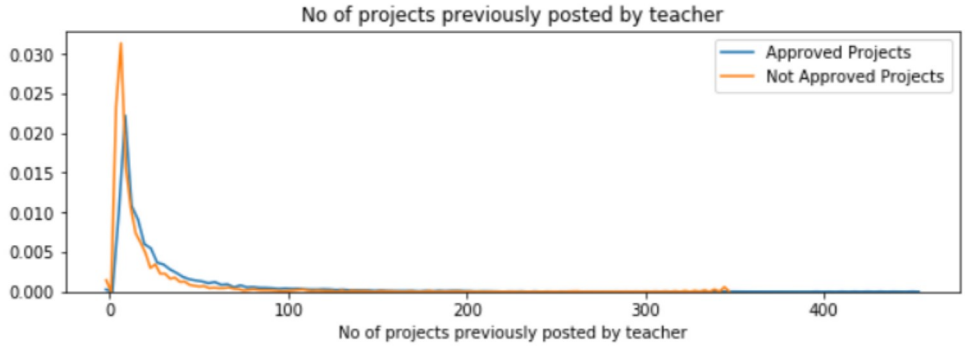


Observation(s) :

1. Above graph depicts that as the no of previously posted projects by teacher's increases, its approval rate also increases

In [46]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_project, hist=False, label="Approved Projects")
sns.distplot(rejected_project, hist=False, label="Not Approved Projects")
plt.title('No of projects previously posted by teacher')
plt.xlabel('No of projects previously posted by teacher')
plt.legend()
plt.show()
```



Observation(s) :

1. Above graph depicts that as the no of previously posted projects by teacher's increases, its approval rate also increases

In [47]:

```
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_project,i), 3), np.round(np.percentile(rejected_pr
oject,i), 3)])
print(x)
```

```
+------------+------------------+----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+------------------+----------------------+
|     0      |        0.0       |          0.0         |
|     5      |        0.0       |          0.0         |
|     10     |        0.0       |          0.0         |
|     15     |        0.0       |          0.0         |
```

```
|      30      |         1.0         |          0.0          |
|      35      |         1.0         |          1.0          |
|      40      |         1.0         |          1.0          |
|      45      |         2.0         |          1.0          |
|      50      |         2.0         |          2.0          |
|      55      |         3.0         |          2.0          |
|      60      |         4.0         |          3.0          |
|      65      |         5.0         |          3.0          |
|      70      |         7.0         |          4.0          |
|      75      |         9.0         |          6.0          |
|      80      |         13.0        |          8.0          |
|      85      |         19.0        |          11.0         |
|      90      |         30.0        |          17.0         |
|      95      |         57.0        |          31.0         |
|     100      |        451.0        |         345.0         |
+--------------+---------------------+-----------------------+
```

Observation(s) :

1. Project approval rate is slightly higher for the teachers who have posted more number of projects in the past.

1.2.10 Univariate Analysis: project_resource_summary

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

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

In [48]:

```
#check presence of digit in string python- https://stackoverflow.com/questions/19859282/check-if-a
-string-contains-a-number
def num_there(s):
    if(type(s)==int or type(s)==float):
        return True
    return any(i.isdigit() for i in s)
```

In [49]:

```
# No common columns to perform merge on. https://stackoverflow.com/questions/40468069/merge-two-da
taframes-by-index
sub_summary = list(project_data['project_resource_summary'].values)
project_data['Is_digit_present']=0
j=0
for i in sub_summary :
    a=num_there(i)
    if(a==True):
        project_data.loc[j,'Is_digit_present']=1
    j=j+1
project_data.tail(2)
```

Out[49]:

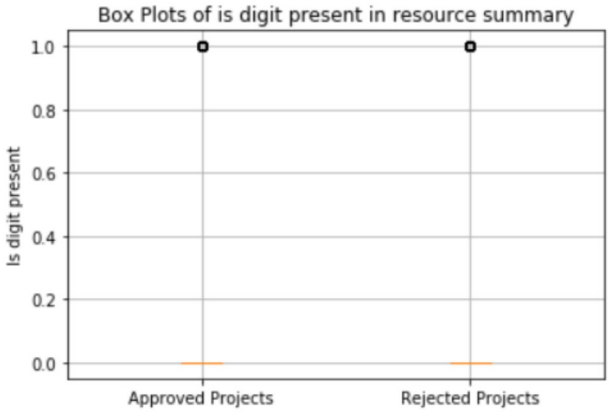
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
109246	164599	p206114	6d5675dbfafa1371f0e2f6f1b716fe2d	Mrs.	NY	2016-07-29 17:53:15
109247	128381	p191189	ca25d5573f2bd2660f7850a886305927	Ms.	VA	2016-06-29 09:17:01

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime

2 rows × 21 columns

In [50]:

```
approved_project = project_data[project_data['project_is_approved']==1]['Is_digit_present'].values

rejected_project = project_data[project_data['project_is_approved']==0]['Is_digit_present'].values

plt.boxplot([approved_project, rejected_project])
plt.title('Box Plots of is digit present in resource summary')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Is digit present')
plt.grid()
plt.show()
```

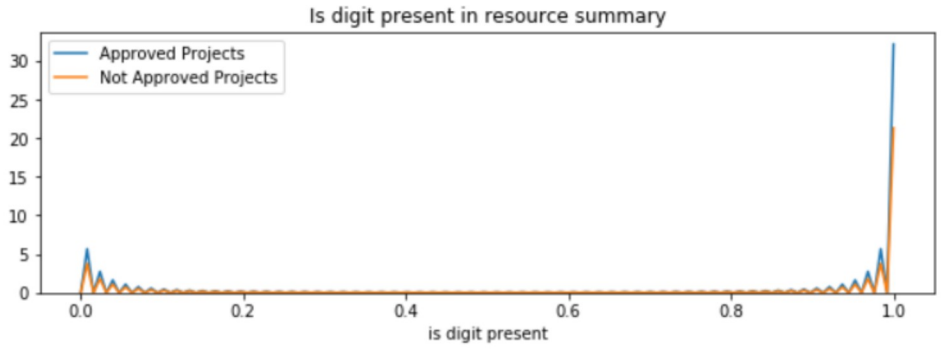


Observation(s) :

1. From the above graph we cant state anything regarding the data as the graph is exactly similar.

In [51]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_project, hist=False, label="Approved Projects")
sns.distplot(rejected_project, hist=False, label="Not Approved Projects")
plt.title('Is digit present in resource summary')
plt.xlabel('is digit present')
plt.legend()
plt.show()
```



Observation(s) :

1. Above graph clearly shows that if digits are present in the resource summary then its approval rate is higher.

1.2 Text preprocessing

1.3.1 Essay Text

In [52]:

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

```
My students are English learners that are working on English as their second or third languages. 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
```

```
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
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir 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 pr
ice 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 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 chairs are the answer and I love then because they dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids don't want to sit and do worksheets. They want to learn to count by ju
mping and playing. Physical engagement is the key to our success. The number toss and color and sh
ape mats can make that happen. My students will forget they are doing work and just have the fun a
6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The grea
t teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% Af
rican-American, making up the largest segment of the student body. A typical school in Dallas is m
ade up of 23.2% African-American students. Most of the students are on free or reduced lunch. We a
ren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As
an educator I am inspiring minds of young children and we focus not only on academics but one smar
t, effective, efficient, and disciplined students with good character.In our classroom we can util
ize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the so
und enough to receive the message. Due to the volume of my speaker my students can't hear videos o
r books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my s
tudents will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will all
ow me to have more room for storage of things that are needed for the day and has an extra part to
it I can use.  The table top chart has all of the letter, words and pictures for students to learn
about different letters and it is more accessible.nannan
==================================================
```

In [53]:

```
# 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 [54]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
from nltk.tokenize import word_tokenize
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', \
```

```
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [55]:

```
# 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)
    sent=sent.lower()
    word_tokens = word_tokenize(sent)
    # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
    sent = ' '.join(w for w in word_tokens if not w in stopwords)
    #sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent)
```

```
100%|██████████████████████████████████████████████████████████████| 109248/109248
[04:44<00:00, 384.40it/s]
```

In [56]:

```
# after preprocesing
preprocessed_essays[20001]
```

Out[56]:

```
'wonderful students 3 4 5 years old located small town outside charlotte nc 22 students children s
chool district employees students bright energetic love learn love hands activities get moving lik
e preschoolers enjoy music creating different things students come wonderful families supportive c
lassroom parents enjoy watching children growth much materials help teach students life cycle butt
erfly watch painted lady caterpillars grow bigger build chrysalis weeks emerge chrysalis beautiful
butterflies already net chrysalises still need caterpillars feeding station unforgettable
experience students student absolutely love hands materials learn much getting touch manipulate
different things supporting materials selected help students understand life cycle exploration nan
nan'
```

1.3.2 Project title Text

In [57]:

```
# similarly you can preprocess the titles also
```

In [58]:

```
preprocessed_title=[]
# 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', ' ')
```

```
    word_tokens = word_tokenize(sent)
    # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
    sent = ' '.join(w for w in word_tokens if not w in stopwords)
    #sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_title.append(sent)
```

```
100%|██████████████████████████████████████████████████████████████| 109248/109248
[00:20<00:00, 5225.86it/s]
```

In [59]:

```
preprocessed_title[20001]
```

Out[59]:

```
'beautiful life butterfly'
```

1. 4 Preparing data for models

In [60]:

```
project_data.columns
```

Out[60]:

```
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',
       'Is_digit_present'],
      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 [61]:

```
# we use count vectorizer to convert the values into one hot encoded features
my_counter_categories= Counter()
for word in project_data['clean_categories'].values:
    my_counter_categories.update(word.split())
cat_dict_categories = dict(my_counter_categories)
sorted_cat_dict = dict(sorted(cat_dict_categories.items(), key=lambda kv: kv[1]))
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
```

```
print(vectorizer.get_feature_names())


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

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

In [62]:

```
# we use count vectorizer to convert the values into one hot encoded features
my_counter_sub_categories= Counter()
for word in project_data['clean_subcategories'].values:
    my_counter_sub_categories.update(word.split())
cat_dict_categories = dict(my_counter_sub_categories)
sorted_cat_dict = dict(sorted(cat_dict_categories.items(), key=lambda kv: kv[1]))
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
)
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', '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  (109248, 30)
```

In [63]:

```
# we use count vectorizer to convert the values into one hot encoded features
my_counter_state=Counter()
for word in project_data['school_state'].values:
    my_counter_state.update(word.split())
cat_dict = dict(my_counter_state)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), 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)
print("Shape of matrix after one hot encodig ",state_one_hot.shape)
```

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

In [64]:

```
#how to fill Nan values in a column- https://youtu.be/fCMrO_VzeL8
project_data['teacher_prefix'].fillna(value='Teacher', inplace=True)
```

In [65]:

```
#to confirm that their is no NaN values in the column teacher_prefix
project_data.columns[project_data.isna().any()].tolist()
```

Out[65]:

In [66]:

```
# we use count vectorizer to convert the values into one hot encoded features
my_counter_prefix=Counter()
for word in project_data['teacher_prefix'].values:
    my_counter_prefix.update(word.split())
cat_dict = dict(my_counter_prefix)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
)
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())


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

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

In [67]:

```
# we use count vectorizer to convert the values into one hot encoded features
my_counter_project_grade=Counter()
for word in project_data['project_grade_category'].values:
    my_counter_project_grade.update(word.split())
cat_dict = dict(my_counter_project_grade)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())


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

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

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [68]:

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

```
Shape of matrix after one hot encodig  (109248, 16510)
```

1.4.2.2 Bag of Words on `project_title`

In [69]:

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

In [70]:

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

1.4.2.3 TFIDF vectorizer

In [71]:

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

```
Shape of matrix after one hot encodig  (109248, 16510)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [72]:

```
# Similarly you can vectorize for title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf_title = vectorizer.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encodig ",text_tfidf_title.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3222)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [73]:

```
# 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
f=open("glove_vectors.txt")
with open('glove_vectors.txt', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [74]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in 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_essay.append(vector)

print(len(avg_w2v_vectors_essay))
print(len(avg_w2v_vectors_essay[0]))
```

```
109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

```
# Similarly you can vectorize for title also
avg_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in 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_vectors_title.append(vector)

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

```
109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [76]:

```
# 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 [77]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in 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_essay.append(vector)

print(len(tfidf_w2v_vectors_essay))
print(len(tfidf_w2v_vectors_essay[0]))
```

```
109248
300
```

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

In [78]:

```
# Similarly you can vectorize for title also
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
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 [79]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in 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_vectors_title.append(vector)

print(len(tfidf_w2v_vectors_title))
print(len(tfidf_w2v_vectors_title[1]))
```

```
109248
300
```

1.4.3 Vectorizing Numerical features

In [80]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
print('Price')
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")
print('\n')
# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))

quantity_scalar = StandardScaler()
quantity_scalar.fit(project_data['quantity'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
print('Quantity')
print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation :
{np.sqrt(quantity_scalar.var_[0])}")
print('\n')
# Now standardize the data with above maen and variance.
quantity_standardized = quantity_scalar.transform(project_data['quantity'].values.reshape(-1, 1))

posted_scalar = StandardScaler()
posted_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)
) # finding the mean and standard deviation of this data
print('No of previous projects posted by teacher')
print(f"Mean : {posted_scalar.mean_[0]}, Standard deviation : {np.sqrt(posted_scalar.var_[0])}")
print('\n')
# Now standardize the data with above maen and variance.
posted_standardized = posted_scalar.transform(project_data['quantity'].values.reshape(-1, 1))

digit_scalar = StandardScaler()
digit_scalar.fit(project_data['Is_digit_present'].values.reshape(-1,1)) # finding the mean and
standard deviation of this data
```

```
print('\n')
# Now standardize the data with above maen and variance.
digit_standardized = posted_scalar.transform(project_data['Is_digit_present'].values.reshape(-1, 1)
)
```

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


Quantity
Mean : 16.965610354422964, Standard deviation : 26.182821919093175


No of previous projects posted by teacher
Mean : 11.153165275336848, Standard deviation : 27.77702641477403


Is digit present in resource summary
Mean : 0.14422231985940245, Standard deviation : 0.35131501862826
```

1.4.4 Merging all the above features

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

In [81]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(grade_one_hot.shape)
print(prefix_one_hot.shape)
print(text_bow_title.shape)
print(text_bow_essay.shape)
print(price_standardized.shape)
print(digit_standardized.shape)
print(quantity_standardized.shape)
print(posted_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 5)
(109248, 5)
(109248, 3222)
(109248, 16510)
(109248, 1)
(109248, 1)
(109248, 1)
(109248, 1)
```

Assignment 2: Apply TSNE

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

In [86]:

```
from scipy.sparse import coo_matrix
from scipy.sparse import csr_matrix
from sklearn.manifold import TSNE
from scipy.sparse import hstack
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot,grade_one_hot, prefix_one_hot, text_bow_titl
e, price_standardized,digit_standardized,quantity_standardized,posted_standardized))
X=csr_matrix(X)
```

```
labels_5000.head(3)
```

Out[86]:

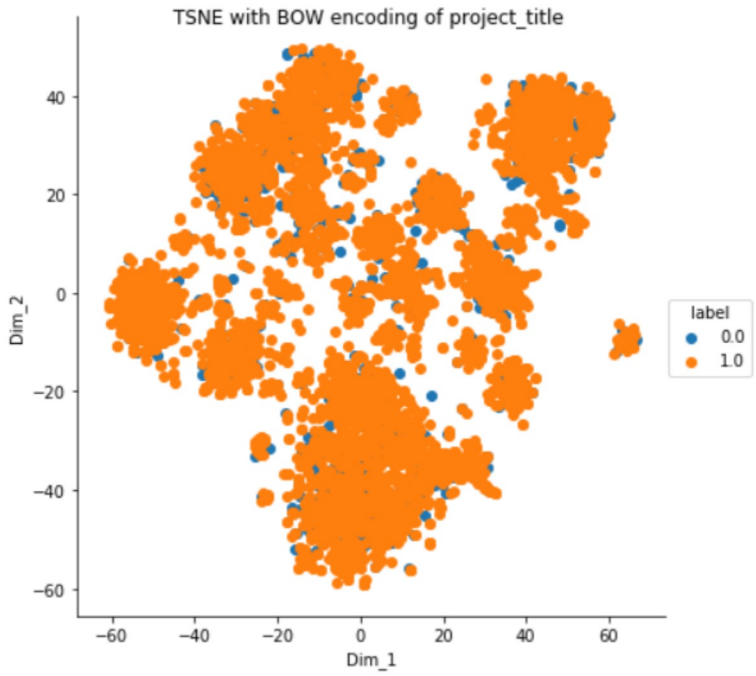
	project_is_approved
0	0
1	1
2	0

In [87]:

```
model = TSNE(n_components=2, random_state=0)

tsne_data = model.fit_transform(data_5000.toarray())
tsne_data = np.hstack((tsne_data, labels_5000))
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))
```

In [88]:

```
# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend() \
.fig.suptitle('TSNE with BOW encoding of project_title');
plt.show()
```



Observation(s) :

1. In the above t-SNE graph we can see some groupings among the labels, as the dataset is small (i.e. 5k), therefore the labels are mixed up.

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

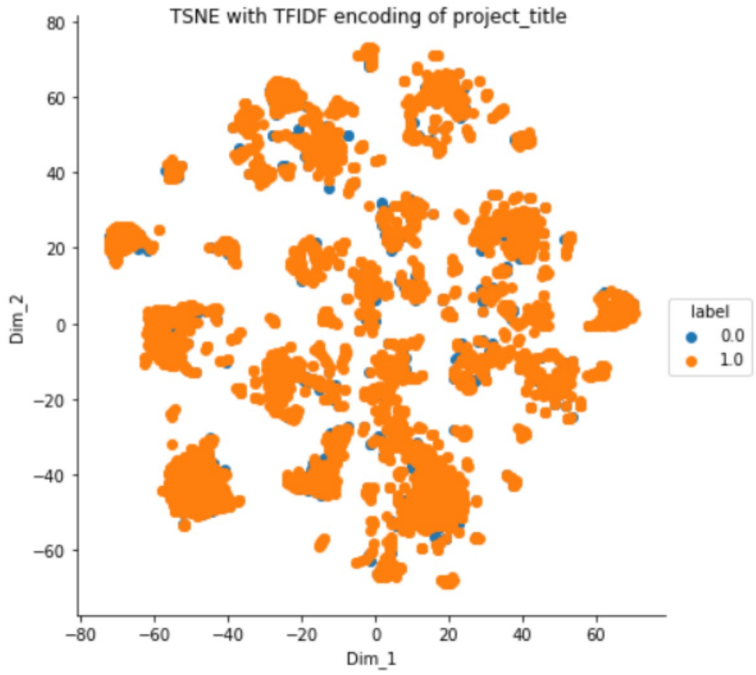
In [89]:

```
X = hstack((categories_one_hot, sub_categories_one_hot,grade_one_hot, prefix_one_hot, text_tfidf_ti
tle, price_standardized,digit_standardized,quantity_standardized,posted_standardized))
X=csr_matrix(X)

data_5000= X[0:5000,:]
```

```
tsne_data = model.fit_transform(data_5000.toarray())
tsne_data = np.hstack((tsne_data, labels_5000))
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))
```

In [90]:

```
# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend() \
.fig.suptitle('TSNE with TFIDF encoding of project_title');
plt.show()
```



Observation(s) :

1. In the above t-SNE graph we can see some groupings among the labels and the groupings are more scattered. As the dataset is small (i.e. 5k), therefore the labels are mixed up.

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

In [91]:

```
avg_w2v_vectors_title=csr_matrix(avg_w2v_vectors_title)
avg_w2v_vectors_title.shape
```

Out[91]:

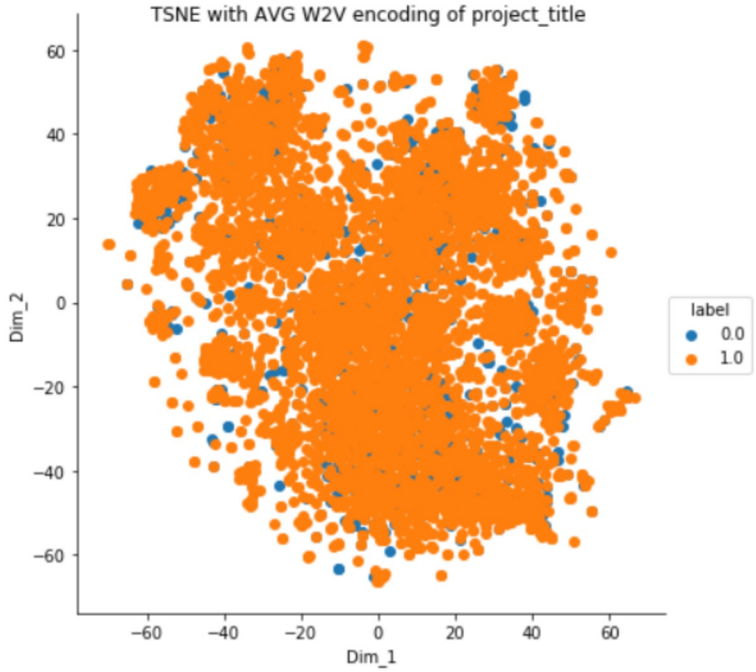
```
(109248, 300)
```

In [92]:

```
X = hstack((categories_one_hot, sub_categories_one_hot,grade_one_hot, prefix_one_hot, avg_w2v_vecto
rs_title, price_standardized,digit_standardized,quantity_standardized,posted_standardized))
X=csr_matrix(X)

data_5000= X[0:5000,:]
model = TSNE(n_components=2, random_state=0)

tsne_data = model.fit_transform(data_5000.toarray())
tsne_data = np.hstack((tsne_data, labels_5000))
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))
```

In [93]:

```
.fig.suptitle('TSNE with AVG W2V encoding of project_title');
plt.show()
```



Observation(s) :

1. In the above t-SNE graph we can see some groupings among the labels and the groupings are more dense together, as the dataset is small (i.e. 5k), therefore the labels are mixed up.

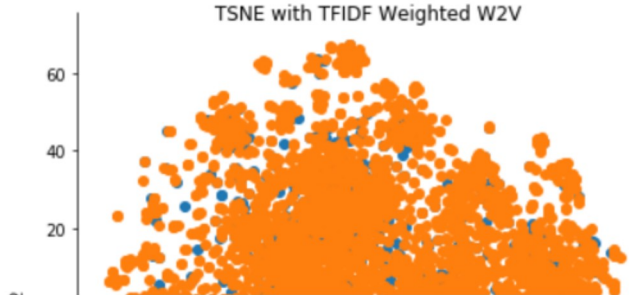
2.4 TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature

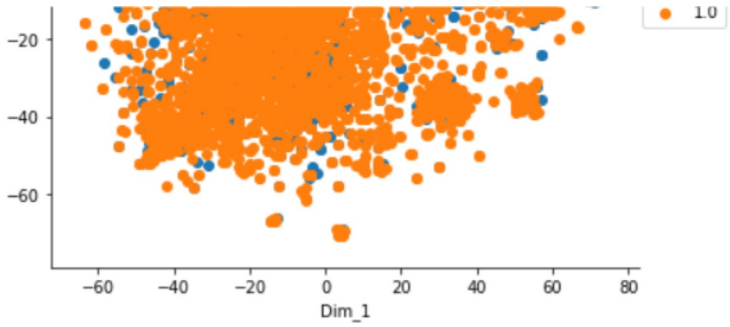
In [94]:

```
fidf_w2v_vectors_title=csr_matrix(avg_w2v_vectors_title)
X = hstack((categories_one_hot, sub_categories_one_hot,grade_one_hot, prefix_one_hot, tfidf_w2v_vec
tors_title, price_standardized,digit_standardized,quantity_standardized,posted_standardized))
X=csr_matrix(X)

data_5000= X[0:5000,:]
model = TSNE(n_components=2, random_state=0)

tsne_data = model.fit_transform(data_5000.toarray())
tsne_data = np.hstack((tsne_data, labels_5000))
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))
```

In [95]:

```
# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend() \
.fig.suptitle('TSNE with TFIDF Weighted W2V');
plt.show()
```





Observation(s) :

1. In the above t-SNE graph we can see some groupings among the labels and the scatterings of groups are more uniform, as the dataset is small (i.e. 5k), therefore the labels are mixed up.

2.5 Summary

Observation(s) :

1. Majority of the projects that are submitted by the teachers are approved by DonorsChoose
2. Delaware state has the highest percentage of approvals for the project. Vermont state has the lowest percentage of apporovals for the project.
3. Every state has greater than 80% success rate in approval
4. Project approval rate is more for teacher's with prefix Mrs. and Ms.
5. Approval rate is high for the projects which are are aimed for lower grades students
6. Approval rate is high for the projects which are related to Literacy and language or math and science
7. Approved projects has slightly more number of words in their essays
8. Approval rate of lower cost projects are higher.
9. Project approval rate is slightly higher for the teachers who have posted more number of projects in the past.
10. If digits are present in the resource summary then its approval rate is higher.