DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [98]:

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

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

```
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [99]:

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

Out[99]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

In [100]:

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

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

Out[100]:

	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
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [101]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
```

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

In [102]:

```
project_data= project_data.head(5000)
```

In [103]:

```
project_data.shape
```

Out[103]:

```
(5000, 17)
```

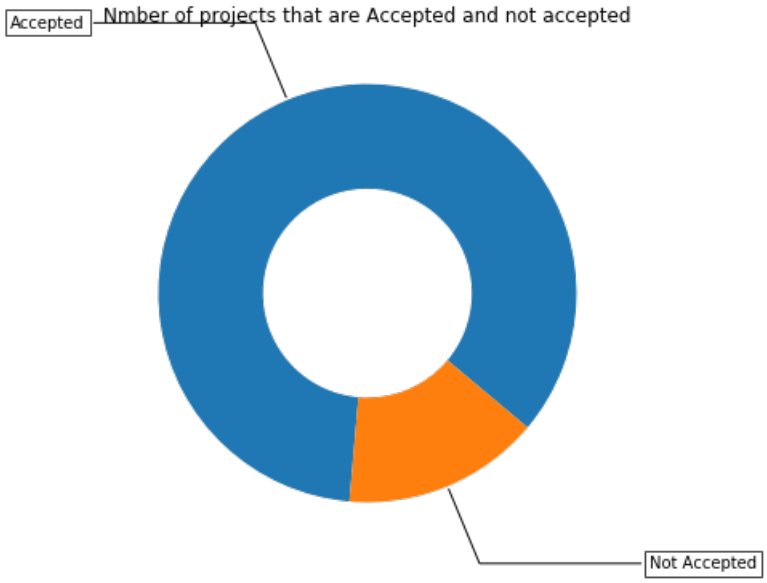
1.2 Exploratory Data Analysis

In [104]:

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


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

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

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

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

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

for i, p in enumerate(wedges):
```

```
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    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  4237 , ( 84.74000000000001 %)
Number of projects thar are not approved for funding  763 , ( 15.260000000000002 %)
```



1.2.1 Univariate Analysis: School State

In [105]:

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

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

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

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

```
    )

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

Out[105]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n\nscl = [[0.0, \'rg
b(242,240,247)\'],[0.2, \'rgb(218,218,235)\'],[0.4, \'rgb(188,189,220)\'],            [0.6, \'rgb(1
58,154,200)\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\ndata = [ dict(\n        ty
pe=\'choropleth\',\n        colorscale = scl,\n        autocolorscale = False,\n        locations =
temp[\'state_code\'],\n        z = temp[\'num_proposals\'].astype(float),\n        locationmode = \
'USA-states\',\n        text = temp[\'state_code\'],\n        marker = dict(line = dict (color = \'
rgb(255,255,255)\',width = 2)),\n        colorbar = dict(title = "% of pro")\n    ) ]\n\nlayout = d
ict(\n        title = \'Project Proposals % of Acceptance Rate by US States\',\n        geo = dict(
\n            scope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n            show
akes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n        ),\n    )\n\nfig =
go.Figure(data=data, layout=layout)\noffline.iplot(fig, filename=\'us-map-heat-map\')\n'
```

In [106]:

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

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.500000
41         SD       0.687500
7          DC       0.695652
0          AK       0.705882
50         WY       0.777778
==================================================
States with highest % approvals
   state_code  num_proposals
11         HI       0.964286
16         KS       0.966667
28         ND       1.000000
8          DE       1.000000
30         NH       1.000000
```

In [107]:

```
#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()
```

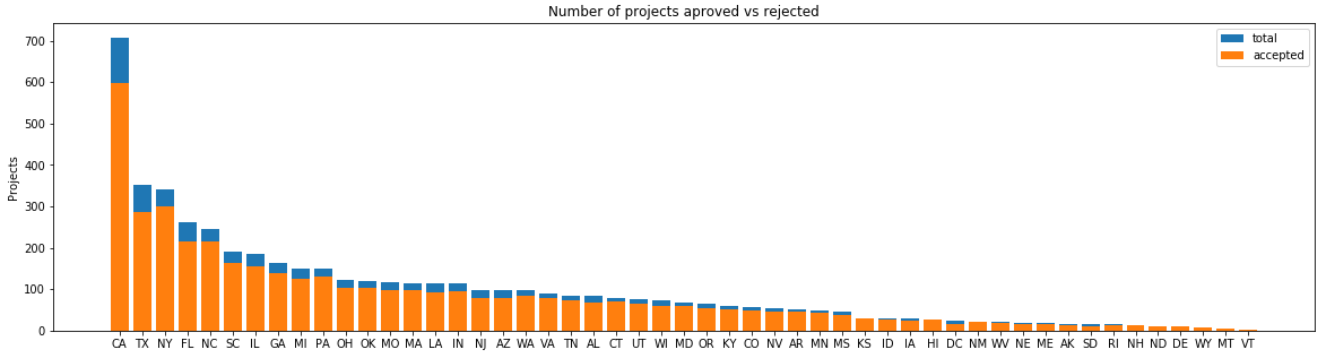
In [108]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/4084039
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).reset_index(
)

    # Pandas dataframe grouby count: https://stackoverflow.com/a/19385591/4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)
```

```
[col2].agg({'total':'count'})).reset_index()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':'mean'})).reset_index()[
'Avg']

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

    if top:
        temp = temp[0:top]

    stack_plot(temp, xtick=col1, col2=col2, col3='total')
    print(temp.head(5))
    print("="*50)
    print(temp.tail(5))
```

In [109]:

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



	school_state	project_is_approved	total	Avg
4	CA	597	707	0.844413
43	TX	286	352	0.812500
34	NY	299	342	0.874269
9	FL	215	261	0.823755
27	NC	216	246	0.878049

==========

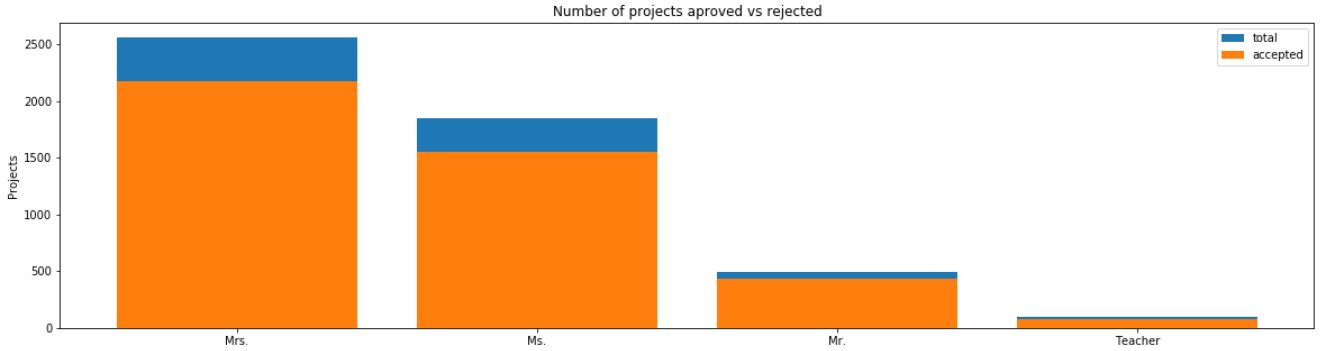
	school_state	project_is_approved	total	Avg
28	ND	11	11	1.000000
8	DE	11	11	1.000000
50	WY	7	9	0.777778
26	MT	5	6	0.833333
46	VT	1	2	0.500000

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

1.2.2 Univariate Analysis: teacher_prefix

In [110]:

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



	teacher_prefix	project_is_approved	total	Avg
1	Mrs.	2173	2560	0.848828

```
2            Ms.                   1554    1845  0.842276
0            Mr.                    433     495  0.874747
3        Teacher                     77     100  0.770000
==================================================
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                  2173   2560  0.848828
2            Ms.                  1554   1845  0.842276
0            Mr.                   433    495  0.874747
3        Teacher                    77    100  0.770000
```

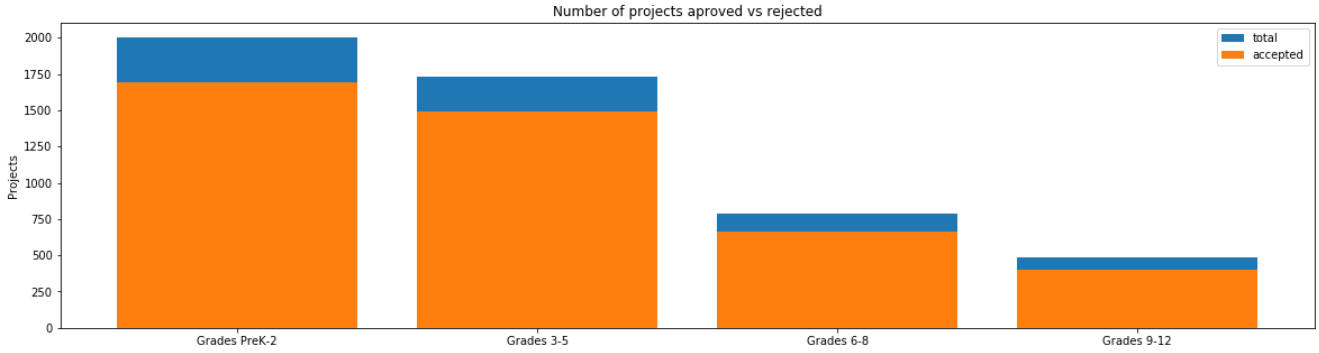
summery:everyone has more than 80% project approved

female teachers have more no of projects proposed and accepted than male teachers

1.2.3 Univariate Analysis: project_grade_category

In [111]:

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




```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                1689   2002  0.843656
0             Grades 3-5                1491   1729  0.862348
1             Grades 6-8                 660    785  0.840764
2            Grades 9-12                 397    484  0.820248
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                1689   2002  0.843656
0             Grades 3-5                1491   1729  0.862348
1             Grades 6-8                 660    785  0.840764
2            Grades 9-12                 397    484  0.820248
```

There are more no. of projects approved and proposed from grade preK-2 to grade 5.rest are decreasing

1.2.4 Univariate Analysis: project_subject_categories

In [112]:

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

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

```
.e removing 'the')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [113]:

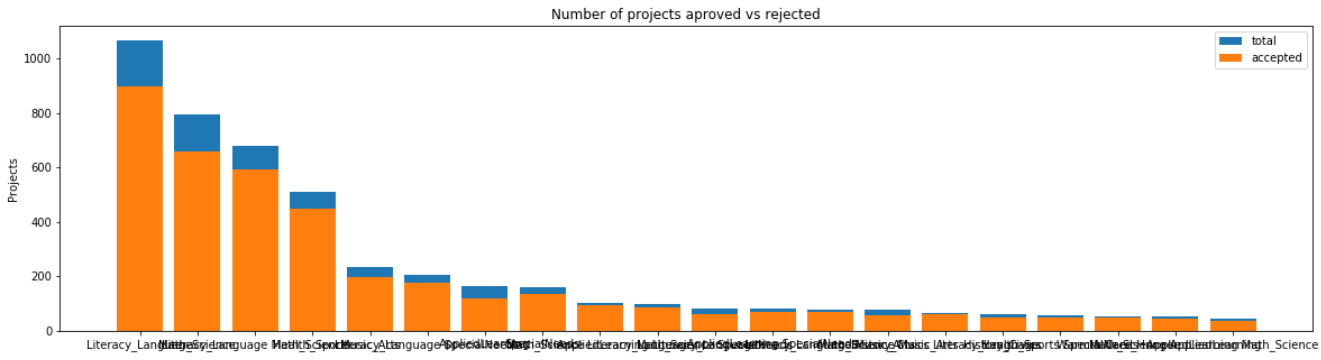
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(5)
```

Out[113]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

In [114]:

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




```
                clean_categories  project_is_approved  total       Avg
23               Literacy_Language                  900   1067  0.843486
30                    Math_Science                  659    795  0.828931
26  Literacy_Language Math_Science                  594    679  0.874816
8                   Health_Sports                  447    509  0.878193
37                     Music_Arts                  199    233  0.854077
==================================================
              clean_categories  project_is_approved  total       Avg
16               History_Civics                   47     63  0.746032
14    Health_Sports SpecialNeeds                  49     57  0.859649
46           Warmth Care_Hunger                   47     53  0.886792
31  Math_Science AppliedLearning                  44     52  0.846154
4   AppliedLearning Math_Science                  35     44  0.795455
```

For literacy and language project proposed and acceptance rate is more which is 87%

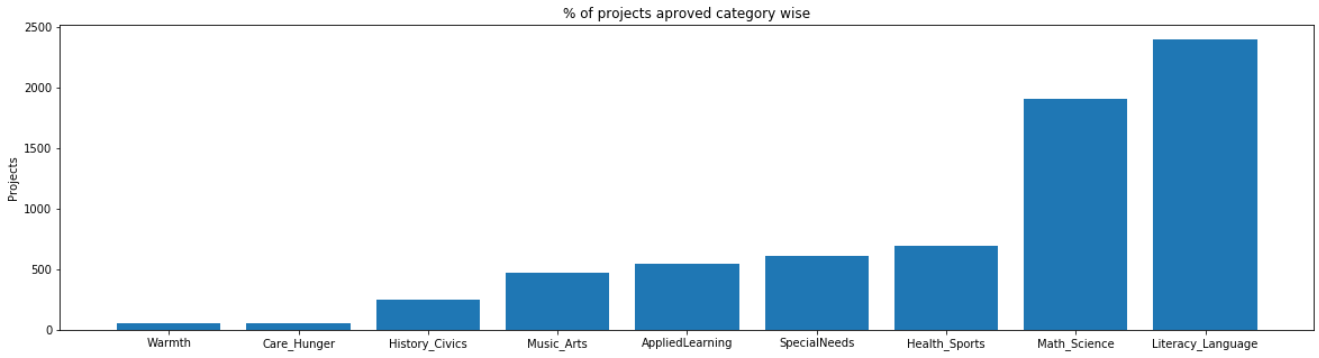
All categories have more than 80% of project acceptance

In [115]:

```
# 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 [116]:

```
# 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 [117]:

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

```
Warmth               :                  58
Care_Hunger          :                  58
History_Civics       :                 252
Music_Arts           :                 476
AppliedLearning      :                 547
SpecialNeeds         :                 614
Health_Sports        :                 697
Math_Science         :                1910
Literacy_Language    :                2400
```

Highest no of projects approved for Literacy following with math science

warmth and care hunger has less no of projects approved

1.2.5 Univariate Analysis: project_subject_subcategories

In [118]:

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

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

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

In [119]:

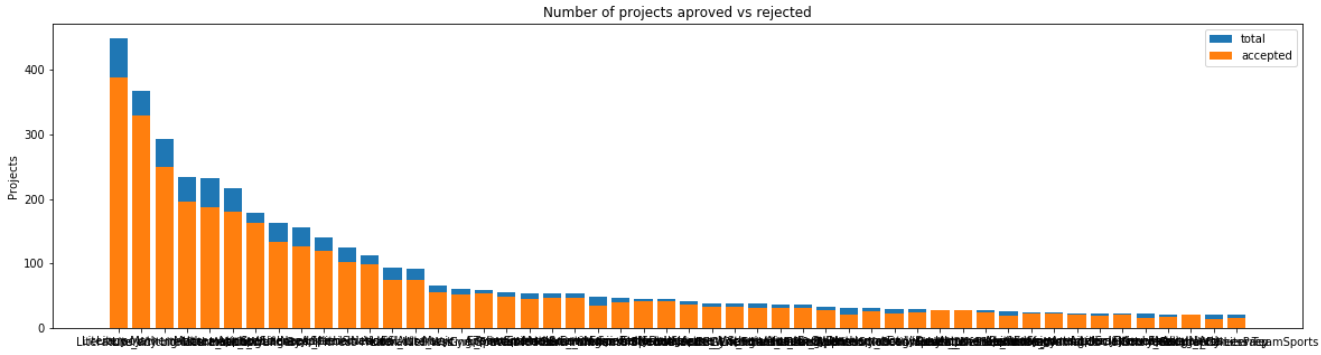
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[119]:

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

In [120]:

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




```
                clean_subcategories  project_is_approved  total       Avg
189                        Literacy                  389    449  0.866370
191             Literacy Mathematics                  329    368  0.894022
201  Literature_Writing Mathematics                  250    293  0.853242
190     Literacy Literature_Writing                  195    234  0.833333
209                     Mathematics                  188    232  0.810345
==================================================
            clean_subcategories  project_is_approved  total       Avg
23   AppliedSciences VisualArts                   15     22  0.681818
230          Other SpecialNeeds                   17     21  0.809524
181  History_Geography Literacy                   20     21  0.952381
56           College_CareerPrep                   14     20  0.700000
177  Health_Wellness TeamSports                   15     20  0.750000
```

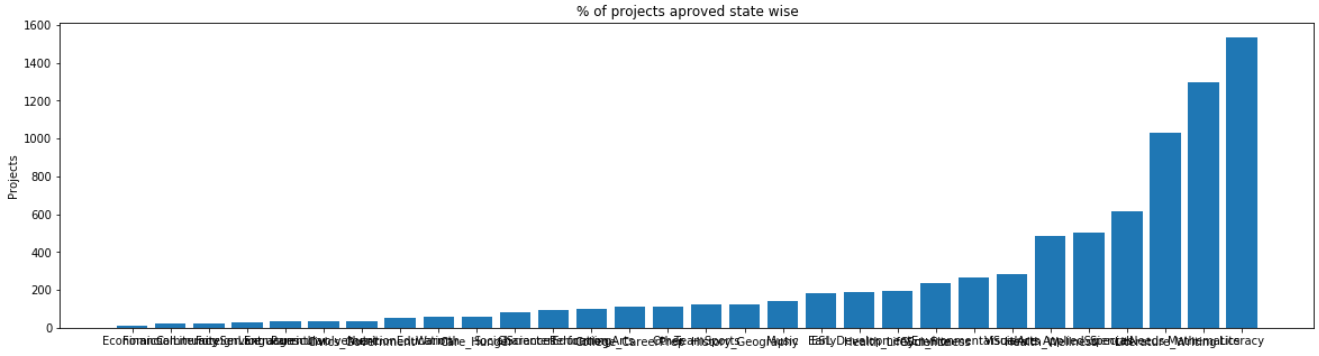
literacy has more no. of projects approved.its acceptance rate is 89%

In [121]:

```
# 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 [122]:

```
# 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 [123]:

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

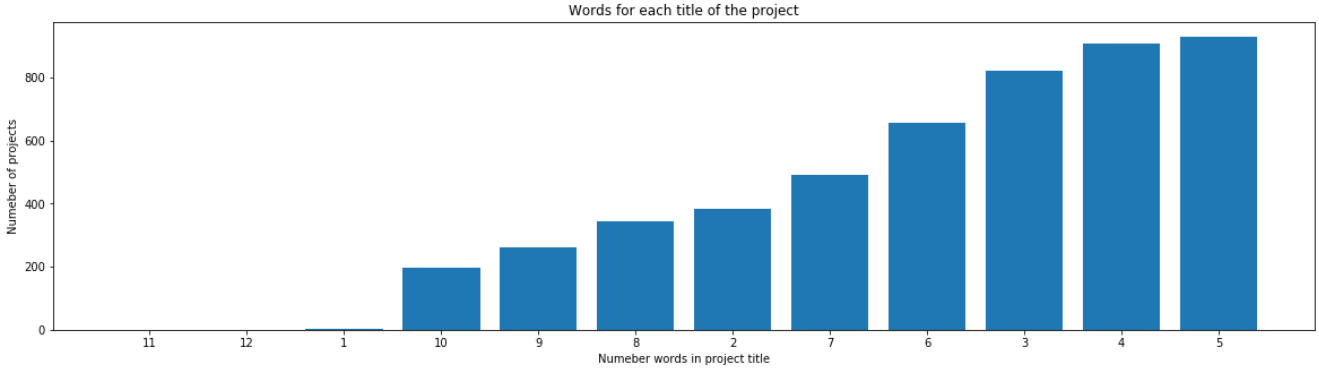
```
Economics            :        14
FinancialLiteracy    :        23
CommunityService     :        26
ForeignLanguages     :        29
Extracurricular      :        33
ParentInvolvement    :        34
Civics_Government    :        36
NutritionEducation   :        54
Warmth               :        58
Care_Hunger          :        58
SocialSciences       :        82
CharacterEducation   :        95
PerformingArts       :       102
College_CareerPrep   :       113
Other                :       114
TeamSports           :       123
History_Geography    :       124
Music                :       142
ESL                  :       182
EarlyDevelopment     :       189
Health_LifeScience   :       196
Gym_Fitness          :       237
EnvironmentalScience :       265
VisualArts           :       282
Health_Wellness      :       486
AppliedSciences      :       504
SpecialNeeds         :       614
```

```
Literature_Writing    :      1032
Mathematics           :      1295
Literacy              :      1534
```

1.2.6 Univariate Analysis: Text features (Title)

In [124]:

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


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

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



most of the project have 4 no. of word in project title

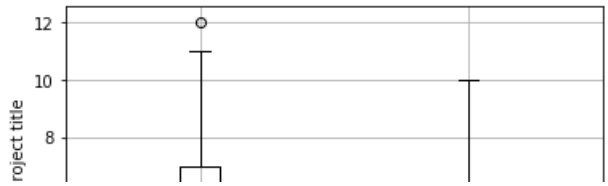
hardly some projects have 10 no.of words in project title

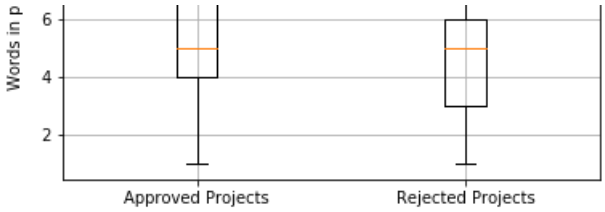
In [125]:

```
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 [126]:

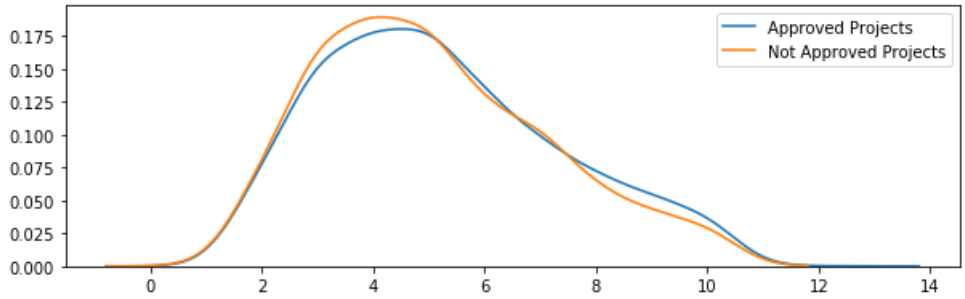
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```





In [127]:

```
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()
```



Approved projects have slightly more no of words in project title compared to not approved projects

Box plot uses percentile

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

In [128]:

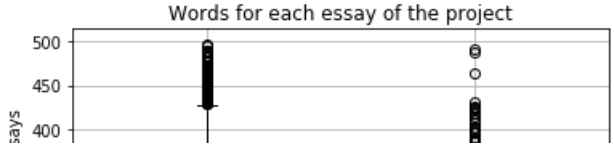
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

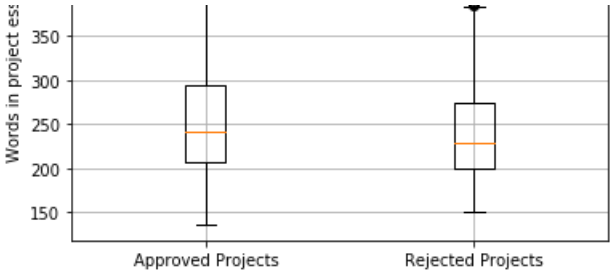
In [129]:

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

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

In [130]:

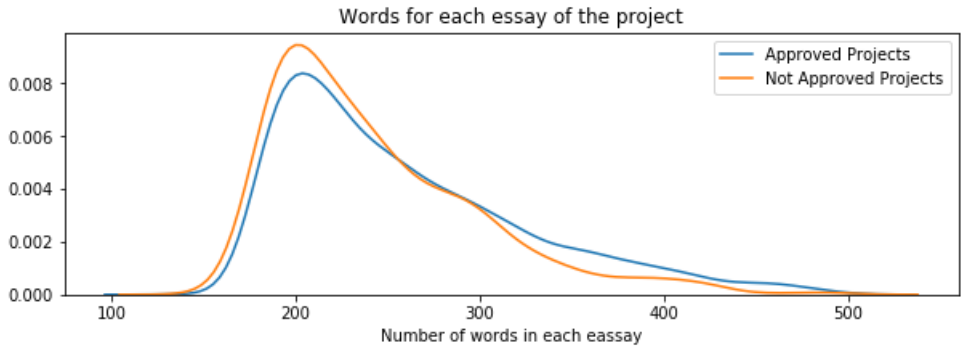
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()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





In [131]:

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



1.2.8 Univariate Analysis: Cost per project

In [132]:

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

Out[132]:

	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 [133]:

```
# 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[133]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [134]:

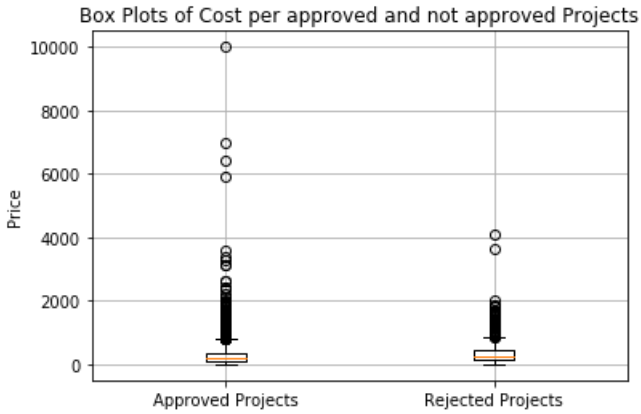
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [135]:

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

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

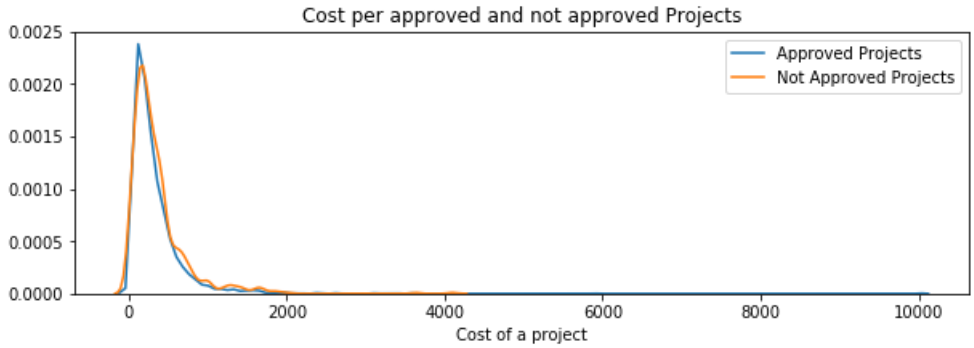
In [136]:

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



In [137]:

```
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()
```



1.from box plot not understandig much as it is overlapping

1.but from PDF we can see projects having high cost are not approved

In [138]:

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

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

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

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

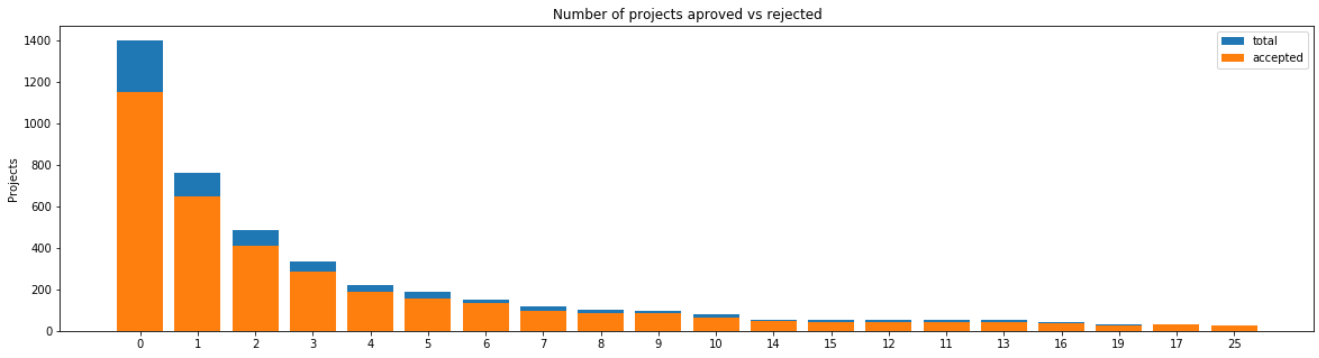
```
+------------+--------------------+------------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+--------------------+------------------------+
|      0     |        1.44        |          5.19          |
|      5     |       14.664       |         40.045         |
|     10     |       35.41        |         75.106         |
|     15     |       56.788       |         104.181        |
|     20     |       75.848       |         126.288        |
|     25     |       100.21       |         145.665        |
|     30     |      119.948       |         159.996        |
|     35     |       139.99       |         180.088        |
|     40     |       159.43       |         207.546        |
|     45     |       179.0        |         232.279        |
|     50     |       200.77       |         258.07         |
|     55     |      229.636       |         289.256        |
|     60     |      259.744       |         314.946        |
|     65     |      288.936       |         357.846        |
|     70     |      326.598       |         393.798        |
|     75     |       376.51       |         428.41         |
|     80     |      423.724       |         483.844        |
|     85     |      495.786       |         604.884        |
|     90     |       602.35       |         715.232        |
|     95     |      820.454       |         1008.799       |
|    100     |       9999.0       |         4102.47        |
+------------+--------------------+------------------------+
```

Not approved projects have more cost

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [139]:

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




```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                1149   1399
1                                             1                 646    760
2                                             2                 409    486
3                                             3                 288    335
4                                             4                 186    221

        Avg
0  0.821301
1  0.850000
2  0.841564
3  0.859701
4  0.841629
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved  total  \
13                                            13                  44     51
```

```
16                                    16                    36      40
19                                    19                    25      30
17                                    17                    29      30
25                                    25                    24      26

         Avg
13  0.862745
16  0.900000
19  0.833333
17  0.966667
25  0.923077
```

Teachers who have previously posted 0 projects,their projcts also acepted and approved.it means it is not mandatory to post projects previously.

1.2.10 Univariate Analysis: project_resource_summary

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

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

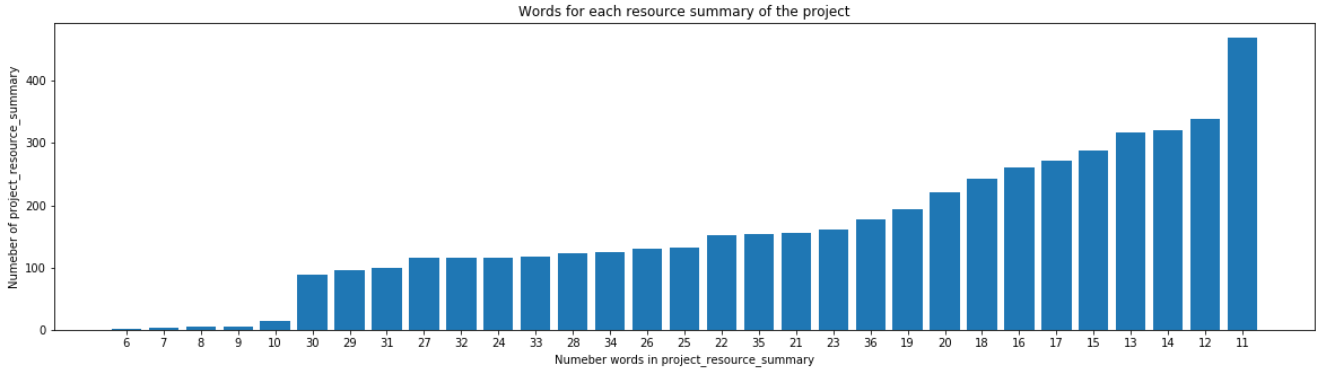
In [140]:

```
print(project_data['project_resource_summary'].values[0])
print('=='*50)
print(project_data['project_resource_summary'].values[1])
print('=='*50)
print(project_data['project_resource_summary'].values[2])
print('=='*50)
print(project_data['project_resource_summary'].values[3])
print('=='*50)
print(project_data['project_resource_summary'].values[4])
```

```
My students need opportunities to practice beginning reading skills in English at home.
====================================================================================================

My students need a projector to help with viewing educational programs
====================================================================================================

My students need shine guards, athletic socks, Soccer Balls, goalie gloves, and training materials
for the upcoming Soccer season.
====================================================================================================

My students need to engage in Reading and Math in a way that will inspire them with these Mini iPa
ds!
====================================================================================================

My students need hands on practice in mathematics. Having fun and personalized journals and charts
will help them be more involved in our daily Math routines.
```

In [141]:

```
#HOW TO CALCULATE NO OF WORDS IN A STRING
word_count= project_data['project_resource_summary'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))

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

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



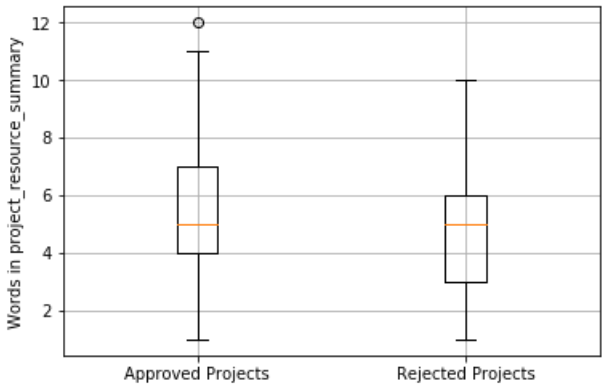
there are 11 no. of words in maximun resource summary

In [142]:

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

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

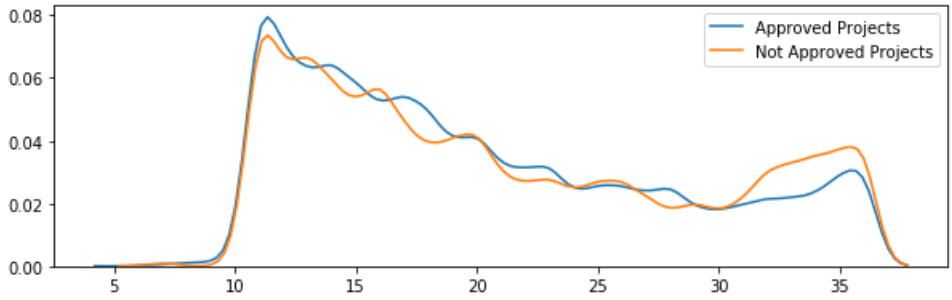
In [143]:

```
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project_resource_summary')
plt.grid()
plt.show()
```



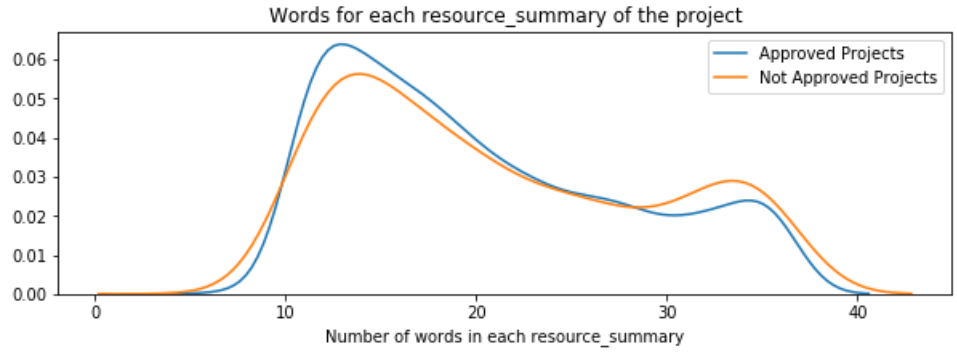
In [144]:

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



In [145]:

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



1.3 Text preprocessing

1.3.1 Essay Text

In [146]:

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

Out[146]:

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

In [149]:

```
# 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[4000])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
```

open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein Our English learner's have a strong support system at home th
at begs for more resources. Many times our parents are learning to read and speak English along s
ide of their children. Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist. All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program. These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch. The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year. The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade. This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of
my own pocket on resources to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==
I teach language arts and social studies to about 50 students each day. I teach two groups of ama
zing kids each day!\r\n\r\nThe students in my classroom range from advanced or gifted learners to
students with various learning disabilities. My school is located in an urban environment in
Maryland. The school is a Title I (low-income) school, and 99% of the students in the school recei
ve free and reduced price lunch. All students at my school receive free breakfast which is the mos
t important meal of the day!High interest reading supports comprehension and learning. I want to e
ncourage a love of reading by choosing books that interest my third grade students. Many of my
students are classified as \"struggling readers\". There is extensive research to support the
premise that the best way to become a better reader is to read more. In order for my students to b
ecome better or more fluent readers I need to increase both the quantity and quality of their read
ing. They need reading materials that they can read and will want to read. \r\n\r\nI want to send
my students into summer vacation with a high interest book. If they find success and interest with
one book, research shows that learning will generate more learning! The book I have chosen is read
able, has a convincing plot, and has realistic characters.nannan
==

In [150]:

```
# 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 [151]:

```
sent = decontracted(project_data['essay'].values[4000])
print(sent)
print("="*50)
```

```
I teach language arts and social studies to about 50 students each day.  I teach two groups of ama
zing kids each day!\r\n\r\nThe students in my classroom range from advanced or gifted learners to
students with various learning disabilities. My school is located in an urban environment in
Maryland. The school is a Title I (low-income) school, and 99% of the students in the school recei
ve free and reduced price lunch. All students at my school receive free breakfast which is the mos
t important meal of the day!High interest reading supports comprehension and learning. I want to e
ncourage a love of reading by choosing books that interest my third grade students. Many of my
students are classified as \"struggling readers\". There is extensive research to support the
premise that the best way to become a better reader is to read more. In order for my students to b
ecome better or more fluent readers I need to increase both the quantity and quality of their read
ing.  They need reading materials that they can read and will want to read. \r\n\r\nI want to send
my students into summer vacation with a high interest book. If they find success and interest with
one book, research shows that learning will generate more learning! The book I have chosen is read
able, has a convincing plot, and has realistic characters.nannan
==================================================
```

In [152]:

```
# \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)
```

```
I teach language arts and social studies to about 50 students each day.  I teach two groups of ama
zing kids each day!    The students in my classroom range from advanced or gifted learners to stud
ents with various learning disabilities. My school is located in an urban environment in Maryland.
The school is a Title I (low-income) school, and 99% of the students in the school receive free an
d reduced price lunch. All students at my school receive free breakfast which is the most
important meal of the day!High interest reading supports comprehension and learning. I want to enc
ourage a love of reading by choosing books that interest my third grade students. Many of my
students are classified as  struggling readers . There is extensive research to support the
premise that the best way to become a better reader is to read more. In order for my students to b
ecome better or more fluent readers I need to increase both the quantity and quality of their read
ing.  They need reading materials that they can read and will want to read.     I want to send my
students into summer vacation with a high interest book. If they find success and interest with on
e book, research shows that learning will generate more learning! The book I have chosen is readab
le, has a convincing plot, and has realistic characters.nannan
```

In [153]:

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

```
I teach language arts and social studies to about 50 students each day I teach two groups of amazi
```

ng kids each day The students in my classroom range from advanced or gifted learners to students w
ith various learning disabilities My school is located in an urban environment in Maryland The sch
ool is a Title I low income school and 99 of the students in the school receive free and reduced p
rice lunch All students at my school receive free breakfast which is the most important meal of th
e day High interest reading supports comprehension and learning I want to encourage a love of read
ing by choosing books that interest my third grade students Many of my students are classified as
struggling readers There is extensive research to support the premise that the best way to become
a better reader is to read more In order for my students to become better or more fluent readers I
need to increase both the quantity and quality of their reading They need reading materials that t
hey can read and will want to read I want to send my students into summer vacation with a high int
erest book If they find success and interest with one book research shows that learning will gener
ate more learning The book I have chosen is readable has a convincing plot and has realistic chara
cters nannan

In [154]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [155]:

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

```
100%|██████████████████████████████████| 5000/5000 [00:05<00:00, 940.40it/s]
```

In [156]:

```
# after preprocesing
preprocessed_essays[4000]
```

Out[156]:

'i teach language arts social studies 50 students day i teach two groups amazing kids day the stud

ents classroom range advanced gifted learners students various learning disabilities my school loc
ated urban environment maryland the school title i low income school 99 students school receive fr
ee reduced price lunch all students school receive free breakfast important meal day high interest
reading supports comprehension learning i want encourage love reading choosing books interest thir
d grade students many students classified struggling readers there extensive research support
premise best way become better reader read in order students become better fluent readers i need i
ncrease quantity quality reading they need reading materials read want read i want send students s
ummer vacation high interest book if find success interest one book research shows learning genera
te learning the book i chosen readable convincing plot realistic characters nannan'

1.3.2 Project title Text

In [157]:

```
# printing titles.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
print(project_data['project_title'].values[15])
print("="*50)
print(project_data['project_title'].values[3])
print("="*50)
print(project_data['project_title'].values[4])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
21st Century learners, 21st century technology!
==================================================
Making Recess Active
==================================================
Techie Kindergarteners
==================================================
Interactive Math Tools
==================================================
```

In [158]:

```
#remopving decontracted words from project titles
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 [159]:

```
sent= decontracted(project_data['project_title'].values[2])
print(sent)
print("="*50)
```

```
Soccer Equipment for AWESOME Middle School Students
==================================================
```

In [160]:

```
# \r \n \t remove from title
sent = sent.replace('\\r', ' ')
```

```
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
Soccer Equipment for AWESOME Middle School Students
```

In [161]:

```
#remove spacial character from title
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Soccer Equipment for AWESOME Middle School Students
```

In [162]:

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

```
100%|██████████████████████████████████| 5000/5000 [00:00<00:00, 11379.62it/s]
```

In [163]:

```
#after preprocessing
print(preprocessed_project_titles[2])
print("="*50)
```

```
soccer equipment for awesome middle school students
==================================================
```

1. 4 Preparing data for models

In [164]:

```
project_data.columns
```

Out[164]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

we are going to consider

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

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

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

1.4.1 Vectorizing Categorical data

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

In [165]:

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

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

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

In [166]:

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


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

```
['Economics', 'FinancialLiteracy', 'CommunityService', 'ForeignLanguages', 'Extracurricular',
'ParentInvolvement', 'Civics_Government', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'CharacterEducation', 'PerformingArts', 'College_CareerPrep', 'Other',
'TeamSports', 'History_Geography', 'Music', 'ESL', 'EarlyDevelopment', 'Health_LifeScience', 'Gym_
Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (5000, 30)
```

In [167]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer(lowercase=False,binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())

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

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

In [168]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

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

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

```
['12', 'Grades', 'PreK']
Shape of matrix after one hot encodig  (5000, 3)
```

In [169]:

```
#value error np.nan is an invalid document
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is-an-invalid-document/39308809#39308809
from sklearn.feature_extraction.text import CountVectorizer
vectorizer=CountVectorizer(lowercase=False,binary=True)
vectorizer.fit(project_data['teacher_prefix'].values.astype('str'))
print(vectorizer.get_feature_names())

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

```
['Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (5000, 4)
```

In [170]:

```
project_data['preprocessed_project_titles'] = preprocessed_project_titles
```

In [171]:

```
project_data.head()
```

Out[171]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

5 rows × 21 columns

In [172]:

```
x=project_data['preprocessed_project_titles']
```

In [173]:

```
y=project_data['project_is_approved']
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [174]:

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

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

```
Shape of matrix after one hot encodig  (5000, 4373)
```

1.4.2.2 Bag of Words on `project_title`

In [175]:

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

In [176]:

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

In [177]:

```
# 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(x)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (5000, 7)
```

1.4.2.3 TFIDF vectorizer

In [178]:

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

```
Shape of matrix after one hot encodig  (5000, 4373)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [179]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(x)
```

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

```
Shape of matrix after one hot encodig  (5000, 7)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [180]:

```
# 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 preprocessed_essays:
    words.extend(i.split(' '))

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

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

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

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

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

```
Loading Glove Model
```

```
1917495it [13:29, 2367.47it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 795220
the unique words in the coupus 17426
The number of words that are present in both glove vectors and our coupus 16989 ( 97.492 %)
word 2 vec length 16989
```

In [183]:

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

In [182]:

```
# 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%|██████████████████████████████████| 5000/5000 [00:04<00:00, 1120.26it/s]
```

```
5000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [184]:

```
# 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(x): # 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%|██████████████████████████████████| 5000/5000 [00:00<00:00, 13878.71it/s]
```

```
5000
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [185]:

```
# 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 [186]:

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

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

```
100%|██████████████████████████████████████| 5000/5000 [00:28<00:00, 174.54it/s]
```

```
5000
300
```

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

In [187]:

```
# Similarly you can vectorize for title also
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(x)
# 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 [86]:

```
# 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(x): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

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

```
100%|██████████████████████████████████████| 5000/5000 [00:00<00:00, 7283.02it/s]
```

```
5000
300
```

1.4.3 Vectorizing Numerical features

In [188]:

```
# 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 : 302.78363, Standard deviation : 376.3799456048676
```

In [189]:

```
price_standardized
```

Out[189]:

```
array([[-0.39370756],
       [-0.01005269],
       [ 0.56875073],
       ...,
       [-0.49814989],
       [-0.35127703],
       [-0.69826683]])
```

majority of projects costs 298 dollars

1.4.4 Merging all the above features

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

In [190]:

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

```
(5000, 4)
(5000, 30)
(5000, 7)
(5000, 1)
```

In [217]:

```
# 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[217]:

```
(5000, 42)
```

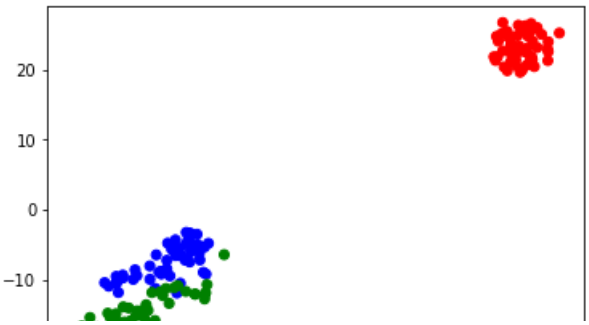
Assignment 2: Apply TSNE

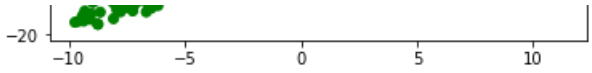
If you are using any code snippet from the internet, you have to provide the reference/citations, as we did in the above cells. Otherwise, it will be treated as plagiarism without citations.

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

In [192]:

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

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

tsne = TSNE(n_components=2, perplexity=30, learning_rate=200, random_state=0)

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

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





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

In [194]:

```
# 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
```

In [218]:

```
from sklearn.manifold import TSNE
X = X.tocsr()
X_new = X[0:5000,:]
```

In [219]:

```
X_new = X_new.toarray()
model = TSNE(n_components = 2, perplexity = 100.0, random_state = 0)
tsne_data_b = model.fit_transform(X_new)
```

In [220]:

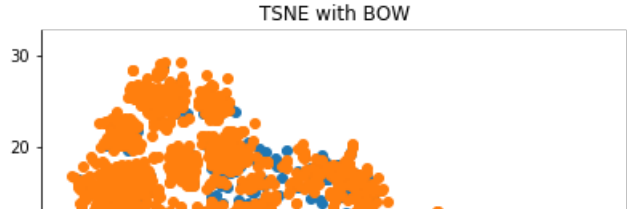
```
labels = project_data["project_is_approved"]
labels_new = labels[0: 5000]
len(labels_new)
```

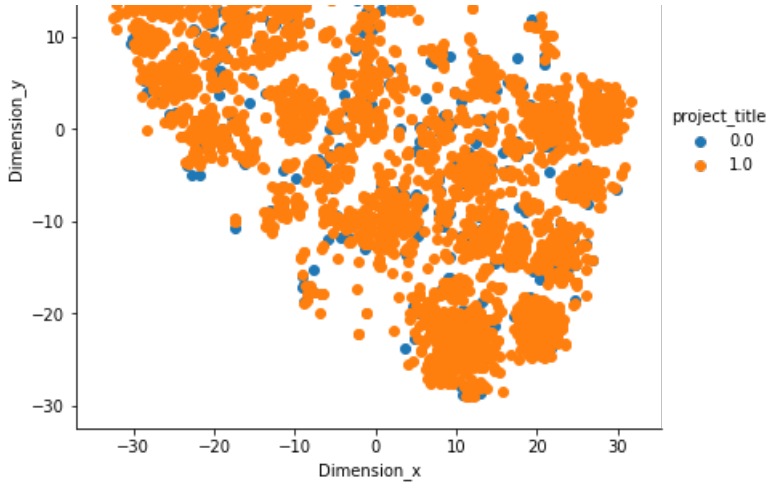
Out[220]:

```
5000
```

In [231]:

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


tsne = TSNE(n_components=2, perplexity=200, learning_rate=200, random_state=0)

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

for_tsne = np.vstack((X_embedding.T, labels_new)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','project_title'])

sns.FacetGrid(for_tsne_df, hue = "project_title", height= 6).map(plt.scatter, "Dimension_x", "Dimen
sion_y").add_legend()
plt.title("TSNE with BOW")
plt.show()
```





conclusion:

As data points are overlapping unable to conclude.

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

In [ ]:

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

In [212]:

```
# 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_tfidf, price_standardized))
X.shape
```

Out[212]:

```
(5000, 42)
```

In [213]:

```
from sklearn.manifold import TSNE
X = X.tocsr()
X_new = X[0:5000,:]
```

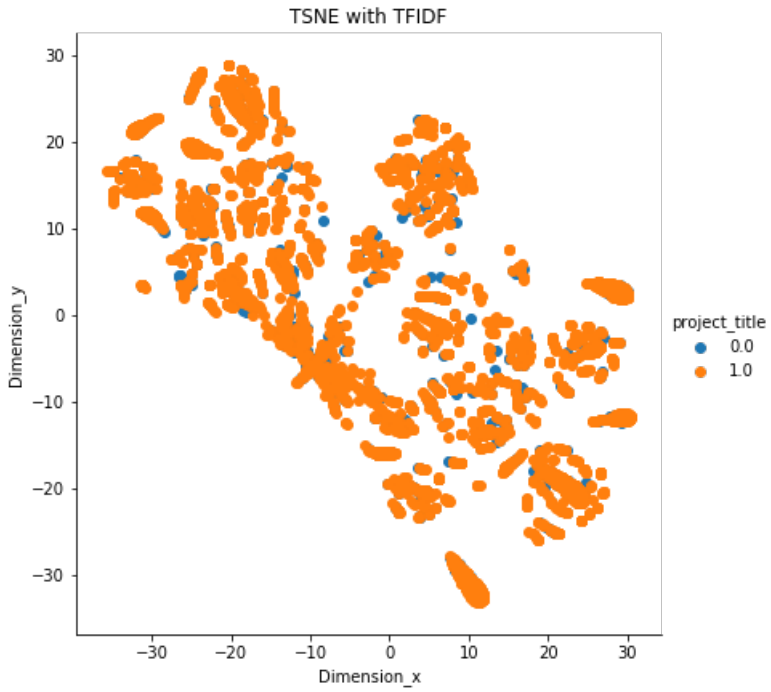
In [214]:

```
X_new = X_new.toarray()
```

In [216]:

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


tsne = TSNE(n_components=2, perplexity=200, learning_rate=200, random_state=0)
```

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

for_tsne = np.vstack((X_embedding.T, labels_new)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','project_title'])

sns.FacetGrid(for_tsne_df, hue = "project_title", height= 6).map(plt.scatter, "Dimension_x", "Dimen
sion_y").add_legend()
plt.title("TSNE with TFIDF")
plt.show()
```



conclusion

in this TFIDF output both the points are noot forming any cluster so it unable to read and conclude

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

In [0]:

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

In [222]:

```
# 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,avg_w2v_vectors, price_standardized))
X.shape
```

Out[222]:

```
(5000, 335)
```

In [223]:

In [223]:

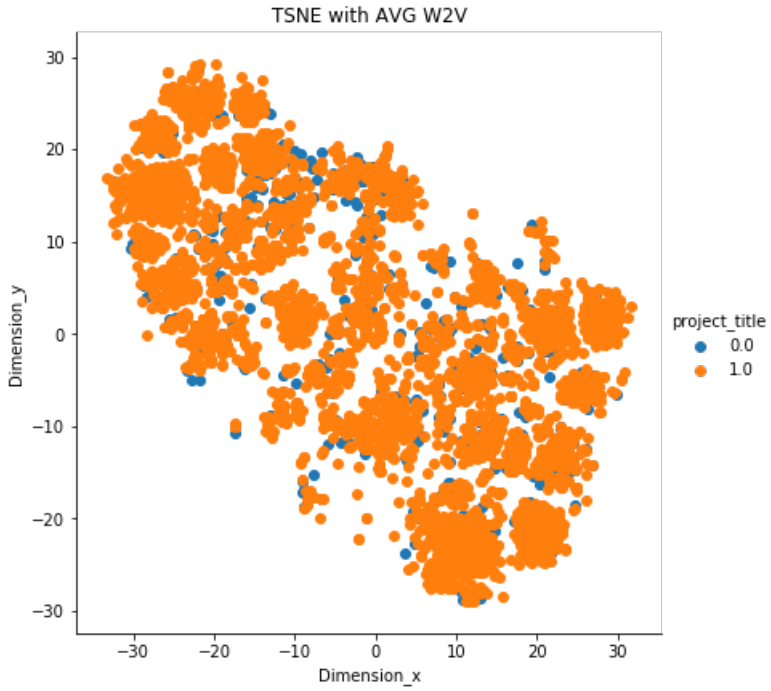
```
from sklearn.manifold import TSNE
X = X.tocsr()
X_new = X[0:5000,:]
```

In [224]:

```
X_new = X_new.toarray()
```

In [232]:

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


tsne = TSNE(n_components=2, perplexity=200, learning_rate=200, random_state=0)

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

for_tsne = np.vstack((X_embedding.T, labels_new)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','project_title'])

sns.FacetGrid(for_tsne_df, hue = "project_title", height= 6).map(plt.scatter, "Dimension_x", "Dimen
sion_y").add_legend()
plt.title("TSNE with AVG W2V")
plt.show()
```



unable to conclude as not forming any cluster

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

In [0]:

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

```
    # c. X-axis label
    # d. Y-axis label
```

In [226]:

```
# 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, tfidf_w2v_vectors, price_standardized))
X.shape
```

Out[226]:

```
(5000, 335)
```

In [227]:

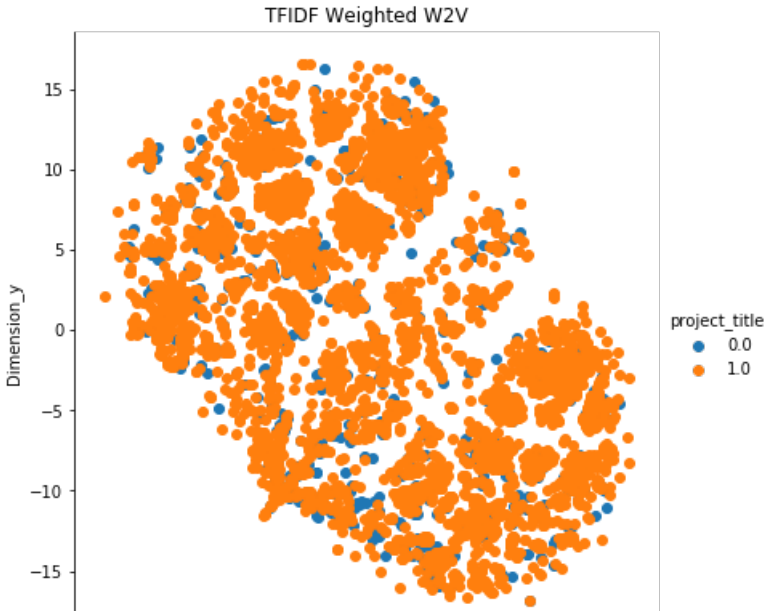
```
from sklearn.manifold import TSNE
X = X.tocsr()
X_new = X[0:5000,:]
```

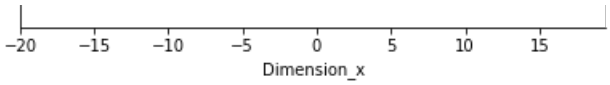
In [228]:

```
X_new = X_new.toarray()
```

In [234]:

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


tsne = TSNE(n_components=2, perplexity=450, learning_rate=200, random_state=0)

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

for_tsne = np.vstack((X_embedding.T, labels_new)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','project_title'])

sns.FacetGrid(for_tsne_df, hue = "project_title", height= 6).map(plt.scatter, "Dimension_x", "Dimen
sion_y").add_legend()
plt.title("TFIDF Weighted W2V")
plt.show()
```





2.5 Summary

In [0]:

```
# Write few sentences about the results that you obtained and the observations you made.
```

unable to conclude as not forming any cluster