DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`

`school_state`	State where school is located ([Two-letter U.S. postal code](#)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** - `Literacy` - `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** - `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: - `nan` - `Dr.` - `Mr.` - `Mrs.` - `Ms.` - `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [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

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 chart_studio.plotly import plot, iplot
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 [4]:

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

```
Number of data points in train data (109248, 1
7)
----------------------------------------------
----
The attributes of data : ['Unnamed: 0' 'id' 't
eacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_c
ategory'
 'project_subject_categories' 'project_subject
_subcategories'
 'project_title' 'project_essay_1' 'project_es
say_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects
' 'project_is_approved']
```

In [5]:

```
print("Number of data points in train data", resource_data.sh
ape)
```

```
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[5]:

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

In [6]:

```
np.unique(project_data["project_grade_category"].values)
```

Out[6]:

```
array(['Grades 3-5', 'Grades 6-8', 'Grades 9-1
2', 'Grades PreK-2'], dtype=object)
```

In [7]:

```
# We need to get rid of The spaces between the text and the h
yphens because they're special characters.
#Rmoving multiple characters from a string in Python
#https://stackoverflow.com/questions/3411771/multiple-charact
er-replace-with-python

project_grade_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" "
, "_").replace("-", "_")
    project_grade_category.append(a)
```

In [8]:

```
project_data.drop(['project_grade_category'], axis = 1, inplace = True)
project_data["project_grade_category"] = project_grade_category
print("After removing the special characters ,Column values: ",np.unique(project_data["project_grade_category"].values))
```

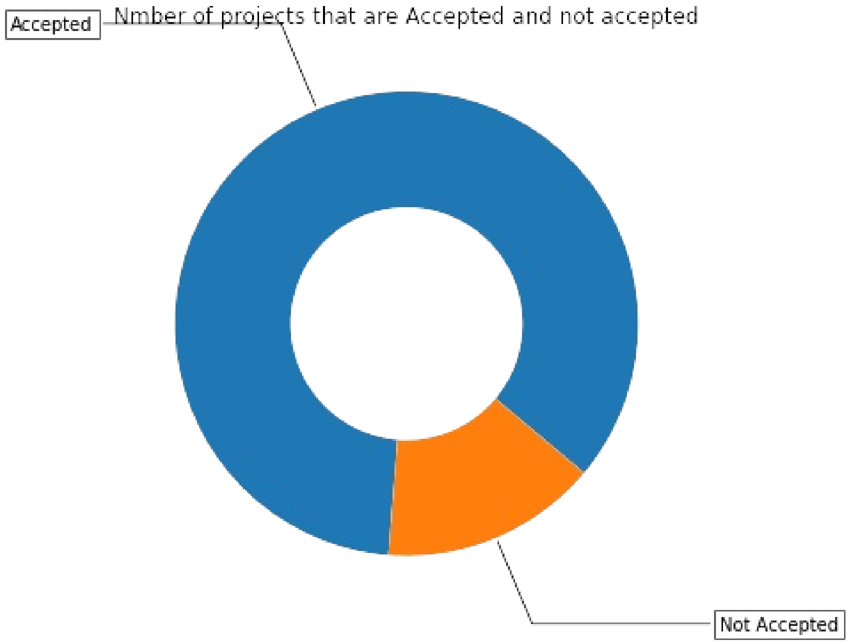
```
After removing the special characters ,Column values:  ['Grades_3_5' 'Grades_6_8' 'Grades_9_12' 'Grades_PreK_2']
```

1.2 Data Analysis

In [9]:

```
# 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))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                 horizontalalignment=horizontalalignment, **kw)

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

plt.show()
```

```
Number of projects thar are approved for funding  92706 , ( 84.8583040422 %)
Number of projects thar are not approved for funding  16542 , ( 15.1416959578 %)
```



From the pie chart,we can observe that out of 109k project proposals:

- 92706 proposals,nearly 85% of the total project proposals were approved for funding
- 16542 proposals , about 15% of the total project proposals were rejected

1.2.1 Univariate Analysis: School State

In [19]:

```
'''Reference == https://datascience.stackexchange.com/questions/9616/how-to-create-us-state-heatmap/9620#9620'''
# 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']



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'],
```

```python
        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='d3-cloropleth-map')
```

In [12]:

```python
# https://www.csi.cuny.edu/sites/default/files/pdf/administra
tion/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
```

Summary:

- **Among all states, it's evident that Vermont has the lowest acceptance rate of 80%**

In [13]:

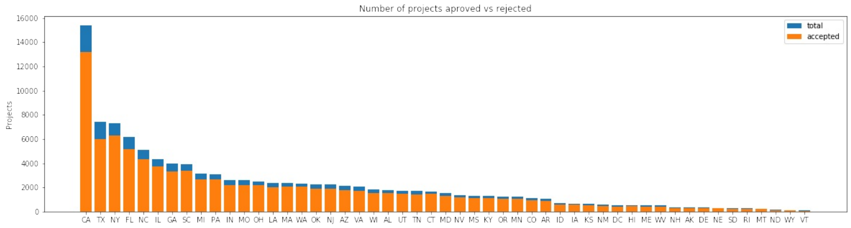
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery
/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3=
'total'):
    ind = np.arange(data.shape[0])

    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 [14]:

```
def univariate_barplots(data, col1, col2='project_is_approved
', top=False):
    # Count number of zeros in dataframe python: https://stac
koverflow.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.co
m/a/19385591/4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)[c
ol2].agg({'total':'count'})).reset_index()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col
2].agg({'Avg':'mean'})).reset_index()['Avg']

    temp.sort_values(by=['total'],inplace=True, ascending=Fal
se)

    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 [15]:

```
univariate_barplots(project_data, 'school_state', 'project_is
_approved', False)
```



```
   school_state  project_is_approved  total
    Avg
4            CA                13205  15388  0
.858136
43           TX                 6014   7396  0
.813142
34           NY                 6291   7318  0
.859661
9            FL                 5144   6185  0
.831690
27           NC                 4353   5091  0
```

```
.855038
==============================================
====
   school_state  project_is_approved  total
    Avg
39           RI                  243    285  0
.852632
26           MT                  200    245  0
.816327
28           ND                  127    143  0
.888112
50           WY                   82     98  0
.836735
46           VT                   64     80  0
.800000
```

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

However,the following observations can be made:

- California has the highest number of project proposals compared to other states

- Delaware has the highest acceptance rate of 89%

- Vermont has the least acceptance rate of 80%

In [18]:

```
#NaN values in techer prefix will create a problem while enco
ding,so we replace NaN values with the mode of that particula
```

```
r column

mode_of_teacher_prefix = project_data['teacher_prefix'].value
_counts().index[0]


project_data['teacher_prefix'] = project_data['teacher_prefix
'].fillna(mode_of_teacher_prefix)
project_data['teacher_prefix']
```

Out[18]:

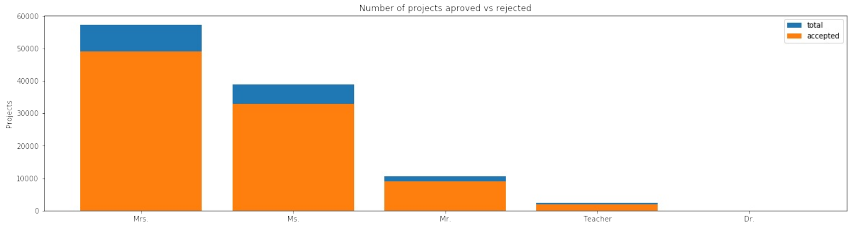
```
0             Mrs.
1              Mr.
2              Ms.
3             Mrs.
4             Mrs.
5             Mrs.
6             Mrs.
7              Ms.
8             Mrs.
9              Ms.
10            Mrs.
11             Ms.
12            Mrs.
13            Mrs.
14             Ms.
15             Ms.
16            Mrs.
17             Ms.
18            Mrs.
19             Ms.
20            Mrs.
21            Mrs.
22             Ms.
23             Mr.
24            Mrs.
25            Mrs.
```

```
26            Ms.
27        Teacher
28           Mrs.
29           Mrs.
           ...
109218       Mrs.
109219    Teacher
109220       Mrs.
109221    Teacher
109222        Ms.
109223        Ms.
109224        Ms.
109225       Mrs.
109226        Ms.
109227       Mrs.
109228       Mrs.
109229       Mrs.
109230        Ms.
109231       Mrs.
109232       Mrs.
109233        Ms.
109234        Ms.
109235       Mrs.
109236       Mrs.
109237       Mrs.
109238       Mrs.
109239       Mrs.
109240       Mrs.
109241       Mrs.
109242       Mrs.
109243        Mr.
109244        Ms.
109245       Mrs.
109246       Mrs.
109247        Ms.
Name: teacher_prefix, Length: 109248, dtype: o
bject
```

1.2.2 Univariate Analysis: teacher_prefix

In [20]:

```
univariate_barplots(project_data, 'teacher_prefix', 'project_
is_approved' , top=False)
```



```
  teacher_prefix  project_is_approved  total
     Avg
2           Mrs.                49000  57272
0.855566
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.                49000  57272
0.855566
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:

- **Female teachers have more number of projects proposed and also have a higher acceptance rate compared to the male teachers**

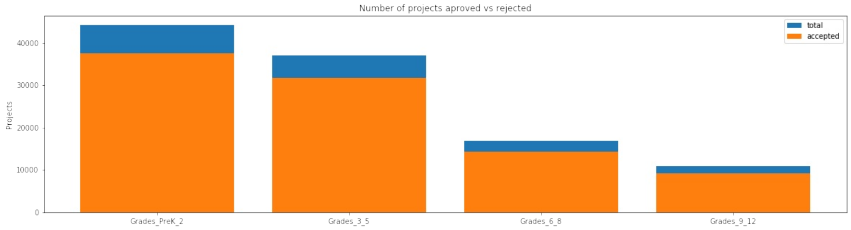
- **Among the female teachers,the ones with prefixes (Mrs.) that is married female teachers have more number of project proposals**

- **Teachers with Dr. prefix have the least number of project proposals and also the least acceptance rate compared to project proposals coming from teachers with other prefixes**

1.2.3 Univariate Analysis: project_grade_category

In [21]:

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



```
  project_grade_category  project_is_approved
 total       Avg
3          Grades_PreK_2                37536
 44225  0.848751
0             Grades_3_5                31729
 37137  0.854377
1             Grades_6_8                14258
 16923  0.842522
2            Grades_9_12                 9183
 10963  0.837636
==============================================
====
  project_grade_category  project_is_approved
 total       Avg
3          Grades_PreK_2                37536
 44225  0.848751
0             Grades_3_5                31729
 37137  0.854377
1             Grades_6_8                14258
 16923  0.842522
2            Grades_9_12                 9183
 10963  0.837636
```

1.2.4 Univariate Analysis: project_subject_categories

In [22]:

```
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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "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 [23]:

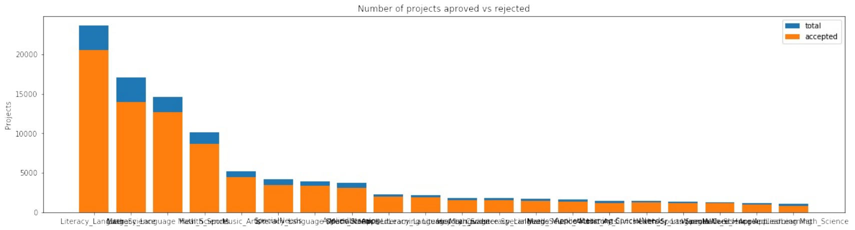
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[23]:

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

In [24]:

```
univariate_barplots(project_data, 'clean_categories', 'projec
t_is_approved', top=20)
```



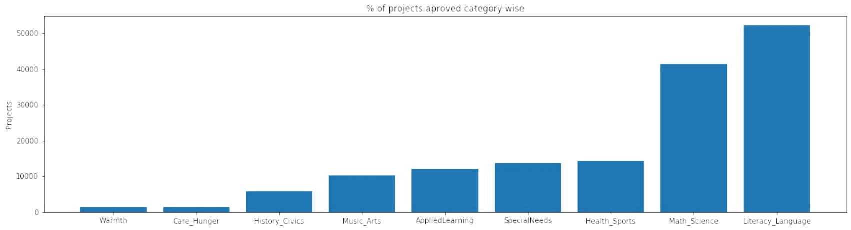
```
                  clean_categories  project_is
_approved  total       Avg
24                 Literacy_Language
    20520  23655  0.867470
32                      Math_Science
    13991  17072  0.819529
28  Literacy_Language Math_Science
    12725  14636  0.869432
8                      Health_Sports
     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
```

In [25]:

```
# 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 [26]:

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


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

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



In [27]:

```
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
```

SUMMARY :

- Projects belonging to the Literacy and Language categories have the highest number of projects proposals

- Warmth,care and Hunger have the least number of project proposals

1.2.5 Univariate Analysis: project_subject_subcategories

In [28]:

```
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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "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 [29]:

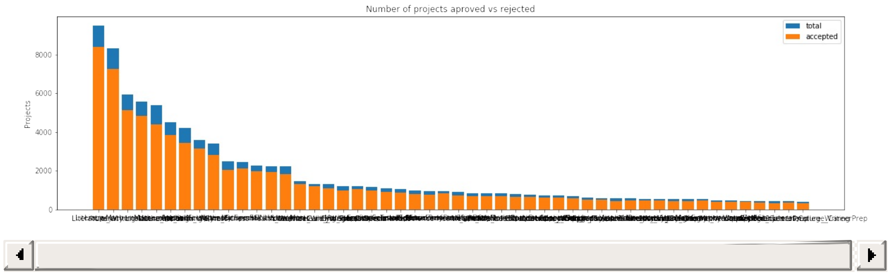
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[29]:

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

In [30]:

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



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

SUMMARY :

- From the bar plot, it can be observed that the number of project proposals and acceptance rate from each category is not uniform and there is a lot variability based on category

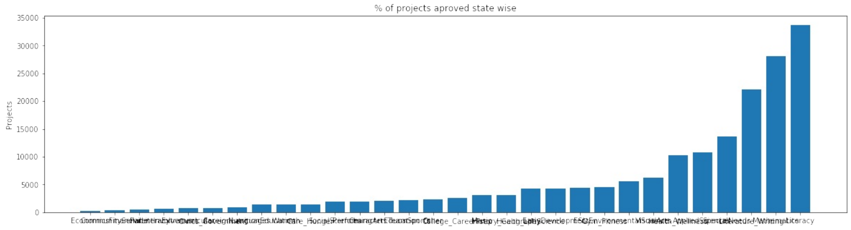
- Projects belonging to Literacy and Language categories have the highest acceptance rate of about 87-88% compared to other categories

In [31]:

```python
# 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 [32]:

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



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

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



In [33]:

```python
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
```

SUMMARY :

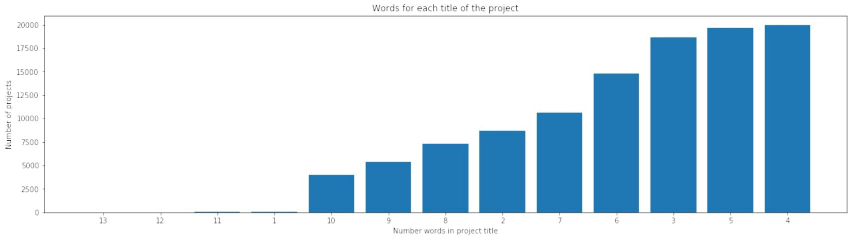
- Literacy has the highest number of project proposals

- Economics has the lowest number of projects proposals with 269 projects only

1.2.6 Univariate Analysis: Text features (Title)

In [35]:

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


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

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



SUMMARY :

- The word count in the title for Majority of the projects is between 3-5 words

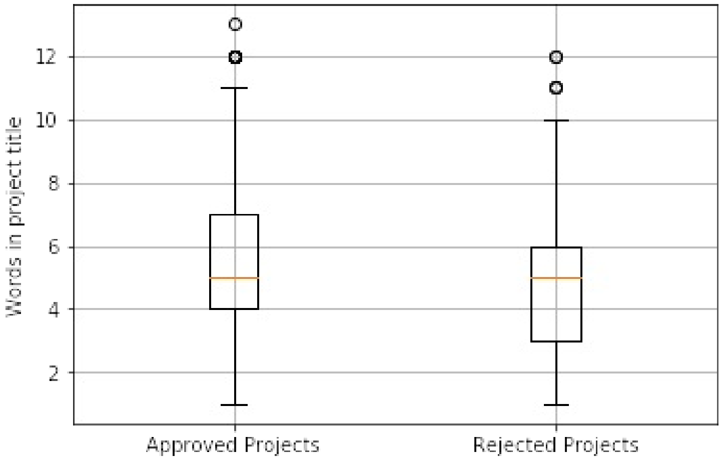
- Very few project titles have more than 10 words

In [36]:

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

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

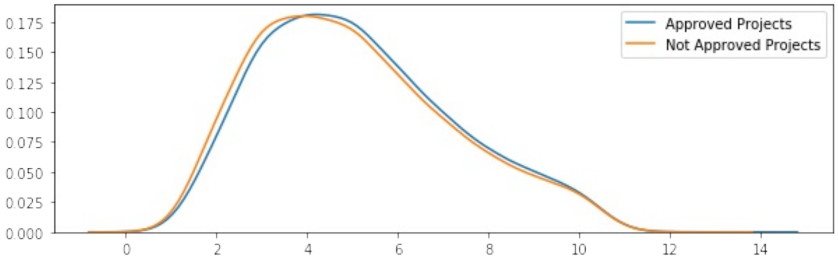
In [37]:

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



In [38]:

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



SUMMARY :

- From the kde plot and box plots,it's evident that there is some loose correlation between the word count in the project title and the approval status of the project

- Projects with more number of words in the title have a slightly better acceptance rate than projects with less number of words

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

In [39]:

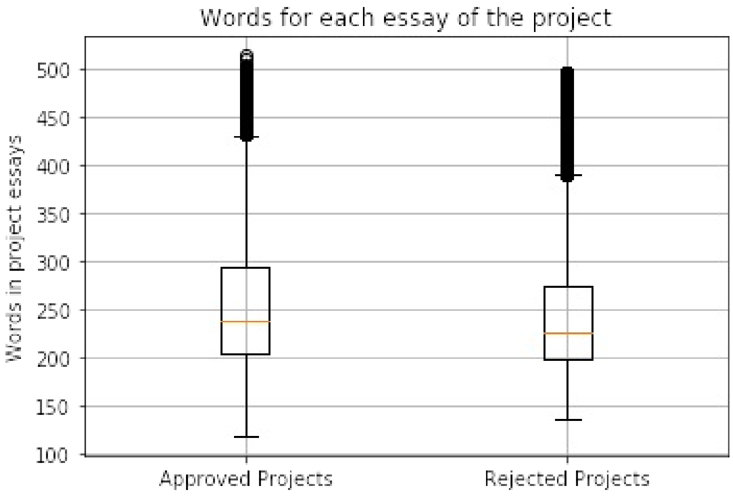
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [40]:

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

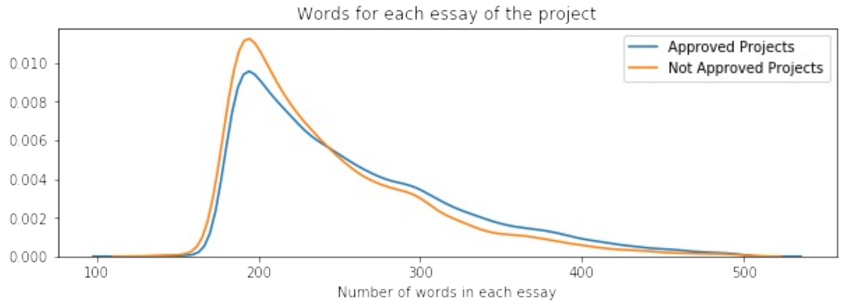
In [41]:

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



In [42]:

```python
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 essay')
plt.legend()
plt.show()
```



SUMMARY :

From the kde plot or the pdf,it's observable that the approved projects have slighly more number of words in their project essays as compared to rejected projects

1.2.8 Univariate Analysis: Cost per project

In [43]:

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

Out[43]:

	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 [44]:

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

Out[44]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [45]:

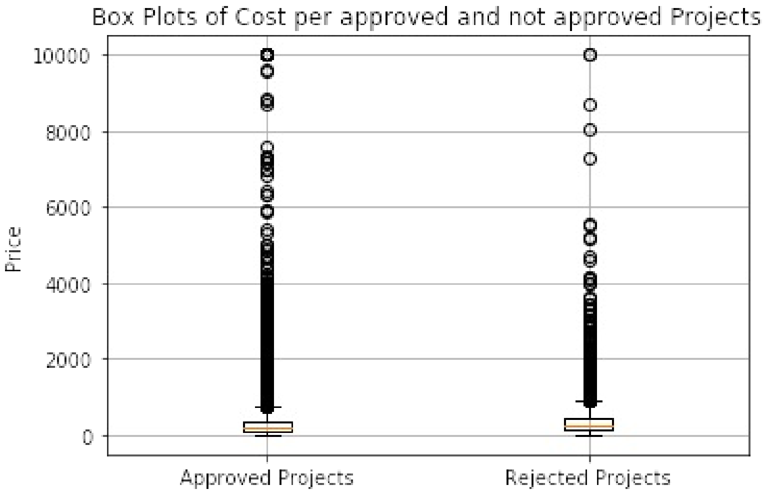
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', ho
w='left')
```

In [46]:

```
approved_price = project_data[project_data['project_is_approv
ed']==1]['price'].values

rejected_price = project_data[project_data['project_is_approv
ed']==0]['price'].values
```

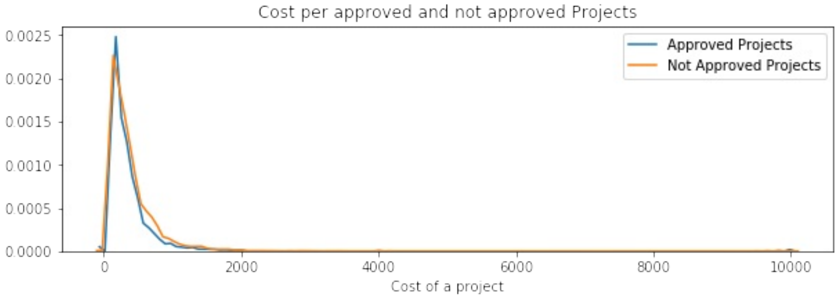
In [47]:

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



In [48]:

```
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 :

- The pdf's of approved projects and rejected projects based on the feature price seem to almost overlap

- However,in general if the price of the project proposal is more or if it's expensive,then it is likely to get rejected

In [49]:

```
# 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 Appr
oved 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_price,i), 3)])
print(x)
```

```
+------------+-------------------+-----------
-----------+
| Percentile | Approved Projects | Not Approve
d 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       |         999
9.0        |
+------------+--------------------+-----------
```

```
-----------+
```

SUMMARY :

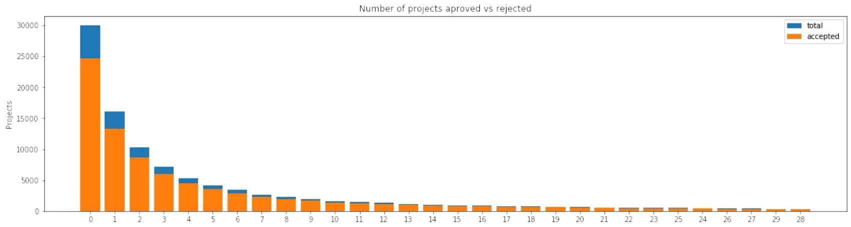
- The approved projects tend to have lower cost when compared to the projects that have not been approved. This is evident from the percentile values. The 50th percentile Cost value for an approved project is 199 dollars whereas the cost for rejected projects is 263 dollars.

- In general, if the price of any project exceeds more than 10,000 dollars,then it will be rejected.

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [50]:

```
univariate_barplots(project_data, 'teacher_number_of_previous
ly_posted_projects', 'project_is_approved',top=30)
```



```
   teacher_number_of_previously_posted_project
s  project_is_approved  total  \
0
0                    24652  30014
```

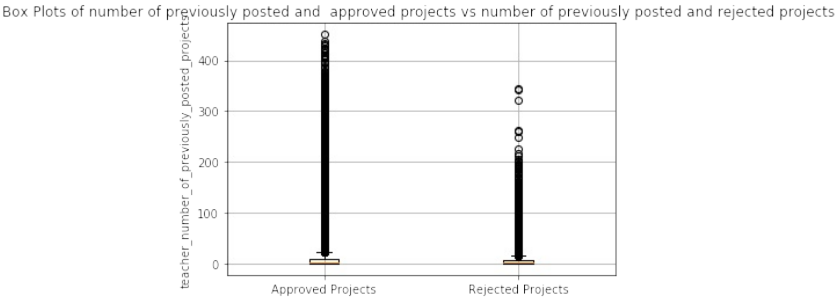
```
1
1                13329  16058
2
2                 8705  10350
3
3                 5997   7110
4
4                 4452   5266

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==============================================
====
    teacher_number_of_previously_posted_projec
ts  project_is_approved  total  \
24
24                  405    449
26
26                  378    445
27
27                  352    394
29
29                  336    370
28
28                  313    352

         Avg
24  0.902004
26  0.849438
27  0.893401
29  0.908108
28  0.889205
```

In [51]:

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

rejected_previously_posted_projects = project_data[project_data['project_is_approved']==0]['teacher_number_of_previously_posted_projects'].values
```

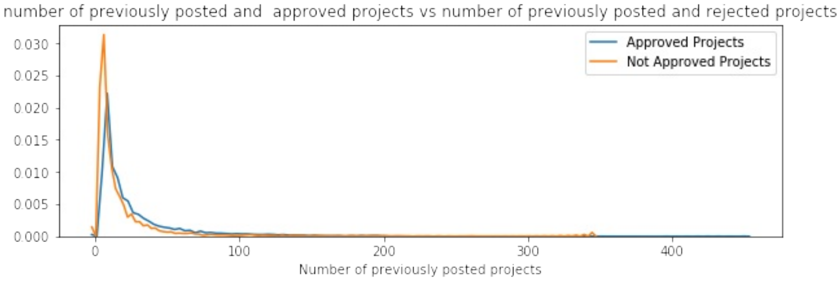
In [52]:

```
plt.boxplot([approved_previously_posted_projects, rejected_previously_posted_projects])
plt.title('Box Plots of number of previously posted and  approved projects vs number of previously posted and rejected projects ')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('teacher_number_of_previously_posted_projects')
plt.grid()
plt.show()
```



In [53]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_previously_posted_projects, hist=False, label="Approved Projects")
sns.distplot(rejected_previously_posted_projects, hist=False, label="Not Approved Projects")
plt.title('number of previously posted and  approved projects
 vs number of previously posted and rejected projects')
plt.xlabel('Number of previously posted projects')
plt.legend()
plt.show()
```



SUMMARY :

- We can observe that the approval status is not affected much by how many projects were submitted in the past. From the barplots, it is evident that majority of the approved project proposals have no previous submissions.

- But among the accepted project proposals with more number of previous submissions, the rate of approval seems to be Higher if the teacher has proposed atleast 19 different projects.

1.2.10 Univariate Analysis: project_resource_summary

In [54]:

```
#PRESENCE OF NUMERIC CHARACTERS IN PROJECT SUMMARY
```

```
#Some summaries contain alphanumeric characters but we don't
want them to be added in the new column.
#Hence,we only consider numeric digits present in each summar
y and store it in a dictionary

summaries = project_data['project_resource_summary']
digits_in_each_summary = {}

for x in tqdm(range(len(project_data))):
    for word in summaries[x].split():
        if word.isdigit() :
            digits_in_each_summary[x] = int(word)      #count
ing number of digits in each summary
```

```
100%|██████████| 109248/109248 [00:01<00:00, 1
06391.59it/s]
```

In [55]:

```
#saving it into a dictionary. If a digit is not present in a
summary,it's value is 0
numeric_digits = {}

for i in tqdm(range(len(summaries))) :
    if i in digits_in_each_summary.keys() :
        numeric_digits[i] = digits_in_each_summary[i]
    else :
        numeric_digits[i] = 0
```

```
100%|██████████| 109248/109248 [00:00<00:00, 2
356743.71it/s]
```

In [56]:

```
#replacing actual values of digit count with 1 because we are
 only interested to know if a digit is present or not
for i in tqdm(range(len(numeric_digits))):
    if numeric_digits.get(i)>0:
        numeric_digits.update({i:1})
```

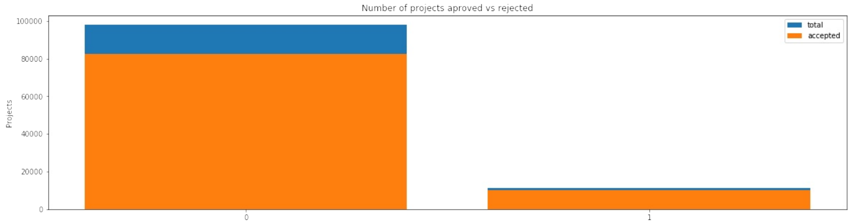
```
100%|██████████| 109248/109248 [00:00<00:00, 2
237704.98it/s]
```

In [57]:

```
#how to convert a dictionary to list: https://www.quora.com/H
ow-do-I-convert-a-dictionary-to-a-list-in-Python
digits_in_summary=[i for i in numeric_digits.values()]
project_data['digits_in_summary'] = digits_in_summary
```

In [58]:

```
#Since we have our new column,let's check if this new feature
 or column is useful to predict the acceptance rate of a proj
ect proposal

univariate_barplots(project_data, 'digits_in_summary', 'proje
ct_is_approved')
```

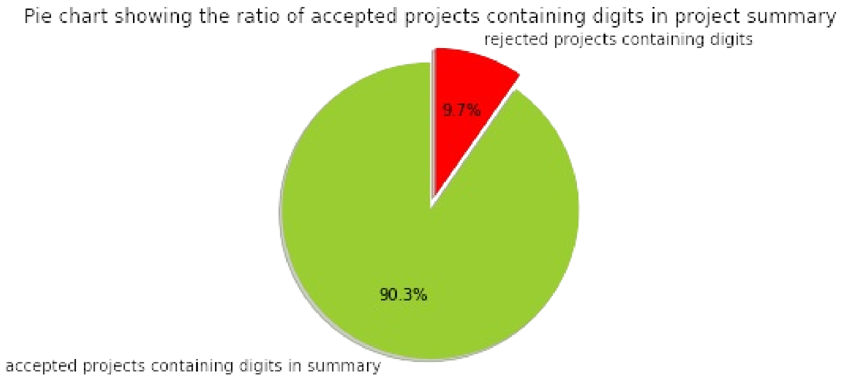


```
   digits_in_summary  project_is_approved  tot
al       Avg
0                  0                82563  980
12  0.842376
1                  1                10143  112
36  0.902723
==============================================
====
   digits_in_summary  project_is_approved  tot
```

```
al        Avg
0                   0                 82563  980
12  0.842376
1                   1                 10143  112
36  0.902723
```

In [59]:

```
accepted_digits = project_data.groupby(['digits_in_summary','project_is_approved']).size()[1][1]
rejected_digits= project_data.groupby(['digits_in_summary','project_is_approved']).size()[1][0]

# How to plot pie chart :https://stackoverflow.com/questions/39969089/how-to-create-pie-chart

labels = 'accepted projects containing digits in summary', 'rejected projects containing digits'
sizes = [accepted_digits, rejected_digits]
colors = ['yellowgreen', 'red']
explode = (0, 0.1)
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)
plt.axis('equal')
plt.title('Pie chart showing the ratio of accepted projects containing digits in project summary')
plt.show()
```



SUMMARY :

- It is not mandatory for project proposals to have numeric digits in their project summary because from the stats it is clear that a majority of approved projects do not have numeric digits in their project summary.

- However,from the pie chart,we can observe that among the project proposals with numeric digits in their summary,90% of them got accepted and about 10% got rejected

1.3 Text preprocessing

1.3.1 Essay Text

In [52]:

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

Out[52]:

	Unnamed: 0	id	teacher_id
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc
1	140945	p258326	897464ce9ddc600bced1151f324dd63a

2 rows × 21 columns

In [60]:

```
# 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 work
ing on English as their second or third langua
ges. We are a melting pot of refugees, immigra
nts, and native-born Americans bringing the gi
ft of language to our school. \r\n\r\n We have
 over 24 languages represented in our English 
Learner program with students at every level o
f mastery.  We also have over 40 countries rep
resented with the families within our school. 
 Each student brings a wealth of knowledge and
 experiences to us that open our eyes to new c
ultures, beliefs, and respect.\"The limits of 
your language are the limits of your world.\"-
Ludwig Wittgenstein  Our English learner's hav
e a strong support system at home that begs fo
r more resources.  Many times our parents are 
learning to read and speak English along side 
of their children.  Sometimes this creates bar
riers for parents to be able to help their chi
ld learn phonetics, letter recognition, and ot
her reading skills.\r\n\r\nBy providing these 
dvd's and players, students are able to contin
ue their mastery of the English language even 
if no one at home is able to assist.  All fami
lies with students within the Level 1 proficie
ncy status, will be a offered to be a part of 
this program.  These educational videos will b
e specially chosen by the English Learner Teac
her and will be sent home regularly to watch. 
 The videos are to help the child develop earl
```

y reading skills.\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\nnannan

==

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity.My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the students who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is mi

```
ssing, my students always say more Hokki Stool
s. They can't get their fill of the 5 stools w
e already have. When the students are sitting
in group with me on the Hokki Stools, they are
 always moving, but at the same time doing the
ir work. Anytime the students get to pick wher
e they can sit, the Hokki Stools are the first
 to be taken. There are always students who he
ad over to the kidney table to get one of the
stools who are disappointed as there are not e
nough of them. \r\n\r\nWe ask a lot of student
s to sit for 7 hours a day. The Hokki stools w
ill 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 min
utes a day of movement by allowing them to act
ivate their core muscles for balance while the
y sit. For many of my students, these chairs w
ill take away the barrier that exists in schoo
ls for a child who can't sit still.nannan
==============================================
====
How do you remember your days of school? Was i
t in a sterile environment with plain walls, r
ows of desks, and a teacher in front of the ro
om? A typical day in our room is nothing like
that. I work hard to create a warm inviting th
emed room for my students look forward to comi
ng 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 sc
hool, which means there is a high enough perce
ntage of free and reduced-price lunch to quali
fy. Our school is an \"open classroom\" concep
t, which is very unique as there are no walls
separating the classrooms. These 9 and 10 year
-old students are very eager learners; they ar
```

e 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 re ceive 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 teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school 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 free or reduced lunch. We aren'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 smart, effective, efficient, and disciplined students with good character.In our classroom we can utilize the Bluetooth for swift transitions during cl

```
ass. I use a speaker which doesn't amplify the
 sound enough to receive the message. Due to t
he volume of my speaker my students can't hear
 videos or books clearly and it isn't making t
he lessons as meaningful. But with the bluetoo
th speaker my students will be able to hear an
d I can stop, pause and replay it at any time.
\r\nThe cart will allow me to have more room f
or storage of things that are needed for the d
ay 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 diffe
rent letters and it is more accessible.nannan
==============================================
====
```

In [61]:

```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 [62]:

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

```
My kindergarten students have varied disabilit
ies ranging from speech and language delays, c
ognitive delays, gross/fine motor delays, to a
utism. They are eager beavers and always striv
e to work their hardest working past their lim
itations. \r\n\r\nThe materials we have are th
e ones I seek out for my students. I teach in
a Title I school where most of the students re
ceive free or reduced price lunch.  Despite th
eir disabilities and limitations, my students
love coming to school and come eager to learn
and explore.Have you ever felt like you had an
ts 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 chai
rs are the answer and I love then because they
 develop their core, which enhances gross moto
r and in Turn fine motor skills. \r\nThey also
 want to learn through games, my kids do not w
ant to sit and do worksheets. They want to lea
rn to count by jumping and playing. Physical e
ngagement is the key to our success. The numbe
r 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 deser
ves.nannan
==============================================
====
```

In [63]:

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

```
My kindergarten students have varied disabilit
ies ranging from speech and language delays, c
ognitive delays, gross/fine motor delays, to a
utism. They are eager beavers and always striv
e to work their hardest working past their lim
itations.     The materials we have are the on
es I seek out for my students. I teach in a Ti
tle I school where most of the students receiv
e 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 i
n 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 mov
e as they learn or so they say.Wobble chairs a
re the answer and I love then because they dev
elop their core, which enhances gross motor an
d 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 engagem
ent 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.na
nnan
```

In [64]:

```
#remove spacial character: https://stackoverflow.com/a/584354
7/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilit
ies ranging from speech and language delays co
gnitive delays gross fine motor delays to auti
sm They are eager beavers and always strive to
 work their hardest working past their limitat
ions The materials we have are the ones I seek
 out for my students I teach in a Title I scho
ol where most of the students receive free or
reduced price lunch Despite their disabilities
 and limitations my students love coming to sc
hool 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 a
nd I love then because they develop their core
 which enhances gross motor and in Turn fine m
otor skills They also want to learn through ga
mes my kids do not want to sit and do workshee
ts 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 ma
ts can make that happen My students will forge
t they are doing work and just have the fun a
6 year old deserves nannan
```

In [65]:

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

```
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', '
ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', '
yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "
it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', '
whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', '
being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', '
if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in
', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where',
 'why', 'how', 'all', 'any', 'both', 'each', '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', "co
uldn't", 'didn', "didn't", 'doesn', "doesn'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 [66]:

```
# 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 stopw
ords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:45<00:00, 2
397.86it/s]
```

In [67]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[67]:

```
'my kindergarten students varied disabilities
ranging speech language delays cognitive delay
s gross fine motor delays autism they eager be
avers always strive work hardest working past
limitations the materials ones i seek students
 i teach title i school students receive free
reduced price lunch despite disabilities limit
ations students love coming school come eager
learn explore have ever felt like ants pants n
eeded groove move meeting this kids feel time
the want able move learn say wobble chairs ans
wer i love develop core enhances gross motor t
urn fine motor skills they also want learn gam
es kids not want sit worksheets they want lear
n count jumping playing physical engagement ke
y success the number toss color shape mats mak
e happen my students forget work fun 6 year ol
d deserves nannan'
```

1.3.2 Project title Text

In [68]:

```
# similarly you can preprocess the titles also

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 stopw
ords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:01<00:00, 5
5441.18it/s]
```

1. 4 Preparing data for models

In [69]:

```
project_data.columns
```

Out[69]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
her_prefix', 'school_state',
       'project_submitted_datetime', 'project_
title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', '
project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'project_grade_category', 'clean_catego
ries', 'clean_subcategories',
       'essay', 'price', 'quantity', 'digits_i
n_summary'],
      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 : nu
```

```
merical
    - price : numerical
```

1.4.1 Vectorizing Categorical data

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

In [70]:

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


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'M
usic_Arts', 'AppliedLearning', 'SpecialNeeds',
 'Health_Sports', 'Math_Science', 'Literacy_La
nguage']
Shape of matrix after one hot encoding  (10924
8, 9)
```

In [71]:

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

```
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot = vectorizer.transform(project_data['clean_subcategories'].values)
print("Shape of matrix after one hot encoding ",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 encoding  (109248, 30)
```

In [72]:

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

my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [73]:

```
school_state_cat_dict = dict(my_counter)
```

```
sorted_school_state_cat_dict = dict(sorted(school_state_cat_dict.items(), key=lambda kv: kv[1]))
```

In [74]:

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

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

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

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE
', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', '
KS', 'IA', '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 encoding  (10924
8, 51)
```

In [75]:

```
my_counter = Counter()
for project_grade in project_data['project_grade_category'].values:
    my_counter.update(project_grade.split())
```

In [76]:

```
project_grade_cat_dict = dict(my_counter)
```

```
sorted_project_grade_cat_dict = dict(sorted(project_grade_cat_dict.items(), key=lambda kv: kv[1]))
```

In [77]:

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

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

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

```
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'G
rades_PreK_2']
Shape of matrix after one hot encoding  (10924
8, 4)
```

In [78]:

```
my_counter = Counter()
for teacher_prefix in project_data['teacher_prefix'].values:
    teacher_prefix = str(teacher_prefix)
    my_counter.update(teacher_prefix.split())
```

In [79]:

```
teacher_prefix_cat_dict = dict(my_counter)
sorted_teacher_prefix_cat_dict = dict(sorted(teacher_prefix_cat_dict.items(), key=lambda kv: kv[1]))
```

In [80]:

```
#https://stackoverflow.com/questions/39303912/tfidfvectorizer
```

```
-in-scikit-learn-valueerror-np-nan-is-an-invalid-document
#ValueError: np.nan is an invalid document, expected byte or unicode string.
vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values.astype("U"))
print(vectorizer.get_feature_names())

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

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

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [81]:

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

```
Shape of matrix after one hot encoding  (109248, 16623)
```

1.4.2.2 Bag of Words on $project_t it \leq$

In [82]:

```
#you can vectorize the title also
#before you vectorize the title make sure you preprocess it
vectorizer = CountVectorizer(min_df=10)
project_title_bow = vectorizer.fit_transform(preprocessed_tit
les)
print("Shape of matrix after one hot encoding ",project_title
_bow.shape)
```

```
Shape of matrix after one hot encoding  (10924
8, 3329)
```

1.4.2.3 TFIDF vectorizer

In [83]:

```
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 encoding ",text_tfidf.sh
ape)
```

```
Shape of matrix after one hot encoding  (10924
8, 16623)
```

1.4.2.4 TFIDF Vectorizer on $project_t it \leq$

In [84]:

```
# Similarly you can vectorize for title also
vectorizer = TfidfVectorizer(min_df=10)
```

```
project_titles_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encoding ",project_titles_tfidf.shape)
```

```
Shape of matrix after one hot encoding  (10924
8, 3329)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [80]:

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

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

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

```
# ============================

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

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

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vec
tors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(word
s)*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.je
ssicayung.com/how-to-use-pickle-to-save-and-load-variables-in
-python/

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


'''
```

Out[80]:

```
'\n# Reading glove vectors in python: https://
stackoverflow.com/a/38230349/4084039\ndef load
GloveModel(gloveFile):\n    print ("Loading Gl
ove Model")\n    f = open(gloveFile,\'r\', enc
oding="utf8")\n    model = {}\n    for line in
 tqdm(f):\n        splitLine = line.split()\n
       word = splitLine[0]\n        embedding
= np.array([float(val) for val in splitLine[1:
]])\n        model[word] = embedding\n    prin
t ("Done.",len(model)," words loaded!")\n    r
eturn model\nmodel = loadGloveModel(\'glove.42
B.300d.txt\')\n\n# ===========================
=\nOutput:\n    \nLoading Glove Model\n1917495
it [06:32, 4879.69it/s]\nDone. 1917495  words
loaded!\n\n# ============================\n\nw
ords = []\nfor i in preproced_texts:\n    word
s.extend(i.split(\' \'))\n\nfor i in preproced
_titles:\n    words.extend(i.split(\' \'))\npr
int("all the words in the coupus", len(words))
\nwords = set(words)\nprint("the unique words
in the coupus", len(words))\n\ninter_words = s
et(model.keys()).intersection(words)\nprint("T
he number of words that are present in both gl
ove vectors and our coupus",       len(inter_w
ords),"(",np.round(len(inter_words)/len(words)
*100,3),"%)")\n\nwords_courpus = {}\nwords_glo
ve = set(model.keys())\nfor i in words:\n    i
f i in words_glove:\n        words_courpus[i]
= model[i]\nprint("word 2 vec length", len(wor
ds_courpus))\n\n\n# stronging variables into p
ickle files python: http://www.jessicayung.com
/how-to-use-pickle-to-save-and-load-variables-
in-python/\n\nimport pickle\nwith open(\'glove
_vectors\', \'wb\') as f:\n    pickle.dump(wor
ds_courpus, f)\n\n\n'
```

In [85]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [86]:

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

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

```
100%|██████████| 109248/109248 [00:19<00:00, 5730.32it/s]
```

```
109248
```

```
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on $project_tit \leq$

In [87]:

```
# Similarly you can vectorize for title also
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles.append(vector)

print(len(avg_w2v_titles))
print(len(avg_w2v_titles[0]))
```

```
100%|██████████| 109248/109248 [00:01<00:00, 102240.02it/s]
```

```
109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [88]:

```
# 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 [89]:

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

```
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|██████████| 109248/109248 [02:24<00:00, 7
55.80it/s]
```

```
109248
300
```

In [90]:

```
# Similarly you can vectorize for title also

tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
# we are converting a dictionary with word as a key, and the
idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

1.4.2.9 Using Pretrained Models: TFIDF weighted W2V on $project_{t}it \leq$

In [91]:

```
# Similarly you can vectorize for title also
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_titles = []; # the avg-w2v for each sentence/review
```

```
 is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_titles.append(vector)

print(len(tfidf_w2v_titles))
print(len(tfidf_w2v_titles[0]))
```

```
100%|██████████| 109248/109248 [00:02<00:00, 43812.54it/s]
```

```
109248
300
```

1.4.3 Vectorizing Numerical features

In [92]:

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

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

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

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

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

In [93]:

```
price_standardized
```

Out[93]:

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

In [94]:

```
quantity_scalar = StandardScaler()

## Finding the mean and standard deviation of this data
quantity_scalar.fit(project_data['quantity'].values.reshape(-1
,1))

print("Mean : {}".format(quantity_scalar.mean_[0]))

print("Standard deviation : {}".format(np.sqrt(quantity_scala
r.var_[0])))

# Now standardize the data with above maen and variance.
quantity_standardized = quantity_scalar.transform(project_dat
a['quantity'].values.reshape(-1, 1))
```

```
Mean : 16.965610354422964
Standard deviation : 26.182821919093175
```

In [95]:

```
quantity_standardized
```

Out[95]:

```
array([[ 0.23047132],
       [-0.60977424],
       [ 0.19227834],
       ...,
       [-0.4951953 ],
       [-0.03687954],
       [-0.45700232]])
```

In [96]:

```
previously_posted_projects_scalar = StandardScaler()

## Finding the mean and standard deviation of this data
previously_posted_projects_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("Mean : {}".format(previously_posted_projects_scalar.mean_[0]))

print("Standard deviation : {}".format(np.sqrt(previously_posted_projects_scalar.var_[0])))

# Now standardize the data with above maen and variance.
previously_posted_projects_scalar_standardized = previously_posted_projects_scalar.transform(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
```

```
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 [97]:

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

```
(109248, 9)
(109248, 30)
(109248, 16623)
```

```
(109248, 1)
```

In [98]:

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

Out[98]:

```
(109248, 16663)
```

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_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)
   - price : numerical
   - teacher_number_of_previously_posted_projects : numerical
4. Now, plot FOUR t-SNE plots with each of these feature sets.
   A. categorical, numerical features + project_title(BOW)
   B. categorical, numerical features + project_title(TFIDF)
   C. categorical, numerical features + project_title(AVG W2V)
   D. categorical, numerical features + project_title(TFIDF W2V)
5. Concatenate all the features and Apply TNSE on the final data matrix
6. Note 1: The TSNE accepts only dense matrices
7. Note 2: Consider only 5k to 6k data points to avoid memory issues. If you run into memory error issues, reduce the number of data points but clearly state the number of datat-poins you are using

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

In [99]:

```
#stacking all the layers of columns from the above shells
X = hstack((categories_one_hot, sub_categories_one_hot, school_state_category_one_hot, project_grade_category_one_hot, teacher_prefix_categories_one_hot, price_standardized, quantity_standardized, previously_posted_projects_scalar_standardized, project_title_bow))
X.shape
```

Out[99]:

```
(109248, 3431)
```

In [100]:

```
#Plotting only first 5k datapoints due to computational constraints
from sklearn.manifold import TSNE
X = X.tocsr()   #Tsne only accepts dense matrices
X_5000 = X[0:5000,:]
```

In [101]:

```
X_5000 = X_5000.toarray()
model = TSNE(n_components = 2, perplexity = 50, random_state = 0)
tsne_data_bow = model.fit_transform(X_5000)
```

In [102]:

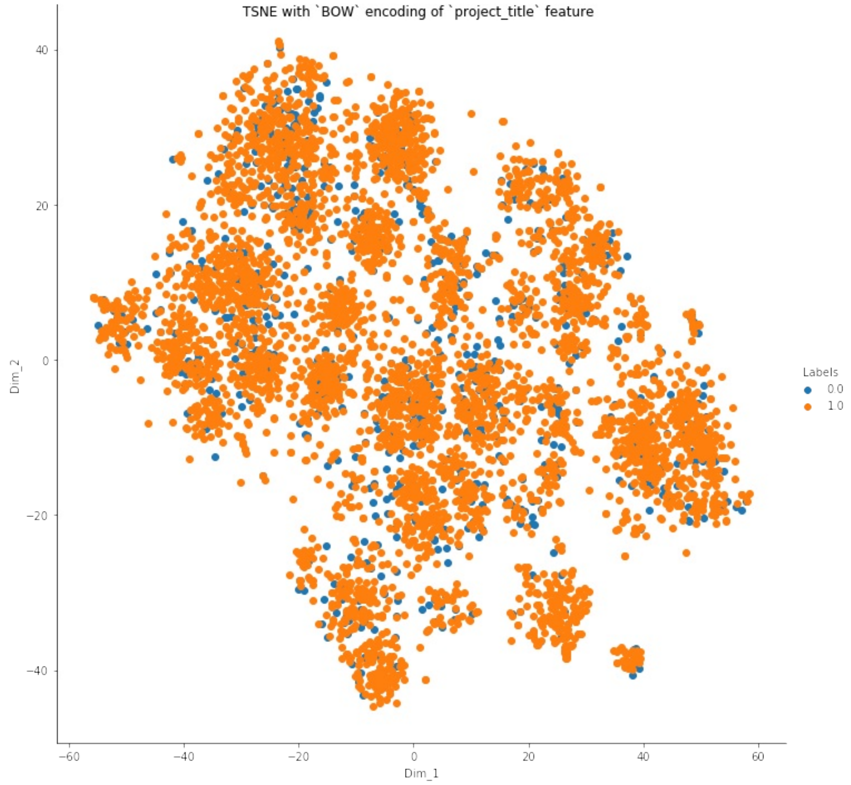
```
labels = project_data["project_is_approved"]
labels_5000 = labels[0: 5000]
```

In [103]:

```
tsne_data_bow = np.vstack((tsne_data_bow.T, labels_5000)).T
tsne_df_bow = pd.DataFrame(tsne_data_bow, columns = ("Dim_1", "Dim_2","Labels"))
```

In [104]:

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

sns.FacetGrid(tsne_df_bow, hue = "Labels", size = 10).map(plt.scatter, "Dim_1", "Dim_2").add_legend().fig.suptitle("TSNE with `BOW` encoding of `project_title` feature  ")
plt.show()
```


TSNE with `BOW` encoding of `project_title` feature
Dim_2
40
20
0
−20
−40
Dim_1
−60
−40
−20
0
20
40
60
Labels
● 0.0
● 1.0

Summary :

- There is a lot of overlapping in the datapoints

- The datapoints based on their class labels are not well separated,not much sense can be made out of this plot

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

In [105]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, school_state_category_one_hot, project_grade_category_one_hot, teacher_prefix_categories_one_hot, price_standardized, quantity_standardized, previously_posted_projects_scalar_standardized, project_titles_tfidf))
X.shape
```

Out[105]:

```
(109248, 3431)
```

In [106]:

```
X = X.tocsr()   #Tsne only accepts dense matrices
X_5000 = X[0:5000,:]
```

In [107]:

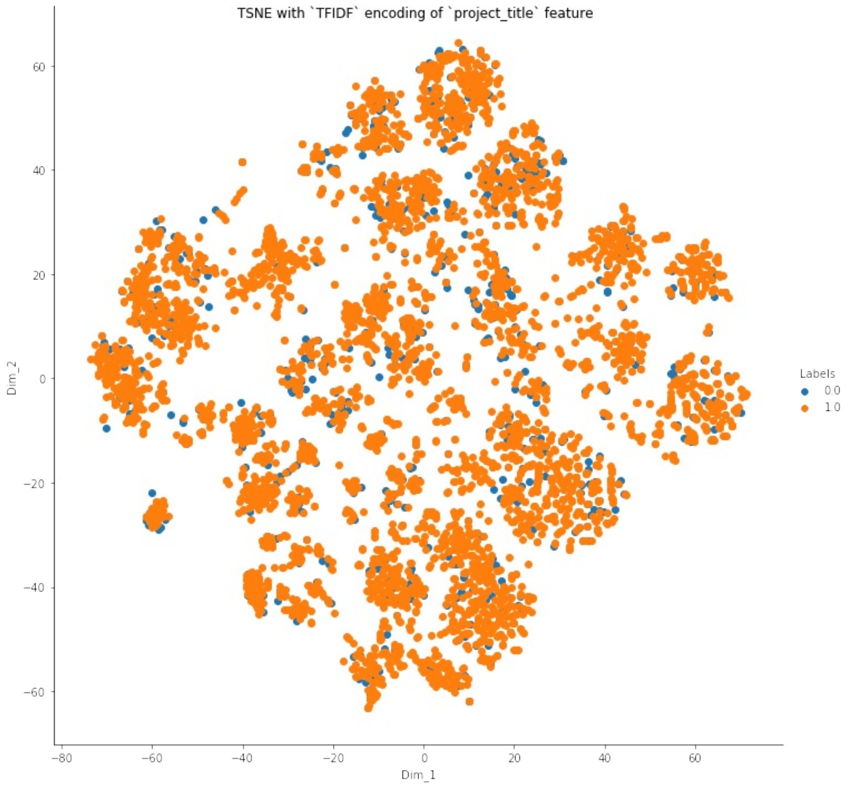
```
X_5000 = X_5000.toarray()
model = TSNE(n_components = 2, perplexity = 50, random_state = 0)
tsne_data_tfidf = model.fit_transform(X_5000)
```

In [108]:

```
tsne_data_tfidf = np.vstack((tsne_data_tfidf.T, labels_5000)).T
tsne_df_tfidf = pd.DataFrame(tsne_data_tfidf, columns = ("Dim_1","Dim_2","Labels"))
```

In [109]:

```
sns.FacetGrid(tsne_df_tfidf, hue = "Labels", size = 10).map(p
lt.scatter, "Dim_1", "Dim_2").add_legend().fig.suptitle("TSNE
 with `TFIDF` encoding of `project_title` feature  ")
plt.show()
```



Summary :

- The datapoints seem to be somewhat scattered,but the overlapping still exists.

- Not much sense can be made out of this plot

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

In [110]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, schoo
l_state_category_one_hot, project_grade_category_one_hot, tea
cher_prefix_categories_one_hot, price_standardized, quantity_
standardized, previously_posted_projects_scalar_standardized,
 avg_w2v_titles))
X.shape
```

Out[110]:

```
(109248, 402)
```

In [111]:

```
X = X.tocsr()   #Tsne only accepts dense matrices
X_5000 = X[0:5000,:]
```

In [112]:

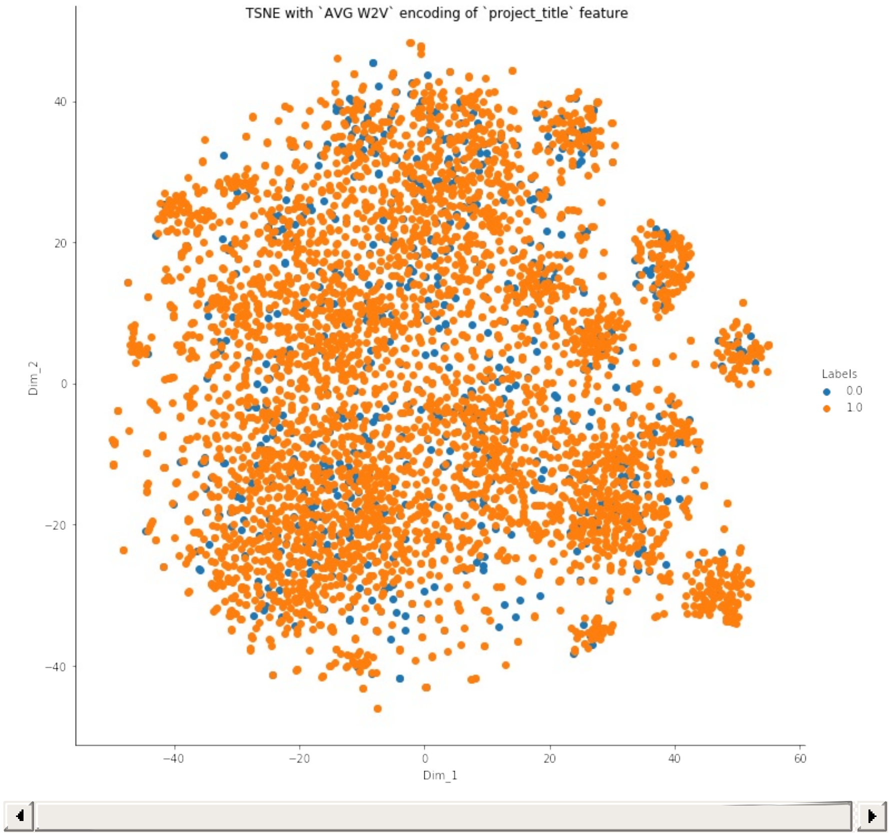
```
X_5000 = X_5000.toarray()
model = TSNE(n_components = 2, perplexity = 50, random_state
= 0)
tsne_data_avg_w2v = model.fit_transform(X_5000)
```

In [113]:

```
tsne_data_avg_w2v = np.vstack((tsne_data_avg_w2v.T, labels_50
00)).T
tsne_df_avg_w2v = pd.DataFrame(tsne_data_avg_w2v, columns = (
"Dim_1","Dim_2","Labels"))
```

In [114]:

```
sns.FacetGrid(tsne_df_avg_w2v, hue = "Labels", size = 10).map
(plt.scatter, "Dim_1", "Dim_2").add_legend().fig.suptitle("TS
NE with `AVG W2V` encoding of `project_title` feature  ")
plt.show()
```



Summary :

- There is a lot of overlapping in the datapoints

- The datapoints based on their class labels are not well scattered or separated,so no proper conclusion can be drawn

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

In [115]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, schoo
l_state_category_one_hot, project_grade_category_one_hot, tea
cher_prefix_categories_one_hot, price_standardized, quantity_
standardized, previously_posted_projects_scalar_standardized,
 tfidf_w2v_titles))
X.shape
```

Out[115]:

```
(109248, 402)
```

In [116]:

```
X = X.tocsr()   #Tsne only accepts dense matrices
X_5000 = X[0:5000,:]
```

In [117]:

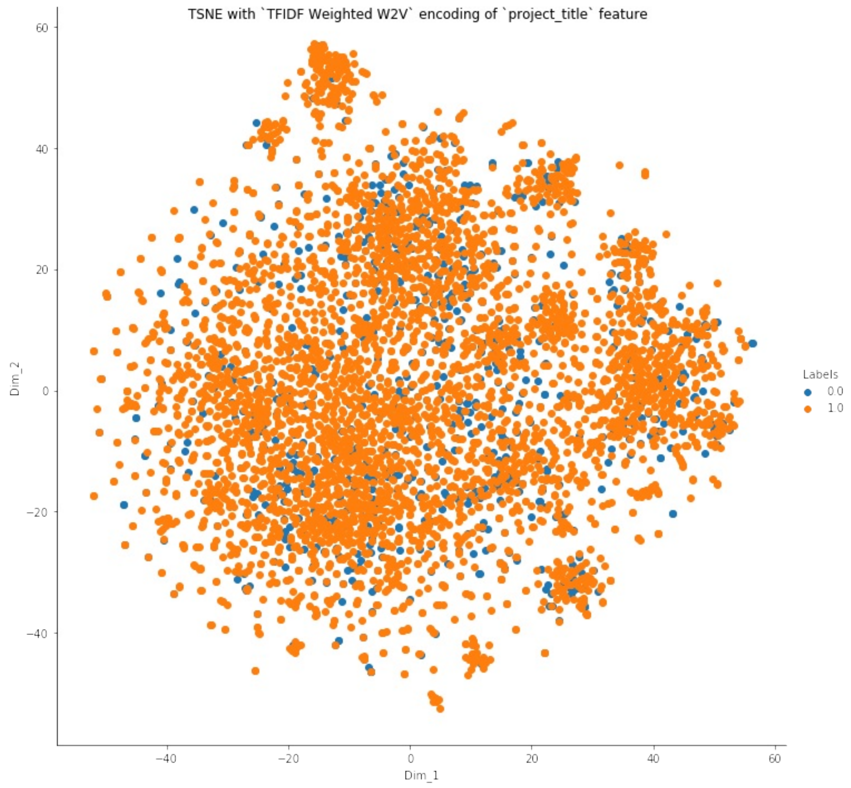
```
X_5000 = X_5000.toarray()
model = TSNE(n_components = 2, perplexity = 50, random_state
= 0)
tsne_data_tfidf_w2v = model.fit_transform(X_5000)
```

In [118]:

```
tsne_data_tfidf_w2v = np.vstack((tsne_data_tfidf_w2v.T, label
s_5000)).T
tsne_df_tfidf_w2v = pd.DataFrame(tsne_data_tfidf_w2v, columns
 = ("Dim_1","Dim_2","Labels"))
```

In [119]:

```
sns.FacetGrid(tsne_df_tfidf_w2v, hue = "Labels", size = 10).map(plt.scatter, "Dim_1", "Dim_2").add_legend().fig.suptitle("TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature  ")
plt.show()
```



Summary :

- Even here, there seems to be a lot of overlapping in the datapoints

- The datapoints based on their class labels are not well scattered or separated,so no proper conclusion can be drawn from this plot

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

In [120]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, schoo
l_state_category_one_hot, project_grade_category_one_hot, tea
cher_prefix_categories_one_hot, price_standardized, quantity_
standardized, previously_posted_projects_scalar_standardized,
project_title_bow,project_titles_tfidf, avg_w2v_titles, tfidf
_w2v_titles))
X.shape
```

Out[120]:

```
(109248, 7360)
```

In [121]:

```
X = X.tocsr()   #Tsne only accepts dense matrices
X_5000 = X[0:5000,:]
```

In [122]:

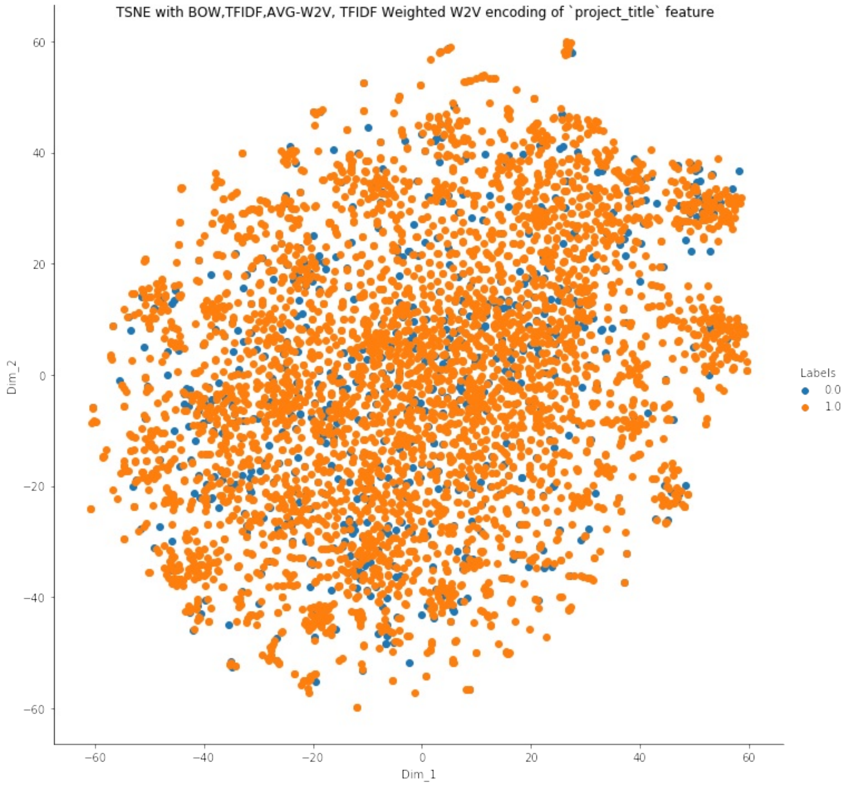
```
X_5000 = X_5000.toarray()
model = TSNE(n_components = 2, perplexity = 50, random_state
= 0)
tsne_data_final = model.fit_transform(X_5000)
```

In [123]:

```
tsne_data_final = np.vstack((tsne_data_final.T, labels_5000))
.T
tsne_df_final = pd.DataFrame(tsne_data_final, columns = ("Dim
_1","Dim_2","Labels"))
```

In [124]:

```
sns.FacetGrid(tsne_df_final, hue = "Labels", size = 10).map(plt.scatter, "Dim_1", "Dim_2").add_legend().fig.suptitle("TSNE with BOW,TFIDF,AVG-W2V, TFIDF Weighted W2V encoding of `project_title` feature  ")
plt.show()
```



Summary :

- **Even here, there seems to be a lot of overlapping in the datapoints**

- **The datapoints based on their class labels are not**

well scattered or separated,so no proper conclusion can be drawn from this plot

Conclusion

- The EDA of this dataset gives us some useful insights on how the features are correlated with the approval status of a project proposal. However,that can't be generalized,so we used T-SNE with the objective of grouping datapoints based on their class labels.

- However, The Visualisation of TSNE with Bag of Words, TF-IDF, Avg Word2Vec, TF-IDF Weighted Word2Vec does not seem to yield the expected result of clustering similar data points

- So alternate methods have to be tried on this dataset so that the approval status of thousands of project proposals could be automated instead of manually screening each of them