In [227]:
```
%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 from train_data set.We are considering 6k observations to ovecome memory issues.

In [228]:
```
project_data1 = pd.read_csv('train_data1.csv')
project_data=project_data1.head(6000)
project_data_O = pd.read_csv('train_data_O.csv')
project_data_Z = pd.read_csv('train_data_Z.csv')
resource_data = pd.read_csv('resources.csv')
```

In [229]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (6000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'scho
ol_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [230]:
```
print("Number of data points in resource data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
# project_data.head(2)
```

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

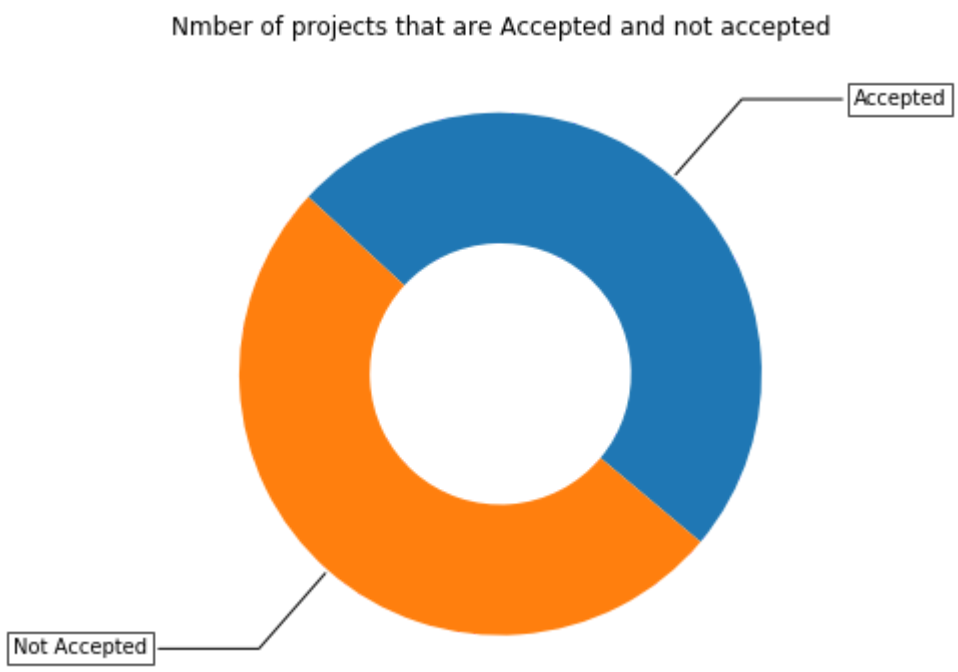
Out[230]:

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

1.2 Data Analysis

In [231]:
```
# 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#s


y_value_counts = project_data['project_is_approved'].value_counts()
print(y_value_counts)
print("Number of projects that are approved for funding ", y_value_counts[1], ",
print("Number of projects that are not approved for funding ", y_value_counts[0],

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()
```

```
0    3044
1    2956
Name: project_is_approved, dtype: int64
Number of projects that are approved for funding  2956 , ( 49.266666666666666
%)
Number of projects that are not approved for funding  3044 , ( 50.7333333333333
3 %)
```



SUMMARY-

We can see that almost equal percentage of project rejections and approvals in the dataset.

1.2.1 Univariate Analysis of School State

In [232]:
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].a
# if you have data which contain only 0 and 1, then the mean = percentage (think

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,

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[232]:
```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n
\nscl (https://datascience.stackexchange.com/a/9620\n\nscl) = [[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)
\']]\n\ndata = [ dict(\n        type=\'choropleth\',\n        colorscale = sc
l,\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 = dic
t(title = "% of pro")\n    ) ]\n\nlayout = dict(\n        title = \'Project Pro
posals % of Acceptance Rate by US States\',\n        geo = dict(\n            s
cope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n
showlakes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n
 ),\n    )\n\nfig = go.Figure(data=data, layout=layout)\noffline.iplot(fig, fil
ename=\'us-map-heat-map\')\n'
```

In [233]:
```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstat
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       0.333333
26         MT       0.363636
18         LA       0.406897
7          DC       0.407407
44         UT       0.427184
==================================================
States with highest % approvals
   state_code  num_proposals
8          DE       0.600000
35         OH       0.604839
29         NE       0.631579
50         WY       0.700000
28         ND       0.857143
```

SUMMARY--From the above output we can note that

1) School state VT has least % of project approvals i.e 33.33%.

2) School state ND has highest % of project approval i.e 85.71%.

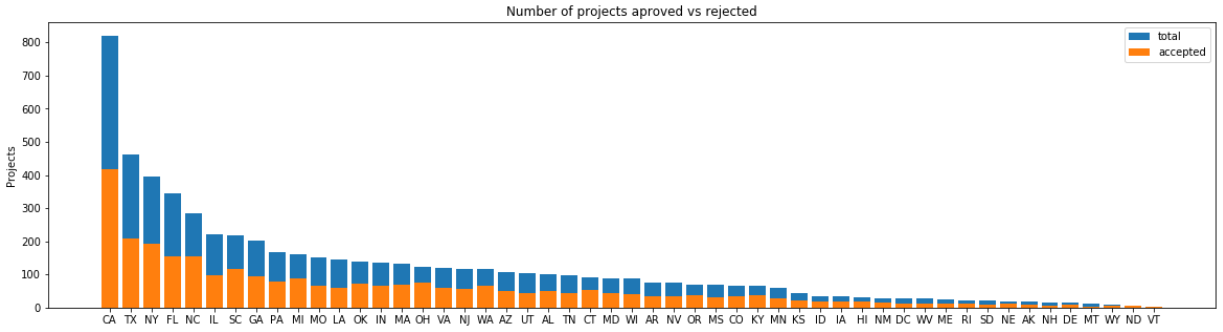
Functions to perform Univariate Analysis

In [234]:
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_mark
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 [235]:
```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/5154
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).su

    # Pandas dataframe grouby count: https://stackoverflow.com/a/19385591/4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'total':'c
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':'mean

    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))
```

Univariate Plots--school_state

In [236]:
```
univariate_barplots(project_data, 'school_state', 'project_is_approved', False)
```



	school_state	project_is_approved	total	Avg
4	CA	416	820	0.507317
43	TX	207	463	0.447084
34	NY	192	394	0.487310
9	FL	155	345	0.449275
27	NC	155	285	0.543860

==========

	school_state	project_is_approved	total	Avg
8	DE	9	15	0.600000
26	MT	4	11	0.363636
50	WY	7	10	0.700000
28	ND	6	7	0.857143
46	VT	1	3	0.333333

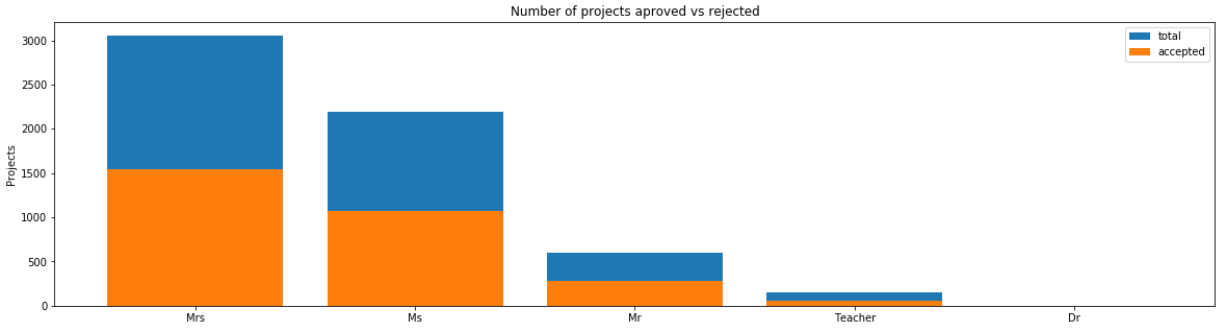
SUMMARY:

1)Top 5 state with highest number of project proposals have greater than 44% project approvals.

2)VT has the least number of project proposals i.e 3.

1.2.2 Univariate Analysis: teacher_prefix

In [237]:
```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=
```



	teacher_prefix	project_is_approved	total	Avg
2	Mrs	1539	3054	0.503929
3	Ms	1077	2188	0.492230
1	Mr	282	602	0.468439
4	Teacher	58	155	0.374194
0	Dr	0	1	0.000000

==========

	teacher_prefix	project_is_approved	total	Avg
2	Mrs	1539	3054	0.503929
3	Ms	1077	2188	0.492230
1	Mr	282	602	0.468439
4	Teacher	58	155	0.374194
0	Dr	0	1	0.000000

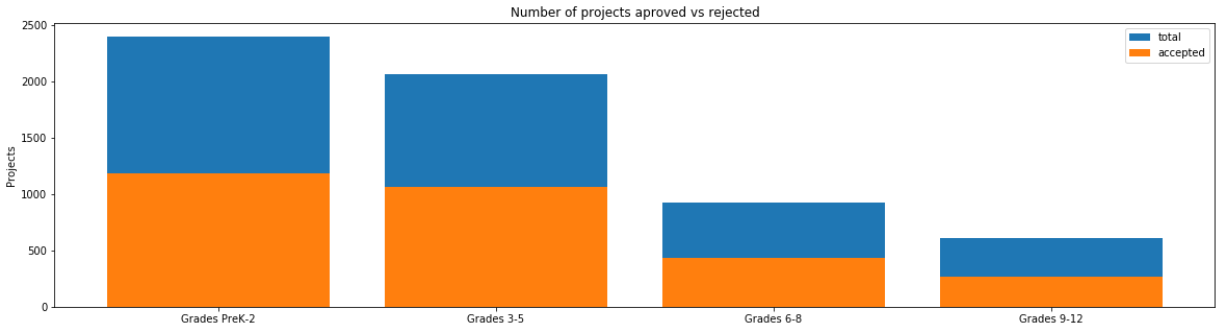
SUMMARY-

1)Teacher prefix "Dr" have the least number of projects proposals and approvals i.e 1 and 0 respectively.

2)Teacher prefix "Mrs" have the most number of projects proposals and approvals among the proposals more than 50% are approved.

1.2.3 Univariate Analysis: project_grade_category

In [238]:
```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved
```




```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                1185   2401  0.493544
0             Grades 3-5                1065   2066  0.515489
1             Grades 6-8                 436    923  0.472373
2            Grades 9-12                 270    610  0.442623
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                1185   2401  0.493544
0             Grades 3-5                1065   2066  0.515489
1             Grades 6-8                 436    923  0.472373
2            Grades 9-12                 270    610  0.442623
```

SUMMARY:

1)Grades above 5 have least number of project proposals and approvals.

2)Grades below 6 have highest number of projects proposals and approvals.

1.2.4 Univariate Analysis: project_subject_categories

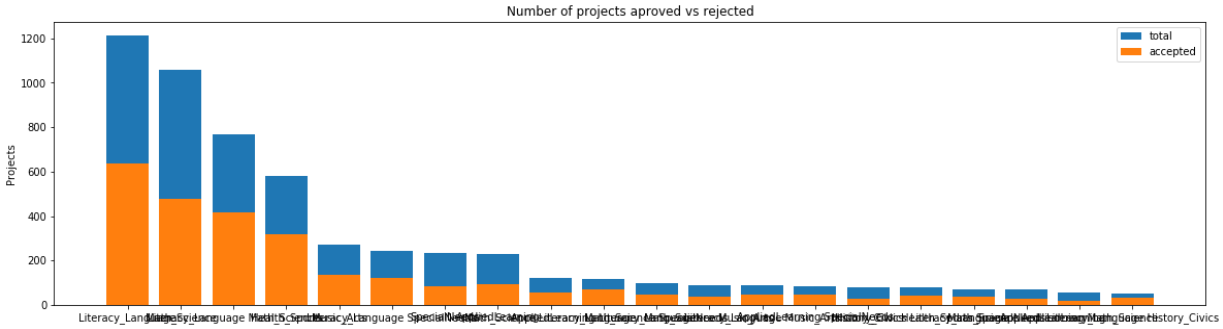
In [239]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

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

In [240]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[240]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	88124	p222195	304ab93dfafcb8d3da141fdd82aa16d0	Mrs	CA	
1	131685	p088154	e910abde42babcf4ba670ece6b860a9d	Mr	NY	

In [241]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=
```




```
                 clean_categories  project_is_approved  total       Avg
21               Literacy_Language                 638   1213  0.525969
29                    Math_Science                 476   1058  0.449905
25  Literacy_Language Math_Science                 415    769  0.539662
7                   Health_Sports                 316    582  0.542955
36                    Music_Arts                  137    270  0.507407
==================================================
                   clean_categories  project_is_approved  total       Avg
17  History_Civics Literacy_Language                   40     77  0.519481
13        Health_Sports SpecialNeeds                   35     68  0.514706
30      Math_Science AppliedLearning                   26     68  0.382353
4       AppliedLearning Math_Science                   18     58  0.310345
24  Literacy_Language History_Civics                   30     49  0.612245
```

SUMMARY:

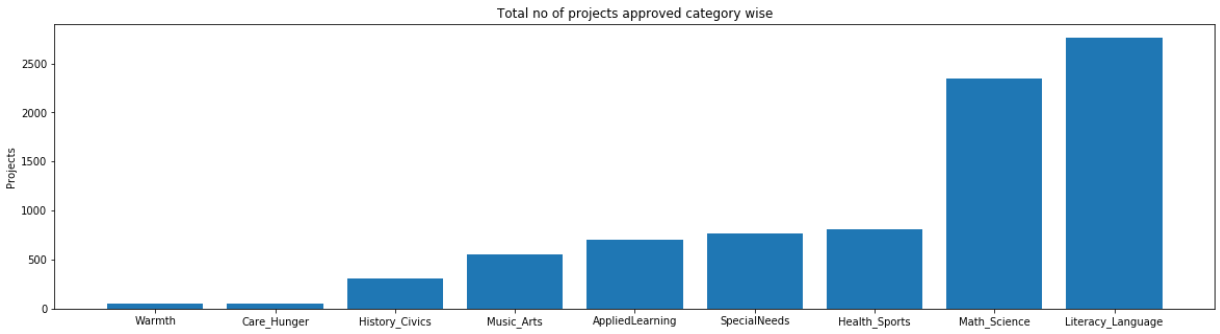
1)Project subject categories "Literacy & Language" has highest number of projects proposals.

2)Project subject categories "Literacy_Language History_Civics" has least number of projects proposals.

In [242]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
print(my_counter)
```

```
Counter({'Literacy_Language': 2762, 'Math_Science': 2347, 'Health_Sports': 806,
'SpecialNeeds': 765, 'AppliedLearning': 698, 'Music_Arts': 552, 'History_Civic
s': 311, 'Warmth': 47, 'Care_Hunger': 47})
```

In [243]:
```
# 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('Total no of projects approved category wise')
plt.xticks(ind, list(sorted_cat_dict.keys()))
plt.show()
```



SUMMARY:

1)Project subject unique category "Literacy_Language" has the highest number of project proposals.

2)Project subject unique category "Warmth" has the least number of project proposals.

In [244]:
```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Warmth               :        47
Care_Hunger          :        47
History_Civics       :       311
Music_Arts           :       552
AppliedLearning      :       698
SpecialNeeds         :       765
Health_Sports        :       806
Math_Science         :      2347
Literacy_Language    :      2762
```

1.2.5 Univariate Analysis: project_subject_subcategories

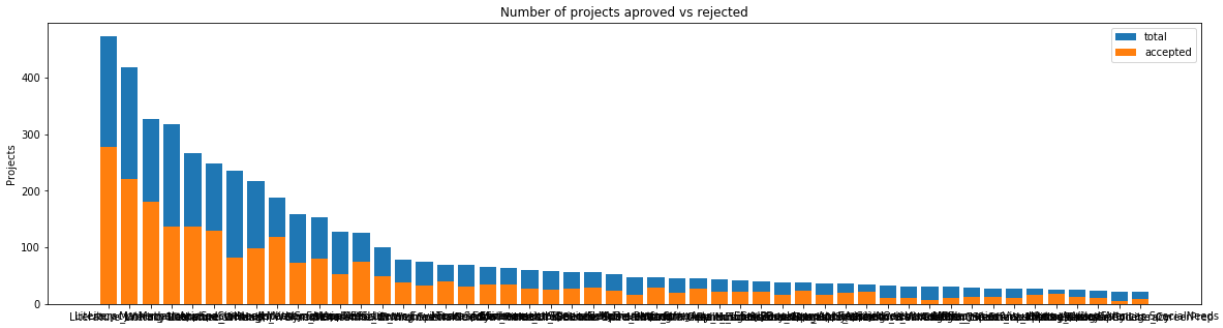
In [245]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

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

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",
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trail
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [246]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[246]:

t_essay_3	project_essay_4	project_resource_summary	teacher_number_of_previously_posted_projects
NaN	NaN	My students need access to tools that will hel...	1
NaN	NaN	My students need a happy learning environment ...	83

In [247]:
```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', 
```



	clean_subcategories	project_is_approved	total	Avg
210	Literacy	277	473	0.585624
212	Literacy Mathematics	221	418	0.528708
223	Literature_Writing Mathematics	181	326	0.555215
231	Mathematics	136	317	0.429022
211	Literacy Literature_Writing	136	267	0.509363

==========

	clean_subcategories	project_is_approved	total	Avg
228	Literature_Writing SocialSciences	18	26	0.692308
220	Literacy VisualArts	13	25	0.520000
202	History_Geography Literacy	10	23	0.434783
3	AppliedSciences College_CareerPrep	6	22	0.272727
18	AppliedSciences SpecialNeeds	9	21	0.428571

SUMMARY:

1)Project subject subcategory "Literacy" has the most number of project proposals.

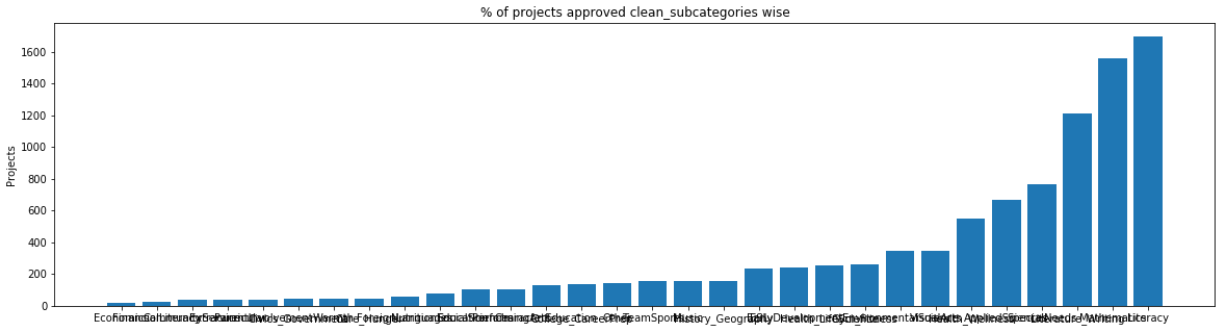
2)Project subject subcategory "AppliedSciences SpecialNeeds" has least number of project proposals.

In [248]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [249]:
```
my_counter
```

Out[249]:
```
Counter({'SpecialNeeds': 765,
         'Warmth': 47,
         'Care_Hunger': 47,
         'Literature_Writing': 1214,
         'Mathematics': 1557,
         'Literacy': 1696,
         'AppliedSciences': 670,
         'EnvironmentalScience': 348,
         'Gym_Fitness': 264,
         'EarlyDevelopment': 244,
         'Other': 143,
         'TeamSports': 154,
         'Civics_Government': 45,
         'PerformingArts': 106,
         'CharacterEducation': 133,
         'ESL': 236,
         'VisualArts': 348,
         'Health_Wellness': 552,
         'History_Geography': 159,
         'Health_LifeScience': 253,
         'College_CareerPrep': 137,
         'Music': 154,
         'Extracurricular': 38,
         'SocialSciences': 103,
         'NutritionEducation': 78,
         'ParentInvolvement': 38,
         'ForeignLanguages': 57,
         'CommunityService': 37,
         'FinancialLiteracy': 23,
         'Economics': 20})
```

In [250]:
```
# 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 approved clean_subcategories wise')
plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
plt.show()
```



SUMMARY:

1)Project subject unique subcategory "Economics" has the least number of project proposals that got approved.

2)Project subject unique subcategory "Literacy" has the most number of project proposals that got approved.

In [251]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

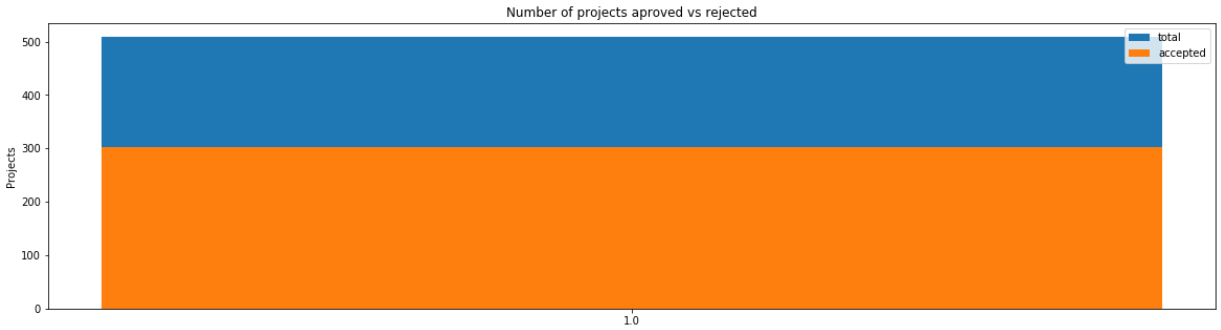
```
Economics            :        20
FinancialLiteracy    :        23
CommunityService     :        37
Extracurricular      :        38
ParentInvolvement    :        38
Civics_Government    :        45
Warmth               :        47
Care_Hunger          :        47
ForeignLanguages     :        57
NutritionEducation   :        78
SocialSciences       :       103
PerformingArts       :       106
CharacterEducation   :       133
College_CareerPrep   :       137
Other                :       143
TeamSports           :       154
Music                :       154
History_Geography    :       159
ESL                  :       236
EarlyDevelopment     :       244
Health_LifeScience   :       253
Gym_Fitness          :       264
EnvironmentalScience :       348
VisualArts           :       348
Health_Wellness      :       552
AppliedSciences      :       670
SpecialNeeds         :       765
Literature_Writing   :      1214
Mathematics          :      1557
Literacy             :      1696
```

Univariate Analysis:digits in summary

In [252]:
```
project_resource_summary = []
new=[]
new1=[]
project_resource_summary = list(project_data['project_resource_summary'].values)
a=0
for i in project_resource_summary:
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(' '):
        if j.isdigit():
            new.append(1)
            new1.insert(a,i)
            print(new1)
            a+=1
            break
        else:
            continue

d_f2=pd.DataFrame({'project_resource_summary':new1,'digits_in_summary':new})
project_data = pd.merge(project_data, d_f2, on='project_resource_summary', how='
project_data.drop_duplicates()
project_data.fillna(0)
```

```
['My students need 2 ipads to help them learn how to read.']
['My students need 2 ipads to help them learn how to read.', 'My students nee
d chair pockets! with 26 students last year, and more coming this year, space
is VERY limited and do not have storage in our classroom that is accessible f
or students  to get materials.']
['My students need 2 ipads to help them learn how to read.', 'My students nee
d chair pockets! with 26 students last year, and more coming this year, space
is VERY limited and do not have storage in our classroom that is accessible f
or students  to get materials.', 'My students need a big 2 sided teaching eas
el that will help benefit in literacy and math stations to help practice with
writing skills.  Both sides are magnetic, so it is interactive and children c
an spell.']
['My students need 2 ipads to help them learn how to read.', 'My students nee
d chair pockets! with 26 students last year, and more coming this year, space
is VERY limited and do not have storage in our classroom that is accessible f
or students  to get materials.', 'My students need a big 2 sided teaching eas
el that will help benefit in literacy and math stations to help practice with
writing skills.  Both sides are magnetic, so it is interactive and children c
```

In [253]:
```
univariate_barplots(project_data, 'digits_in_summary', 'project_is_approved', Fa
```



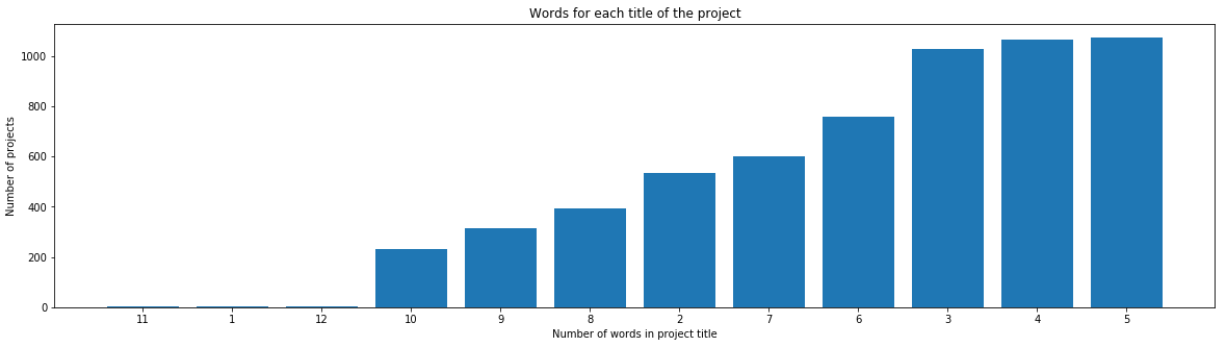

```
   digits_in_summary  project_is_approved  total       Avg
0                1.0                  303    509  0.595285
==================================================
   digits_in_summary  project_is_approved  total       Avg
0                1.0                  303    509  0.595285
```

SUMMARY-

1)We can see that 60% of the project proposals which contained digits in them got approved.

1.2.6 Univariate Analysis: Text features (Title)

In [254]:
```
#How to calculate number of words in a string in DataFrame: https://stackoverflow
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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

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

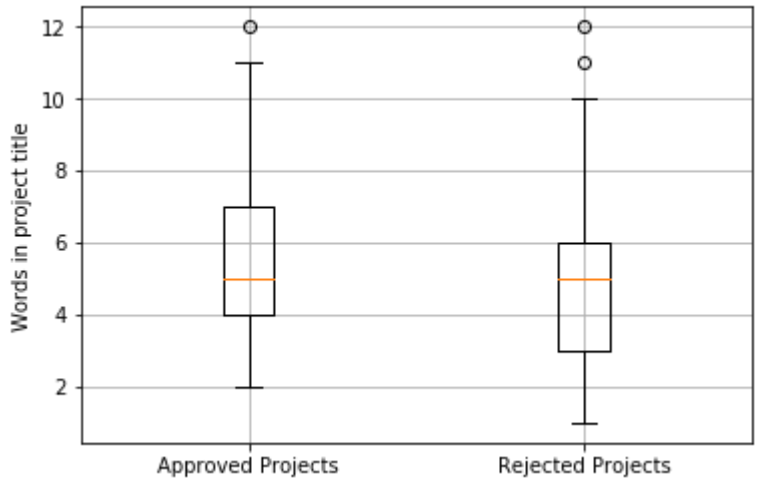


SUMMARY:

1) Majority of the project titles have 4 or 5 words in their project title.

In [255]:
```
approved_title_word_count = project_data[project_data['project_is_approved']==1]
approved_title_word_count = approved_title_word_count.values

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

In [256]:
```
# 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()
```

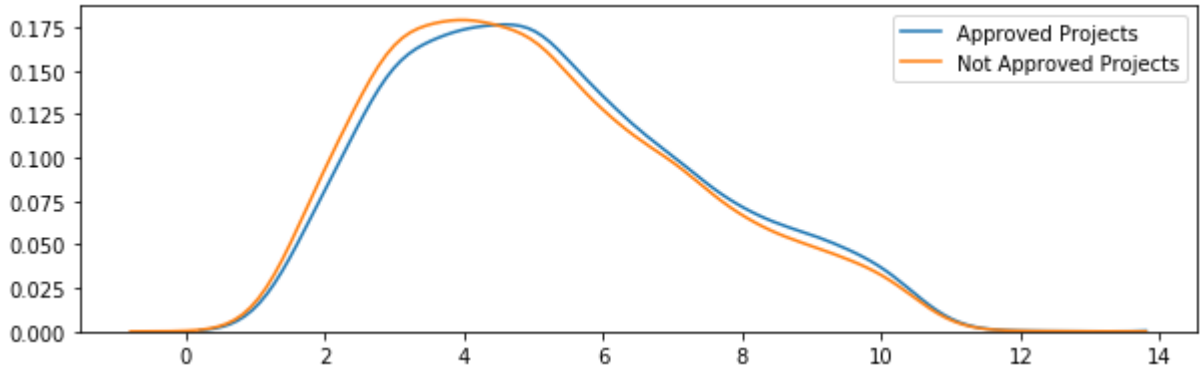


SUMMARY:

1)We can see approx common mean for both approved and rejected projects.

2)We can see that more project titles with >= 5 words got approved as compared to rejected projects.

In [257]:
```
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:

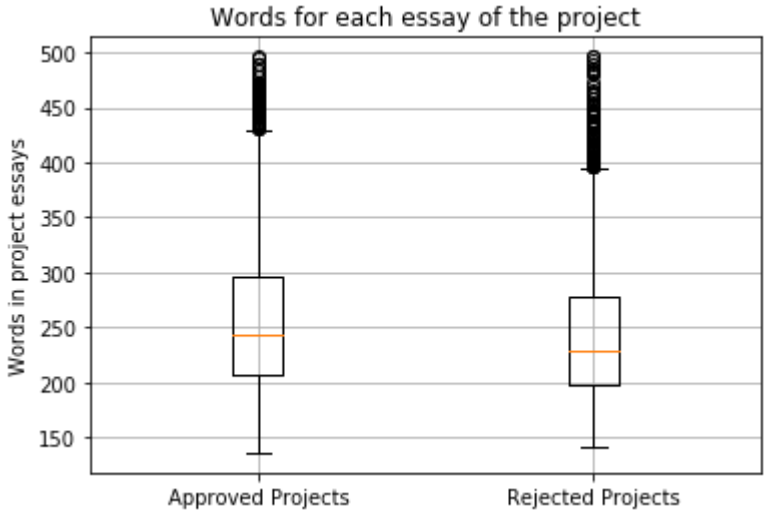
1)Approved projects have more no of words in project title than rejected projects.

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

In [258]:
```
# 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 [259]:
```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay
approved_word_count = approved_word_count.values
rejected_word_count = project_data[project_data['project_is_approved']==0]['essay
rejected_word_count = rejected_word_count.values
```

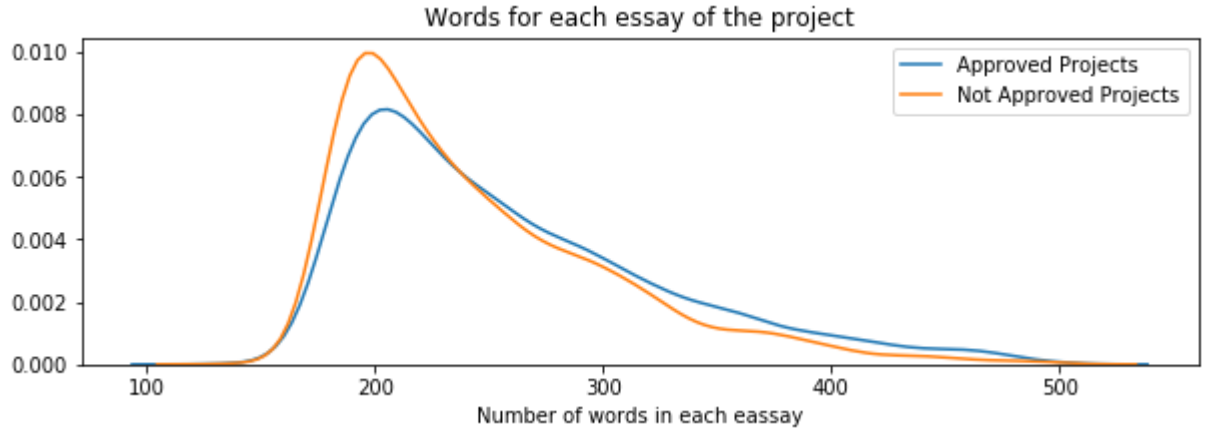
In [260]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()
```



SUMMARY:

1)Approved projects have more number of words in essay as compared to rejected projects.

In [261]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```



SUMMARY:

SUMMARY:
1)Approved projects have more no of words in essay as compared to rejected words.

1.2.8 Univariate Analysis: Cost per project

In [262]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[262]:

	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 [263]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).r
price_data.head(2)
```

Out[263]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [264]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
project_data.head(2)
```

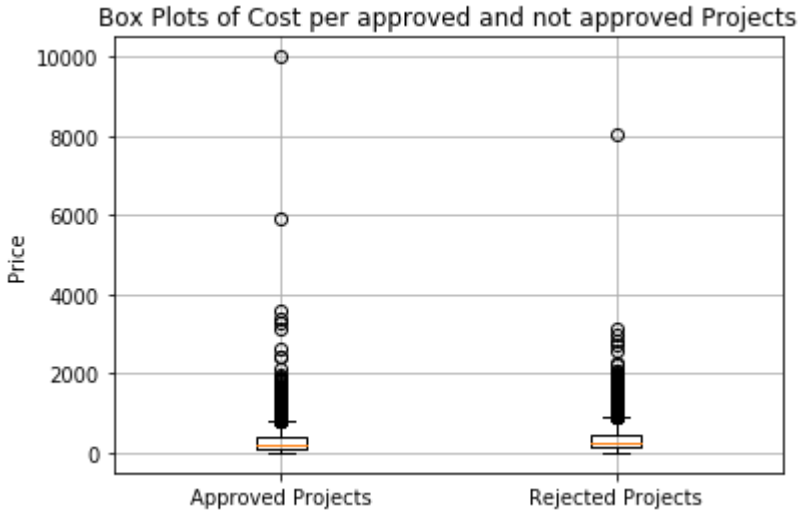
Out[264]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	88124	p222195	304ab93dfafcb8d3da141fdd82aa16d0	Mrs	CA	
1	131685	p088154	e910abde42babcf4ba670ece6b860a9d	Mr	NY	

2 rows × 21 columns

In [265]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].va

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

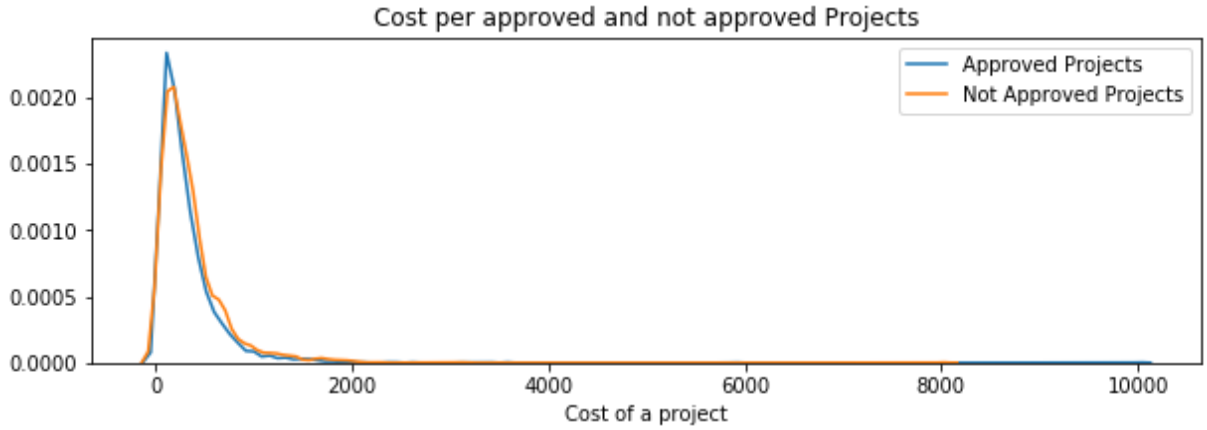
In [266]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Price')
plt.grid()
plt.show()
```



Summary:

Not much can be inferred from the above plot.

In [267]:
```
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:

1)We can see that rejected projects costs slightly higher than the approved projects.

In [268]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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.percer
print(x)
```

```
+------------+------------------+----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+------------------+----------------------+
|     0      |        1.83       |          2.79        |
|     5      |       13.944      |         39.672       |
|     10     |       34.07       |         73.422       |
|     15     |       55.958      |         99.96        |
|     20     |       76.732      |         119.97       |
|     25     |       99.99       |         144.95       |
|     30     |       118.96      |         166.74       |
|     35     |       139.23      |        187.912       |
|     40     |       159.43      |        212.834       |
|     45     |       179.0       |        239.828       |
|     50     |       203.01      |         269.31       |
|     55     |      232.298      |         298.41       |
|     60     |      263.844      |        329.328       |
|     65     |      296.012      |        360.708       |
|     70     |       330.51      |         399.99       |
|     75     |       377.67      |         447.17       |
|     80     |      428.224      |        518.154       |
|     85     |      500.664      |        612.986       |
|     90     |       622.88      |         722.69       |
|     95     |       826.86      |        977.784       |
|    100     |       9999.0      |        8017.53       |
+------------+------------------+----------------------+
```

SUMMARY:

1)From the above table we can infer that the approved projects cost less as compared to rejected projects.

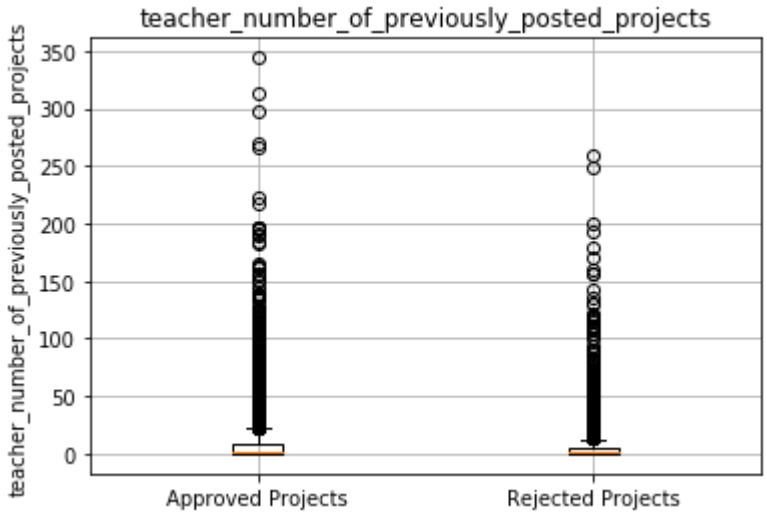
In [269]:
```
print('+++'*40)
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++
```

1.2.9 Univariate Analysis:teacher_number_of_previously_posted_projects

In [270]:
```
# teacher_number_of_previously_posted_projects
approved_teacher_project_count = project_data[project_data['project_is_approved'
# approved_title_word_count
rejected_teacher_project_count = project_data[project_data['project_is_approved'
# rejected_word_count
```

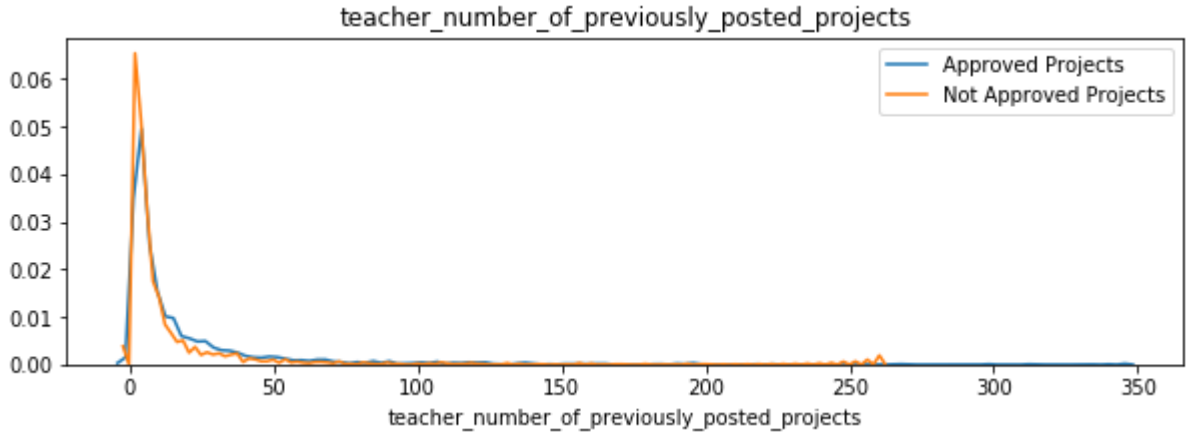
In [271]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_teacher_project_count, rejected_teacher_project_count])
plt.title('teacher_number_of_previously_posted_projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('teacher_number_of_previously_posted_projects')
plt.grid()
plt.show()
```



SUMMARY:

Cant infer much from the above graph.

In [272]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_teacher_project_count, hist=False, label="Approved Projects
sns.distplot(rejected_teacher_project_count, hist=False, label="Not Approved Proj
plt.title('teacher_number_of_previously_posted_projects')
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.legend()
plt.show()
```



SUMMARY:

1)We can say that teachers who made more no of project proposals have slightly higher no of approved projects.

In [273]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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_teacher_project_count,i), 3), n
print(x)
```

```
+------------+------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+------------------+-----------------------+
|     0      |        0.0       |          0.0          |
|     5      |        0.0       |          0.0          |
|     10     |        0.0       |          0.0          |
|     15     |        0.0       |          0.0          |
|     20     |        0.0       |          0.0          |
|     25     |        0.0       |          0.0          |
|     30     |        1.0       |          0.0          |
|     35     |        1.0       |          1.0          |
|     40     |        1.0       |          1.0          |
|     45     |        2.0       |          1.0          |
|     50     |        2.0       |          2.0          |
|     55     |        3.0       |          2.0          |
|     60     |        4.0       |          2.0          |
|     65     |        5.0       |          3.0          |
|     70     |        6.0       |          4.0          |
|     75     |        9.0       |          5.0          |
|     80     |        13.0      |          7.0          |
|     85     |        19.0      |          10.0         |
|     90     |        29.0      |          16.0         |
|     95     |        53.2      |          32.0         |
|    100     |       344.0      |         260.0         |
+------------+------------------+-----------------------+
```

SUMMARY-

1)Teachers who posted more number of projects had higher project approvals as compared to project rejections.

1.3 Text Preprocessing

1.3.1 Essay Text

In [274]:
```
print('++'*55)
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++
```

In [275]:
```
project_data.head(2)
# Sorted_cat_dict.keys()
```

Out[275]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	88124	p222195	304ab93dfafcb8d3da141fdd82aa16d0	Mrs	CA	
1	131685	p088154	e910abde42babcf4ba670ece6b860a9d	Mr	NY	

2 rows × 21 columns

In [276]:
```
# 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[3000])
print("="*50)
print(project_data['essay'].values[4999])
print("="*50)
```

```
Communication is a basic human right, yet my students all struggle to communica
te their basic wants and needs. Students from preschool-transition (age 22) wit
h complex communication needs. Imagine not having a voice. Not being able to sa
y that you are thirsty. Not being able to ask for help. Not being able to say y
ou were happy. That is what my students struggle with. We use a variety of tool
s, ranging from low-tech to high-tech options to help them communicate their ne
eds. \r\n\r\n\"All people with a disability of any extent or severity have a ba
sic right to affect, through communication, the conditions of their existence.
Beyond this general right, a number of specific communication rights should be
ensured in all daily interactions and interventions involving persons who have
severe disabilities.\"\r\n\r\nMy students have complex communication needs, mea
ning that they struggle to make their basic wants and needs known. Imagine not
having a voice. Imagine not being able to say to someone that you were hungry,
that you were thirsty, or that you needed help. That is life is like for my stu
dents. It is my job, as an AT/AAC specialist, to help them communicate those wa
nts, needs and opinions too. \r\nnannan
==================================================
My students love to learn about topics they feel are relevant to their lives.
Cooking is a great way to experience their global world.  They thrive on prepar
ing new foods and because of their high engagement will learn about the various
cultures by experiencing the culinary history of the country.My students are gr
eat!  They are compassionate and considerate and love learning in a small setti
ng.  We are an alternative school with a free and reduced lunch rate of 46%.
Many students are or have overcome obstacles in their lives but continue to str
ive to make their futures better.  I love teaching and interacting with my stud
ents - each has a unique story and view on their future goals and make me proud
to be a teacher in their school.My students will use these materials to learn a
bout cultures of the world, cultural foods and cooking techniques unique to the
various cultures.  We are a project based learning school  which is standard ba
sed, and our students learn best through hands on experiences.  They will learn
preparation techniques through teacher demonstration, videos, and student lab p
ractice.  My students will practice the preparation techniques unique to each c
ulture, experience the unique foods and ingredients of the culture, and learn h
ow and why these cooking techniques, ingredients, etc., are representative of t
his culture and it's history.This project will positively impact my students by
allowing them to experience different cultures through their foods.  During thi
s experience they will learn the backgrounds of the foods from the countries, h
ow to prepare them,  why they are culturally significant as well as the traditi
ons and cultural resources that led to this food being a main staple.
==================================================
My students come to me with great ideas, solutions to problems they see around
their community, and I want to help them accomplish their goals. The school is
part of a very large district but our school is a home and great community. The
fact that we are a Title I school where 90% receive free and reduced lunch does
n't deter their determination. They don't want to be seen as less fortunate. Th
```

ey want to work hard and reach their goals. They walk into my room ready for an y challenge I give them. They work together and make sure no one is falling beh ind. They are my inspiration and I can't wait to get to school to be with them. I want to enter my students into the National Rube Goldberg Machine Contest. Th ey first need to explore simple machines and how they work. I want to allow the m time to work with these hands-on materials so when they enter the contest the y can use their knowledge, creativity, and hard work to build the most complex Rube Goldberg machines their minds can fathom.\r\n\r\n By exploring the simple machines , they can then build more and find how these machines improve their e veryday lives. By seeing engineering as a problem solving tool we will then us e this as a catalyst to exploring other problems my students want to tackle in their community.\r\n\r\nI want them to realize they can use science and enginee ring to solve many of the problems they encounter on a daily basis.nannan

==========

My students come from a title 1 school with very little money to spend on helpi ng the students as people as well as learners. We teach them so much everyday, but they need to have a more inviting environment to learn and be creative. Our school is a STEM school, but teaching with ancient classroom practices. My stud ents need the freedom to learn, but still have the structure needed to teach. P lease help us to fully cross over the next generation of learning to produce ou r future leaders.Students will enjoy learning and working cooperatively to lear n advanced skills necessary for their success in the future. Students are losin g more and more free time during the school day; which leads to preventable beh avior problems. Students can learn and still move around in productive ways tha t do not disrupt their not the other student's learning around them. Students w ill be taught how to chose which seating environment is best for their learnin g. They will begin to take pride and ownership of their learning on a daily/hou rly bases. The most important thing about teaching is making sure that ALL chil dren are learning.nannan

==========

My students are a bunch of creative middle school children who are figuring out how to leave their mark on the world. These specific children have chosen to us e photography as an outlet to share their views.\r\n   In addition, they are a diverse group from various backgrounds and   work incredibly well together de spite coming from different cultures and socioeconomic classes. I am fortunate to be a part of their creative experience and look forward to providing them wi th proper supplies!My project will benefit my photography students by providing them with the proper tools necessary to create, research and edit photo creatio ns. In our technically advanced world today it is necessary to exposed children to the many different programs available to enhance their creative experienc e.\r\n   Creating an area where children could use these laptops at their con venience will be very helpful when they are researching innovative photographer s as well writing their research papers. It is important for me to infuse liter acy into Photography and these laptops will be very helpful. Many students do n ot have their own computers and having this tech center will teach them the ski lls they will need as they progress to high school and college.nannan

==========

In [277]:
```
# 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 [278]:
```
sent = decontracted(project_data['essay'].values[4999])
print(sent)
print("="*50)
```

```
My students are a bunch of creative middle school children who are figuring out
how to leave their mark on the world. These specific children have chosen to us
e photography as an outlet to share their views.\r\n     In addition, they are
a diverse group from various backgrounds and   work incredibly well together de
spite coming from different cultures and socioeconomic classes. I am fortunate
to be a part of their creative experience and look forward to providing them wi
th proper supplies!My project will benefit my photography students by providing
them with the proper tools necessary to create, research and edit photo creatio
ns. In our technically advanced world today it is necessary to exposed children
to the many different programs available to enhance their creative experienc
e.\r\n     Creating an area where children could use these laptops at their con
venience will be very helpful when they are researching innovative photographer
s as well writing their research papers. It is important for me to infuse liter
acy into Photography and these laptops will be very helpful. Many students do n
ot have their own computers and having this tech center will teach them the ski
lls they will need as they progress to high school and college.nannan
==================================================
```

In [279]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-bre
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My students are a bunch of creative middle school children who are figuring out
how to leave their mark on the world. These specific children have chosen to us
e photography as an outlet to share their views.       In addition, they are a
diverse group from various backgrounds and   work incredibly well together desp
ite coming from different cultures and socioeconomic classes. I am fortunate to
be a part of their creative experience and look forward to providing them with
proper supplies!My project will benefit my photography students by providing th
em with the proper tools necessary to create, research and edit photo creation
s. In our technically advanced world today it is necessary to exposed children
to the many different programs available to enhance their creative experience.
Creating an area where children could use these laptops at their convenience wi
ll be very helpful when they are researching innovative photographers as well w
riting their research papers. It is important for me to infuse literacy into Ph
otography and these laptops will be very helpful. Many students do not have the
ir own computers and having this tech center will teach them the skills they wi
ll need as they progress to high school and college.nannan
```

In [280]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My students are a bunch of creative middle school children who are figuring out
how to leave their mark on the world These specific children have chosen to use
photography as an outlet to share their views In addition they are a diverse gr
oup from various backgrounds and work incredibly well together despite coming f
rom different cultures and socioeconomic classes I am fortunate to be a part of
their creative experience and look forward to providing them with proper suppli
es My project will benefit my photography students by providing them with the p
roper tools necessary to create research and edit photo creations In our techni
cally advanced world today it is necessary to exposed children to the many diff
erent programs available to enhance their creative experience Creating an area
where children could use these laptops at their convenience will be very helpfu
l when they are researching innovative photographers as well writing their rese
arch papers It is important for me to infuse literacy into Photography and thes
e laptops will be very helpful Many students do not have their own computers an
d having this tech center will teach them the skills they will need as they pro
gress to high school and college nannan
```

In [281]:
```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itse
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 't
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than'
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should'v
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "d
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn
            'won', "won't", 'wouldn', "wouldn't"]
```

In [282]:
```
# 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%|██████████████████████████████████| 6002/6002 [00:07<00:00, 765.32it/s]
```

In [283]:
```
# after preprocesing
preprocessed_essays[4999]
```

Out[283]:
```
'my students bunch creative middle school children figuring leave mark world th
ese specific children chosen use photography outlet share views in addition div
erse group various backgrounds work incredibly well together despite coming dif
ferent cultures socioeconomic classes i fortunate part creative experience look
forward providing proper supplies my project benefit photography students provi
ding proper tools necessary create research edit photo creations in technically
advanced world today necessary exposed children many different programs availab
le enhance creative experience creating area children could use laptops conveni
ence helpful researching innovative photographers well writing research papers
it important infuse literacy photography laptops helpful many students not comp
uters tech center teach skills need progress high school college nannan'
```

1.3.2 Project title Text

1. 4 Preparing data for models

In [284]:
```
project_data.columns
```

Out[284]:
```
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', 'digits_in_summary', '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

Handling-categorical-and-numerical-features

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

Vectorizing Categorical data: clean_categories(Project subject categories)

In [285]:
```
# 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=F
vectorizer.fit(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
categories_one_hot = vectorizer.transform(project_data['clean_categories'].values
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'S
pecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (6002, 9)
```

Vectorizing Categorical data: clean_subcategories(Project subject subcategories)

In [286]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowerca
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
sub_categories_one_hot = vectorizer.transform(project_data['clean_subcategories'
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'FinancialLiteracy', 'CommunityService', 'Extracurricular', 'Pare
ntInvolvement', 'Civics_Government', 'Warmth', 'Care_Hunger', 'ForeignLanguage
s', 'NutritionEducation', 'SocialSciences', 'PerformingArts', 'CharacterEducati
on', 'College_CareerPrep', 'Other', 'TeamSports', 'Music', 'History_Geography',
'ESL', 'EarlyDevelopment', 'Health_LifeScience', 'Gym_Fitness', 'EnvironmentalS
cience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'L
iterature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (6002, 30)
```

In [287]:
```
# Please do the similar feature encoding with state, teacher_prefix and project_g
type(sub_categories_one_hot)
```

Out[287]:
```
scipy.sparse.csr.csr_matrix
```

Vectorizing Categorical data: school_state

In [288]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())
school_state_dict = dict(my_counter)
sorted_school_state_dict = dict(sorted(school_state_dict.items(), key=lambda kv:
# sorted_school_state_dict
```

In [289]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lo
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())
school_state_one_hot = vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encodig ",school_state_one_hot.shape)
```

```
['VT', 'ND', 'WY', 'MT', 'DE', 'NH', 'AK', 'NE', 'RI', 'SD', 'ME', 'NM', 'WV',
'DC', 'HI', 'IA', 'ID', 'KS', 'MN', 'KY', 'CO', 'MS', 'OR', 'NV', 'AR', 'WI',
'MD', 'CT', 'TN', 'AL', 'UT', 'AZ', 'NJ', 'WA', 'VA', 'OH', 'MA', 'IN', 'OK',
'LA', 'MO', 'MI', 'PA', 'GA', 'SC', 'IL', 'NC', 'FL', 'NY', 'TX', 'CA']
Shape of matrix after one hot encodig  (6002, 51)
```

Vectorizing Categorical data: project_grade_category

In [290]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.split())
project_grade_dict = dict(my_counter)
sorted_project_grade_dict = dict(sorted(project_grade_dict.items(), key=lambda ky
```

In [291]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_dict.keys()), l
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())
project_grade_category_one_hot = vectorizer.transform(project_data['project_grade
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot.sha
```

```
['9-12', '6-8', '3-5', 'PreK-2', 'Grades']
Shape of matrix after one hot encodig  (6002, 5)
```

Vectorizing Categorical data: teacher_prefix

In [292]:
```
#To overcome the blanks in the teacher_prefix categry the .fillna is used
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna("")
# project_data1=project_data.dropna()
```

In [293]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
my_counter1=[]
# project_data['teacher_prefix']=str(project_data['teacher_prefix'])
for word in project_data['teacher_prefix'].values:
    my_counter.update(word.split())
teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda
# teacher_prefix_dict
```

In [294]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()),
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())
teacher_prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].valu
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
```

```
['Dr', 'Teacher', 'Mr', 'Ms', 'Mrs']
Shape of matrix after one hot encodig  (6002, 5)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words:Essays

In [295]:
```
# We are considering only the words which appeared in at least 10 documents(rows
vectorizer = CountVectorizer(min_df=10,max_features=1000)
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  (6002, 1000)
```

1.4.2.1 Bag of words:Project Title

In [296]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    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_project_title.append(sent.lower().strip())
```

```
100%|████████████████████████████████| 6002/6002 [00:00<00:00, 14289.66it/s]
```

In [297]:
```
preprocessed_project_title[4999]
```

Out[297]:
```
'create edit print'
```

In [298]:
```
# We are considering only the words which appeared in at least 10 documents(rows
vectorizer = CountVectorizer(min_df=10,max_features=1000)
project_title_bow = vectorizer.fit_transform(preprocessed_project_title)
print("Shape of matrix after one hot encodig ",project_title_bow.shape)
```

```
Shape of matrix after one hot encodig  (6002, 431)
```

1.4.2.3 TFIDF vectorizer:Essay

In [299]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,max_features=1000)
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  (6002, 1000)
```

TFIDF vectorizer:Project Title

In [300]:
```
# We are considering only the words which appeared in at least 10 documents(rows
vectorizer = TfidfVectorizer(min_df=10,max_features=1000)
project_title_Tfidf = vectorizer.fit_transform(preprocessed_project_title)
print("Shape of matrix after one hot encodig ",project_title_Tfidf.shape)
```

```
Shape of matrix after one hot encodig  (6002, 431)
```

1.4.2.5 Using Pretrained Models: Avg W2V-Essays

In [301]:

```
"""# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084
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 preprocessed_project_title:
    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-t

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

"""""
```

Out[301]:
```
'"# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\n#Output:\n    \n#Loading Glove Model\n#1917495it [06:32, 4879.69it/s]\n#Done. 1917495  words loaded!\n\n# ============================\n\nwords = []\nfor i in preprocessed_essays:\n    words.extend(i.split(\' \'))\n\nfor i in preprocessed_project_title:\n    words.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words that are present in both glove vectors and our coupus",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport (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(words_courpus, f)\n    \n\n'
```

In [302]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to
# make sure you have the glove_vectors file
import pickle
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [303]:
```
# average Word2Vec
# compute average word2vec for each review.
from tqdm import tqdm
avg_w2v_vectors_essay = []; # the avg-w2v for each sentence/review is stored in 
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_essay.append(vector)

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

```
100%|██████████████████████████████████| 6002/6002 [00:04<00:00, 1321.37it/s]

6002
300
```

In [304]:
```
# # We are considering only the words which appeared in at least 10 documents(rou
# vectorizer = CountVectorizer(min_df=10,max_features=1000)
# project_essay_avg_w2v = vectorizer.fit_transform(avg_w2v_vectors_essay)
# print("Shape of matrix after one hot encodig ",project_essay_avg_w2v.shape)


import scipy
avg_w2v_vectors_essay=scipy.sparse.csr_matrix(avg_w2v_vectors_essay)
type(avg_w2v_vectors_essay)
```

Out[304]:
```
scipy.sparse.csr.csr_matrix
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [305]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_Pro_title = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(preprocessed_project_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_Pro_title.append(vector)

print(len(avg_w2v_vectors_Pro_title))
print(len(avg_w2v_vectors_Pro_title[0]))
```

```
100%|██████████████████████████████████| 6002/6002 [00:00<00:00, 22478.12it/s]

6002
300
```

In [306]:
```
import scipy
avg_w2v_vectors_Pro_title=scipy.sparse.csr_matrix(avg_w2v_vectors_Pro_title)
type(avg_w2v_vectors_Pro_title)
```

Out[306]:
```
scipy.sparse.csr.csr_matrix
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V-Essay

In [307]:
```
# 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 [308]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_essay = []; # the avg-w2v for each sentence/review is stored in
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 valu
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay.append(vector)

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

```
100%|██████████████████████████████████| 6002/6002 [00:33<00:00, 177.02it/s]

6002
300
```

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

In [309]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_project_title)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [310]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_Pro_title = []; # the avg-w2v for each sentence/review is store
for sentence in tqdm(preprocessed_project_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf valu
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())
            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_Pro_title.append(vector)

print(len(tfidf_w2v_vectors_Pro_title))
print(len(tfidf_w2v_vectors_Pro_title[0]))
```

```
100%|██████████████████████████████████| 6002/6002 [00:00<00:00, 11431.72it/s]

6002
300
```

In [311]:
```
import scipy
tfidf_w2v_vectors_Pro_title=scipy.sparse.csr_matrix(tfidf_w2v_vectors_Pro_title)
type(tfidf_w2v_vectors_Pro_title)
```

Out[311]:
```
scipy.sparse.csr.csr_matrix
```

1.4.3 Vectorizing Numerical features--Price

In [312]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/skle
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.
# 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
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scala

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

```
Mean : 325.8704381872709, Standard deviation : 364.83632035568274
```

In [313]:
```
price_standardized
```

Out[313]:
```
array([[ 1.8353972 ],
       [-0.50126708],
       [-0.40124963],
       ...,
       [ 0.56745875],
       [ 0.41972126],
       [ 0.92679797]])
```

Vectorizing Numerical features--teacher_number_of_previously_posted_projects

In [314]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/skle
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.
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].va
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scala

# Now standardize the data with above maen and variance.
Teacher_posted_projects_standardized = price_scalar.transform(project_data['teacl
```

```
Mean : 8.955848050649783, Standard deviation : 22.560272486327626
```

In [315]:
```
Teacher_posted_projects_standardized
```

Out[315]:
```
array([[-0.35264858],
       [ 3.28205929],
       [-0.39697429],
       ...,
       [-0.17534576],
       [-0.39697429],
       [ 2.48419659]])
```

Vectorizing Numerical features--digits_in_summary

In [316]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/skle
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.
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['digits_in_summary'].values.reshape(-1,1)) # findi
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scala

# Now standardize the data with above maen and variance.
digits_in_summary_standardized = price_scalar.transform(project_data['digits_in_
digits_in_summary_standardized=np.nan_to_num(digits_in_summary_standardized)
```

```
Mean : 1.0, Standard deviation : 0.0
```

In [317]:
```
digits_in_summary_standardized
```

Out[317]:
```
array([[0.],
       [0.],
       [0.],
       ...,
       [0.],
       [0.],
       [0.]])
```

1.4.4 Merging all the above features

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

In [318]:
```
#catogorical
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(school_state_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(teacher_prefix_one_hot.shape)

#numerical vectors
print(price_standardized.shape)
print(Teacher_posted_projects_standardized.shape)
print(digits_in_summary_standardized.shape)

#text
print(project_title_bow.shape)
print(project_title_Tfidf.shape)
print(avg_w2v_vectors_Pro_title.shape)
print(tfidf_w2v_vectors_Pro_title.shape)
type(digits_in_summary_standardized)
```

```
(6002, 9)
(6002, 30)
(6002, 51)
(6002, 5)
(6002, 5)
(6002, 1)
(6002, 1)
(6002, 1)
(6002, 431)
(6002, 431)
(6002, 300)
(6002, 300)
```

Out[318]: numpy.ndarray

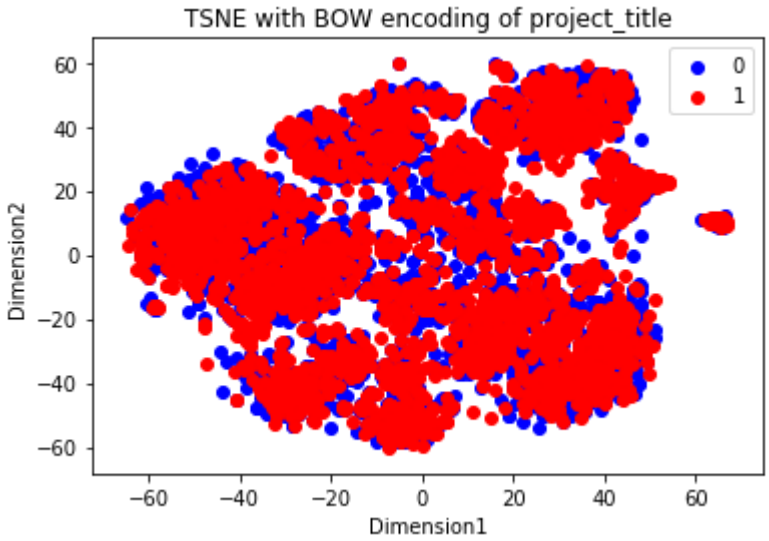
In [319]:
```
# categories_one_hot=categories_one_hot.todense()
# sub_categories_one_hot=sub_categories_one_hot.todense()
# price_standardized=price_standardized.todense()
# text_bow=text_bow.astype('float32')
# text_bow=text_bow.todense()
```

In [320]:
```
# 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
X_bow = hstack(( categories_one_hot,sub_categories_one_hot,school_state_one_hot,p
X_Tfidf = hstack(( categories_one_hot,sub_categories_one_hot,school_state_one_hot
X_avg_w2v_ = hstack(( categories_one_hot,sub_categories_one_hot,school_state_one_
X_tfidf_w2v = hstack(( categories_one_hot,sub_categories_one_hot,school_state_one
X_All = hstack(( categories_one_hot,sub_categories_one_hot,school_state_one_hot,p

X_bow=X_bow.todense()
#To convert into dense vector we use .todense()
X_Tfidf=X_Tfidf.todense()
X_avg_w2v_=X_avg_w2v_.todense()
X_tfidf_w2v=X_tfidf_w2v.todense()
X_All=X_All.todense()

X_bow.shape
X_Tfidf.shape
X_avg_w2v_.shape
X_tfidf_w2v.shape
X_All.shape
```

Out[320]: (6002, 1565)

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

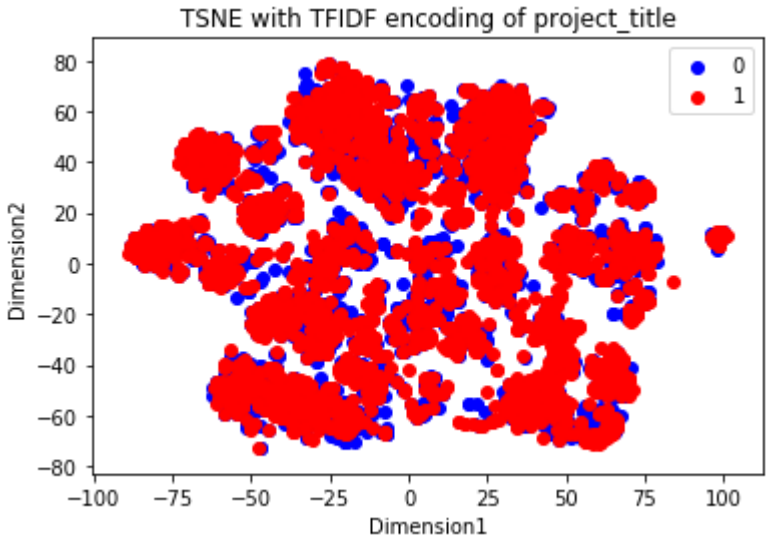
In [321]:
```
from sklearn.manifold import TSNE
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix

tsne = TSNE(n_components=2, perplexity=40, learning_rate=5000)


X_embedding = tsne.fit_transform(X_bow)
y=project_data['project_is_approved']
y=y.values
for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','
colors = {0:'blue', 1:'red'}

#https://stackoverflow.com/questions/47006268/matplotlib-scatter-plot-with-color

scatter_x = for_tsne_df['Dimension_x'].values
scatter_y = for_tsne_df['Dimension_y'].values

group=project_data['project_is_approved'].values
fig, ax = plt.subplots()

for g in np.unique(group):
    ix = np.where(group == g)
    ax.scatter(scatter_x[ix], scatter_y[ix], c=colors[g],label=g)

plt.xlabel('Dimension1')
plt.ylabel('Dimension2')
plt.title("TSNE with BOW encoding of project_title")
ax.legend()
plt.show()
```



From BOW TSNE plot, we can find that the project vectors form recognisible no of clusters post classification and are also overlapping and hence cant be classified easily.

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

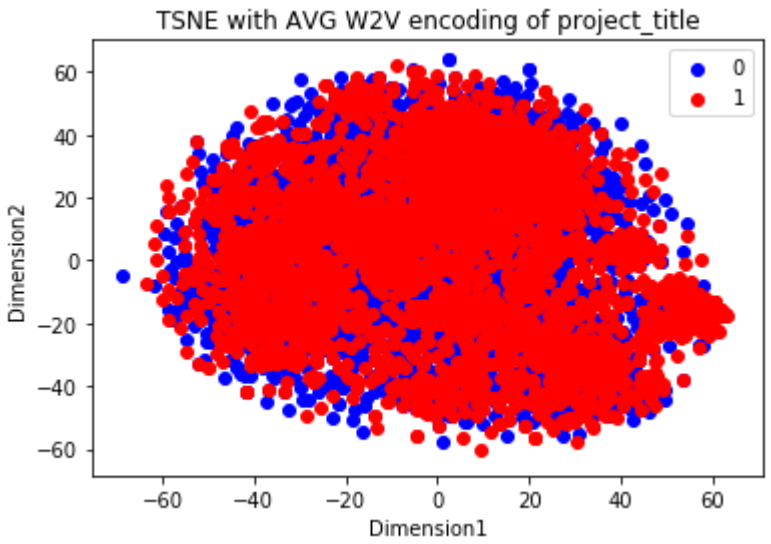
In [322]:
```
from sklearn.manifold import TSNE
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix

tsne = TSNE(n_components=2, perplexity=40, learning_rate=5000)


X_embedding = tsne.fit_transform(X_Tfidf)
y=project_data['project_is_approved']
y=y.values
for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','S
colors = {0:'blue', 1:'red'}

#https://stackoverflow.com/questions/47006268/matplotlib-scatter-plot-with-color

scatter_x = for_tsne_df['Dimension_x'].values
scatter_y = for_tsne_df['Dimension_y'].values

group=project_data['project_is_approved'].values
fig, ax = plt.subplots()

for g in np.unique(group):
    ix = np.where(group == g)
    ax.scatter(scatter_x[ix], scatter_y[ix], c=colors[g],label=g)

plt.xlabel('Dimension1')
plt.ylabel('Dimension2')
plt.title("TSNE with TFIDF encoding of project_title")
ax.legend()
plt.show()
```



From TFIDF TSNE plot, we can find that the project vectors form recognisible no of clusters post classification and are also overlapping and hence cant be classified easily.

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

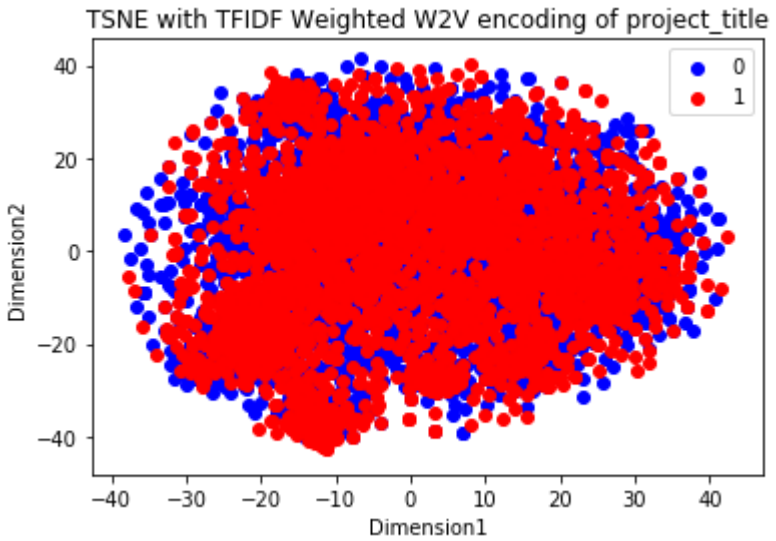
In [323]:
```
from sklearn.manifold import TSNE
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix

tsne = TSNE(n_components=2, perplexity=50, learning_rate=5000)


X_embedding = tsne.fit_transform(X_avg_w2v_)
y=project_data['project_is_approved']
y=y.values
for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','
colors = {0:'blue', 1:'red'}

#https://stackoverflow.com/questions/47006268/matplotlib-scatter-plot-with-color
scatter_x = for_tsne_df['Dimension_x'].values
scatter_y = for_tsne_df['Dimension_y'].values

group=project_data['project_is_approved'].values
fig, ax = plt.subplots()

for g in np.unique(group):
    ix = np.where(group == g)
    ax.scatter(scatter_x[ix], scatter_y[ix], c=colors[g],label=g)

plt.xlabel('Dimension1')
plt.ylabel('Dimension2')
plt.title("TSNE with AVG W2V encoding of project_title")
ax.legend()
plt.show()
```



From AVG W2V TSNE plot, we can find that the project vectors form partial clusters post classification and are also overlapping and hence cant be classified easily.

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

In [324]:
```
from sklearn.manifold import TSNE
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix

tsne = TSNE(n_components=2, perplexity=100, learning_rate=5000)


X_embedding = tsne.fit_transform(X_tfidf_w2v)
y=project_data['project_is_approved']
y=y.values
for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','S
colors = {0:'blue', 1:'red'}

#https://stackoverflow.com/questions/47006268/matplotlib-scatter-plot-with-color
scatter_x = for_tsne_df['Dimension_x'].values
scatter_y = for_tsne_df['Dimension_y'].values

group=project_data['project_is_approved'].values
fig, ax = plt.subplots()

for g in np.unique(group):
    ix = np.where(group == g)
    ax.scatter(scatter_x[ix], scatter_y[ix], c=colors[g],label=g)

plt.xlabel('Dimension1')
plt.ylabel('Dimension2')
plt.title("TSNE with TFIDF Weighted W2V encoding of project_title")
ax.legend()
plt.show()
```

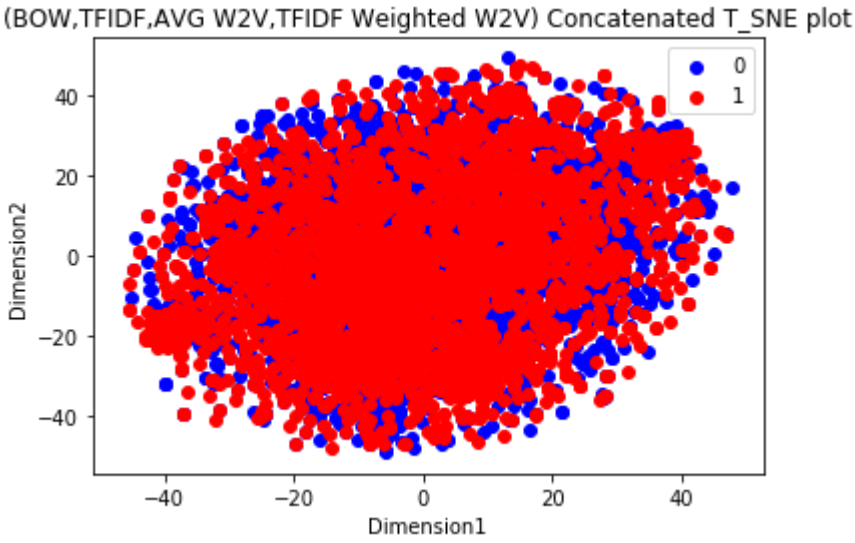


From TFIDF Weighted W2V TSNE plot, we can find that the project vectors form less no of

clusters post classification and are also overlapping and hence cant be classified easily.

2.5 All features(BOW,TFIDF,AVG W2V,TFIDF Weighted W2V) Concatenated and T_SNE applied on Project_title feature

In [325]:
```
from sklearn.manifold import TSNE
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix

tsne = TSNE(n_components=2, perplexity=100, learning_rate=5000)


X_embedding = tsne.fit_transform(X_All)
y=project_data['project_is_approved']
y=y.values
for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','
colors = {0:'blue', 1:'red'}

#https://stackoverflow.com/questions/47006268/matplotlib-scatter-plot-with-color

scatter_x = for_tsne_df['Dimension_x'].values
scatter_y = for_tsne_df['Dimension_y'].values

group=project_data['project_is_approved'].values
fig, ax = plt.subplots()

for g in np.unique(group):
    ix = np.where(group == g)
    ax.scatter(scatter_x[ix], scatter_y[ix], c=colors[g],label=g)

plt.xlabel('Dimension1')
plt.ylabel('Dimension2')
plt.title("(BOW,TFIDF,AVG W2V,TFIDF Weighted W2V) Concatenated T_SNE plot")
ax.legend()
plt.show()
```



From (BOW,TFIDF,AVG W2V,TFIDF Weighted W2V) Concatenated T_SNE plot, we can find that the project vectors form less no of clusters post classification and are also overlapping and hence cant be classified easily.

2.6 Summary

1. Among all the school states, VT has least % of project approvals i.e 33.33%.

2. Among all the school states, ND has highest % of project approvals i.e 85.71%.

3. Top 5 state with highest number of project proposals have greater than 44% project approvals.

4. Among all the school states, VT has the least number of project proposals i.e 3.

5. Among the teacher's prefix category "Dr " has the least number of projects proposals and approvals i.e 1 and 0 respectively.

6. Among the teacher's prefix category "Mrs" has the most number of projects proposals and approvals .Among the proposals more than 50% are approved.

7. The School Grade above 5 have the least number of project proposals and approvals.

8. The School Grade below 6 have the highest number of projects proposals and approvals.

9. Project subject categories "Literacy & Language" has highest number of projects proposals.

10. Project subject categories "Literacy_Language History_Civics" has least number of projects proposals.

11. Project subject unique category "Literacy_Language" has the highest number of project proposals.

12. Project subject unique category "Warmth" has the least number of project proposals.

13. Project subject subcategory "Literacy" has the most number of project proposals.

14. Project subject subcategory "AppliedSciences SpecialNeeds" has least number of project proposals.

15. Project subject unique subcategory "Economics" has the least number of project proposals that got approved.

16. Project subject unique subcategory "Literacy" has the most number of project proposals that got approved.

17. 60% of the project proposals which contained digits in their project resource summary got approved.

18. Majority of the project titles have 4 or 5 words in their project title.

19. Approved projects have more number of words in essay as compared to rejected projects.

20. Rejected projects costs slightly higher than the approved projects.

21. Teachers who made more no of project proposals have slightly higher no of approved projects.

22. From BOW TSNE plot, we can find that the project vectors form recognisible no of clusters post classification and are overlapping and hence cant be classified easily. ¶

23. From TFIDF TSNE plot, we can find that the project vectors form recognisible no of clusters post classification and are also overlapping and hence cant be classified easily.

24. From AVG W2V TSNE plot, we can find that the project vectors form good no of clusters post classification and are also overlapping and hence cant be classified easily.

25. From TFIDF Weighted W2V TSNE plot, we can find that the project vectors form less no of clusters post classification and are also overlapping and hence cant be classified easily.

26. From (BOW,TFIDF,AVG W2V,TFIDF Weighted W2V) Concatenated T_SNE plot, we can find that the project vectors form less no of clusters post classification and are also overlapping and hence cant be classified easily.

In [ ]: