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 [1]:

```
%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
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
```

```
C:\Anaconda\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunki
ze to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [3]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
#Loading top 4000 data points
#project_data = project_data.head(4000)
#resource_data = resource_data.head(4000)
```

In [4]:

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

```
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[4]:

	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

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
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 [5]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[5]:

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

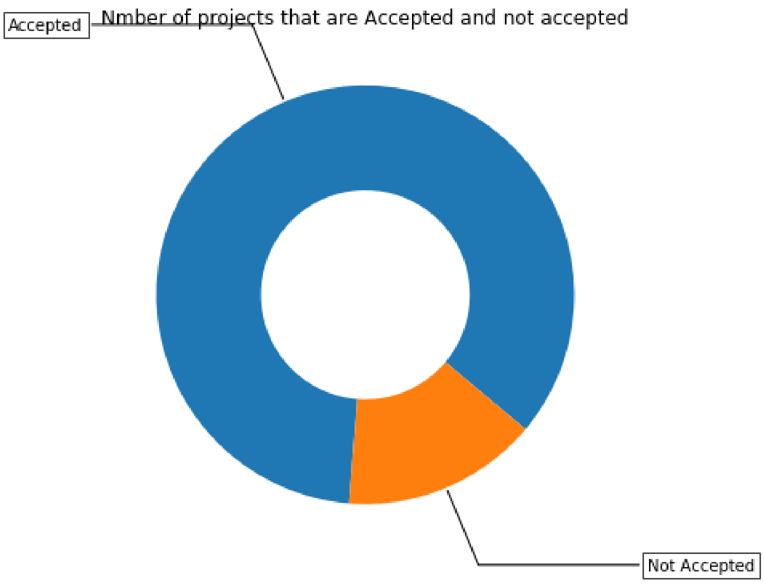
1.2 Data Analysis

In [6]:

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


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

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

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

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

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

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                 horizontalalignment=horizontalalignment, **kw)

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

```
plt.show()
```

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



1.2.1 Univariate Analysis: School State

In [7]:

```
# 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[7]:

```
'# 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 = c
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 [8]:

```
# 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))
print(temp.info())
```

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.800000
7          DC       0.802326
43         TX       0.813142
26         MT       0.816327
18         LA       0.831245
==================================================
States with highest % approvals
   state_code  num_proposals
30         NH       0.873563
35         OH       0.875152
47         WA       0.876178
28         ND       0.888112
8          DE       0.897959
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51 entries, 46 to 8
Data columns (total 2 columns):
state_code       51 non-null object
num_proposals    51 non-null float64
dtypes: float64(1), object(1)
memory usage: 1.2+ KB
None
```

In [9]:

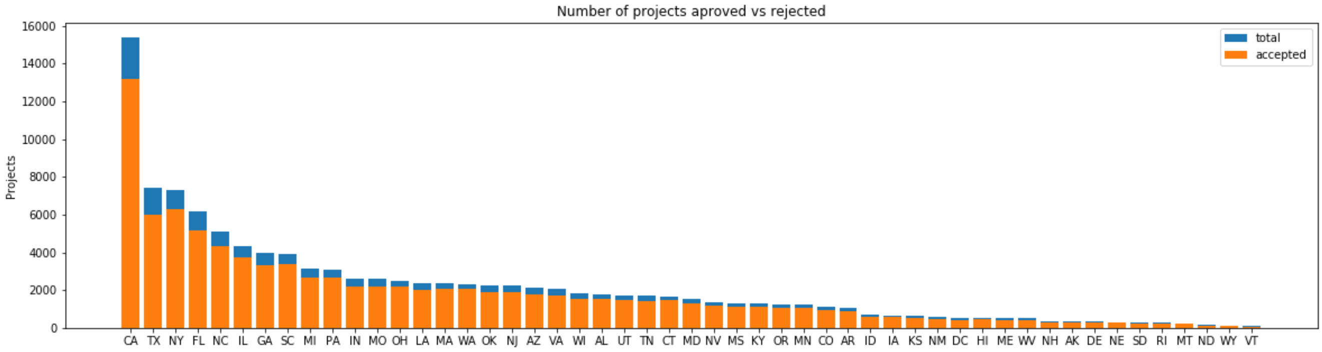
```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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 [10]:

```
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 [11]:

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



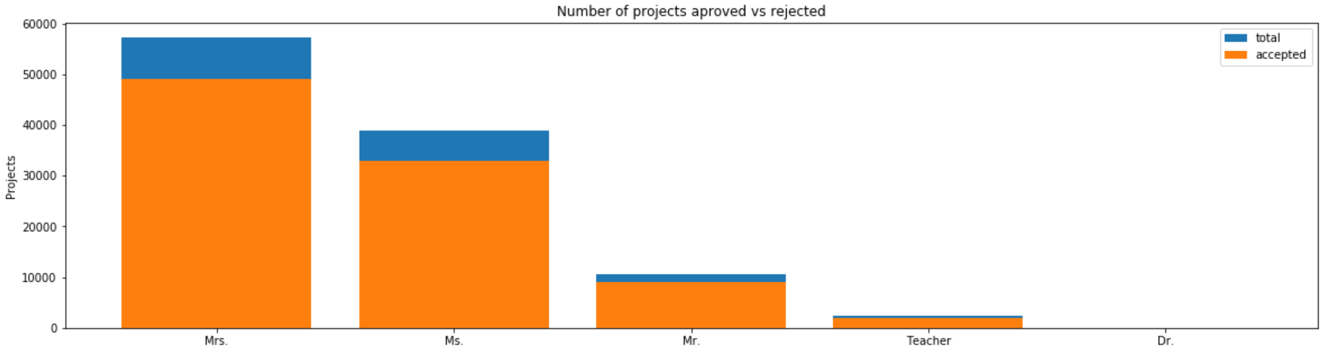
```
   school_state  project_is_approved  total       Avg
4            CA                13205  15388  0.858136
43           TX                 6014   7396  0.813142
34           NY                 6291   7318  0.859661
9            FL                 5144   6185  0.831690
27           NC                 4353   5091  0.855038
==================================================
   school_state  project_is_approved  total       Avg
39           RI                  243    285  0.852632
26           MT                  200    245  0.816327
28           ND                  127    143  0.888112
50           WY                   82     98  0.836735
46           VT                   64     80  0.800000
```

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

1.2.2 Univariate Analysis: teacher_prefix

In [12]:

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



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

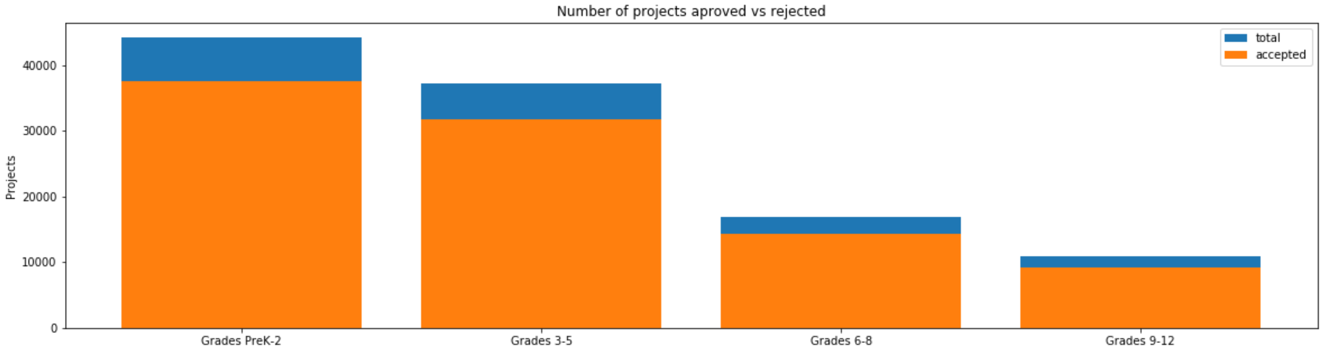
```
4        Teacher                   1877   2360  0.795339
0            Dr.                      9     13  0.692308
==================================================
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.     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

SUMMARY: The teacher_prefix containing Dr. has least success rate and the remaining data contain around 80% success rate in approval

1.2.3 Univariate Analysis: project_grade_category

In [13]:

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



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

Summary: From 109k data we considered, we can see that all the data existing under the column Project grade category has an average success rate greater than 83%.

1.2.4 Univariate Analysis: project_subject_categories

In [14]:

```python
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 &
        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 [15]:

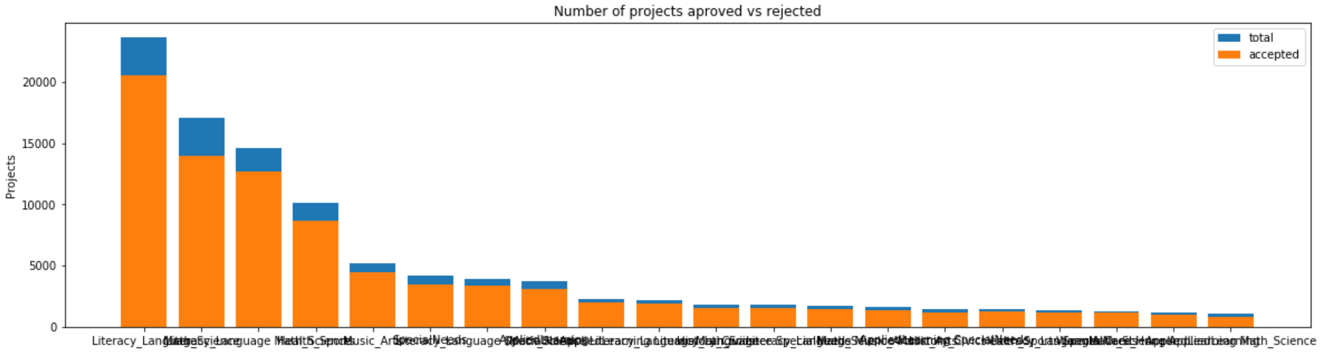
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[15]:

	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 [16]:

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



```
                  clean_categories  project_is_approved  total       Avg
24               Literacy_Language                20520  23655  0.867470
32                    Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12725  14636  0.869432
8                    Health_Sports                 8640  10177  0.848973
40                      Music_Arts                 4429   5180  0.855019
==================================================
                    clean_categories  project_is_approved  total       Avg
19  History_Civics Literacy_Language                 1271   1421  0.894441
14        Health_Sports SpecialNeeds                 1215   1391  0.873472
50                Warmth Care_Hunger                 1212   1309  0.925898
33      Math_Science AppliedLearning                 1019   1220  0.835246
4       AppliedLearning Math_Science                  855   1052  0.812738
```

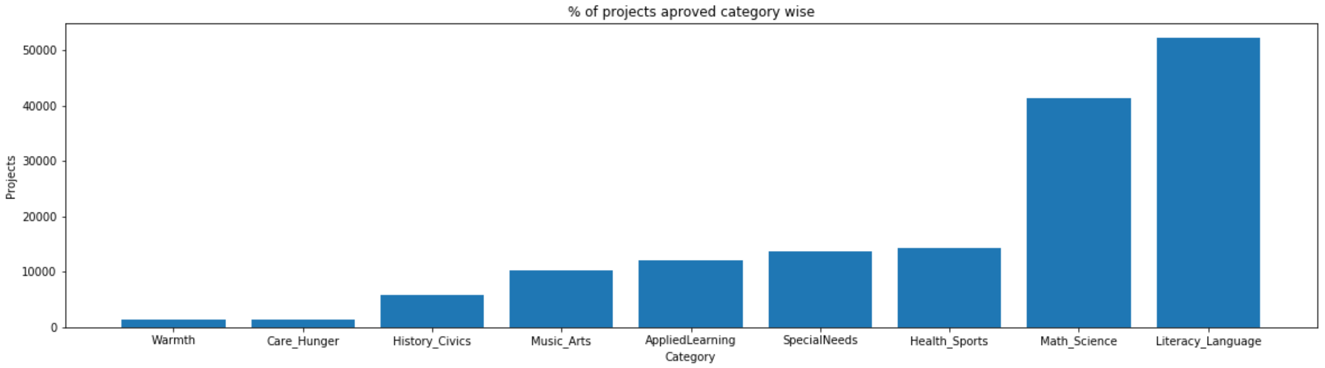
SUMMARY: The project_subject_categories column has been cleaned with neccessary changes and renamed with clean***categories. From the above graph we see that the all combinations of categories have the approval rate around 80%.***

In [17]:

```
# 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 [18]:

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



In [19]:

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

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

SUMMARY: From the data sample we considered, we can analyse that the projects related to Literacy_Language category have highest approval rate while the Warmth category have least approval rate.

1.2.5 Univariate Analysis: project_subject_subcategories

In [20]:

```
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 [21]:

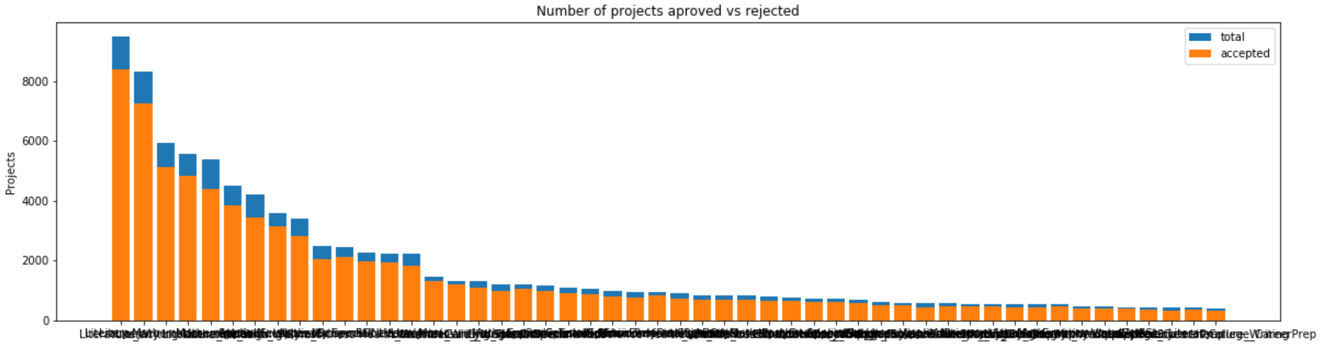
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[21]:

	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 [22]:

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



```
                 clean_subcategories  project_is_approved  total       Avg
317                         Literacy                 8371   9486  0.882458
319             Literacy Mathematics                 7260   8325  0.872072
331  Literature_Writing Mathematics                 5140   5923  0.867803
318     Literacy Literature_Writing                 4823   5571  0.865733
342                      Mathematics                 4385   5379  0.815207
==================================================
                  clean_subcategories  project_is_approved  total       Avg
196       EnvironmentalScience Literacy                  389    444  0.876126
127                                 ESL                  349    421  0.828979
79                   College_CareerPrep                  343    421  0.814727
17     AppliedSciences Literature_Writing                  361    420  0.859524
3      AppliedSciences College_CareerPrep                  330    405  0.814815
```

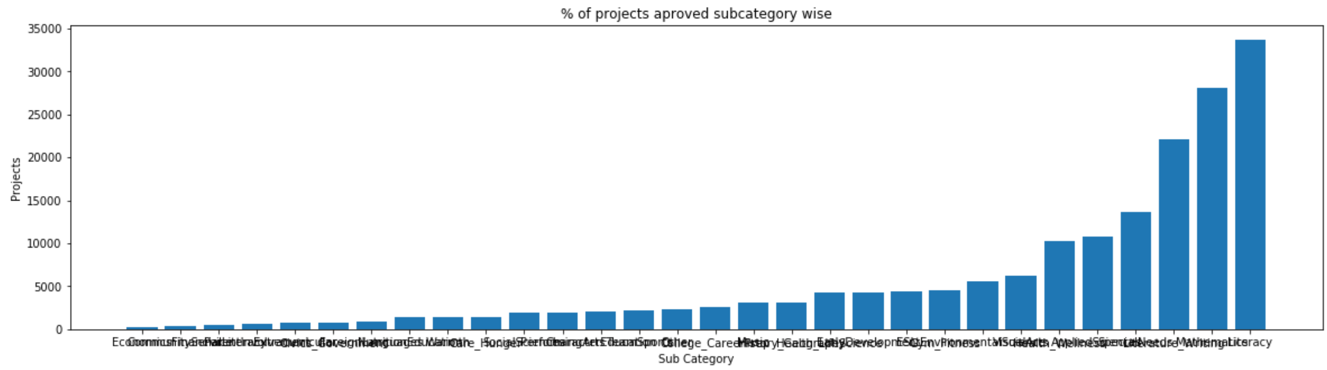
SUMMARY : When we consider subcategories, we can see that projects with all combinations of sub categories have an average approval rate greater than 80%.

In [23]:

```
# 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 [24]:

```
# 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.xlabel('Sub Category')
plt.title('% of projects aproved subcategory wise')
plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
plt.show()
```



In [25]:

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

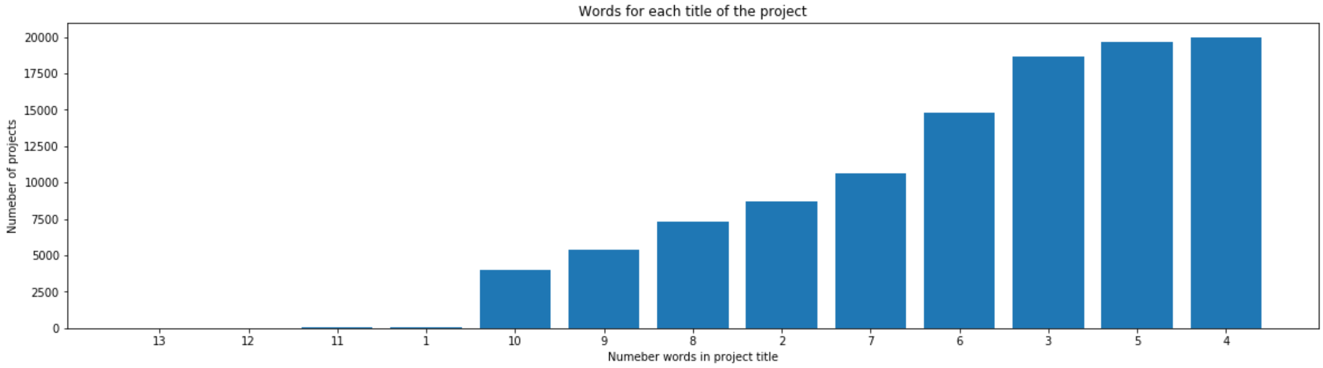
```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
NutritionEducation   :      1355
Warmth               :      1388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

1.2.6 Univariate Analysis: Text features (Title)

In [26]:

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
word_count = project_data['project_title'].str.split().apply(len).value_counts()
```

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


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

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



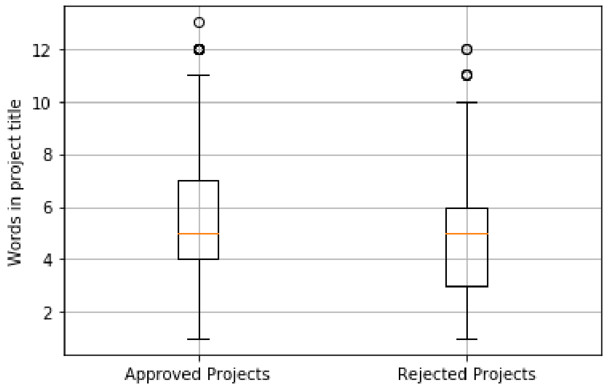
SUMMARY : From the above graph we can see that most of the submitted project titles has a word count of >=3 or we may also say the average word count is around 3-7.

In [27]:

```
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 [28]:

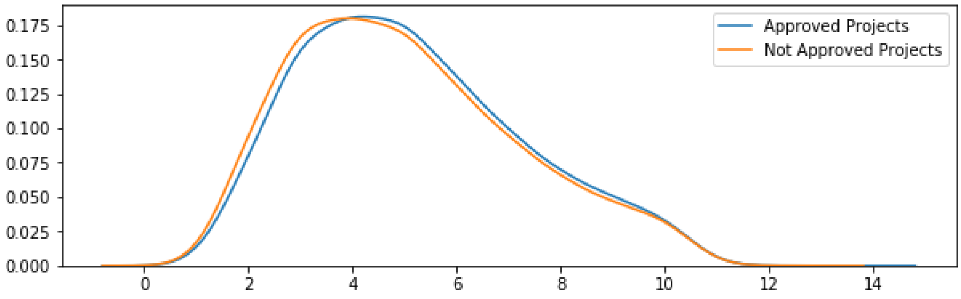
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



In [29]:

```
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 : This above KDE plot give us a detailed view stating that the average word count in the project title is around 4 for both approved and not approved projects.

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

In [30]:

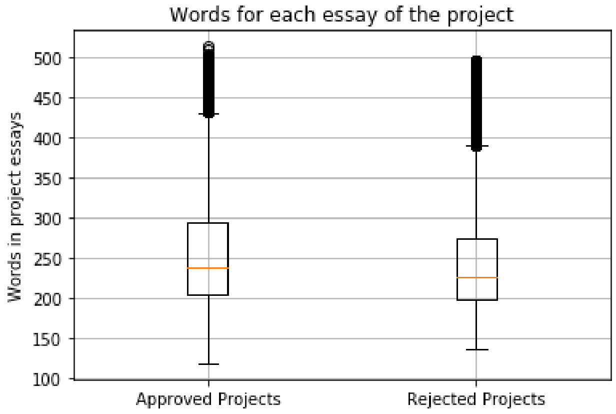
```
# merge two column text dataframe: 
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [31]:

```
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 [32]:

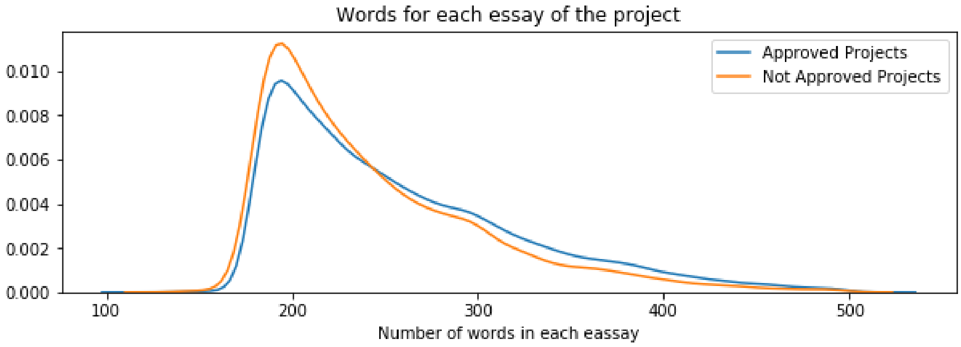
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()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



In [33]:

```
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 : The above two PDF's of the approved and not approved projects has an average word count around 200 in Project essay. Also a nice observation is that the the after the count 250 the project not approved line has been decline with the increase in the word frequency.

1.2.8 Univariate Analysis: Cost per project

In [34]:

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

Out[34]:

	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 [35]:

```
# 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[35]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [36]:

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

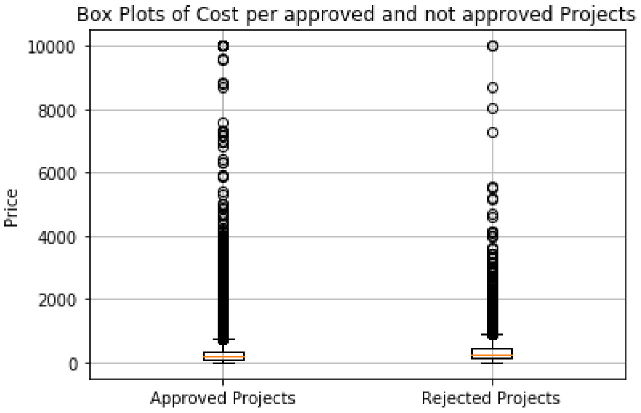
In [37]:

```
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 [38]:

```
# 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 [39]:

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

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

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

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

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

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

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

In [40]:

```
abc = project_data['teacher_number_of_previously_posted_projects']
```

In [41]:

```
abc.describe()
```

Out[41]:

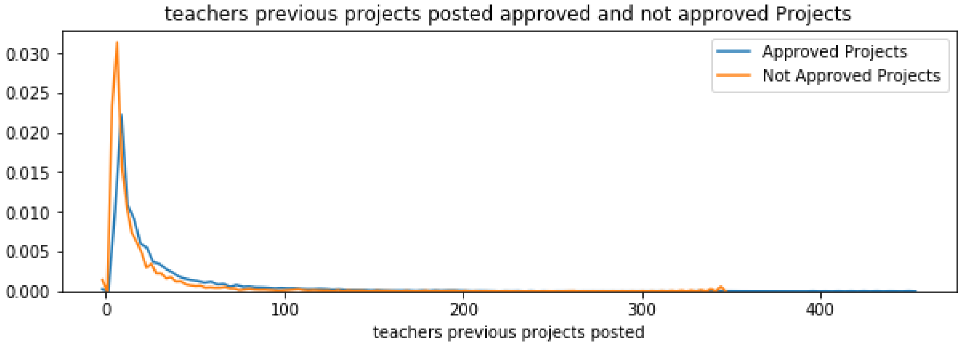
```
count    109248.000000
mean         11.153165
std          27.777154
min           0.000000
25%           0.000000
50%           2.000000
75%           9.000000
max         451.000000
Name: teacher_number_of_previously_posted_projects, dtype: float64
```

In [42]:

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

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

In [43]:

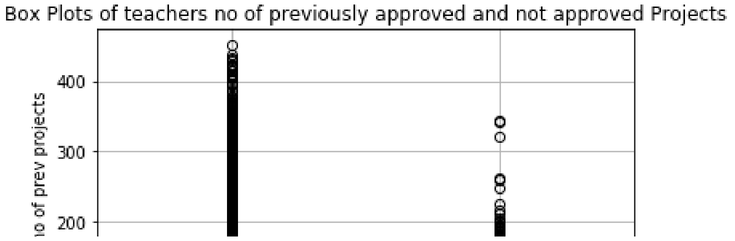
```
plt.figure(figsize=(10,3))
sns.distplot(approved_tppp, hist=False, label="Approved Projects")
sns.distplot(rejected_tppp, hist=False, label="Not Approved Projects")
plt.title('teachers previous projects posted approved and not approved Projects')
plt.xlabel('teachers previous projects posted')
plt.legend()
plt.show()
```

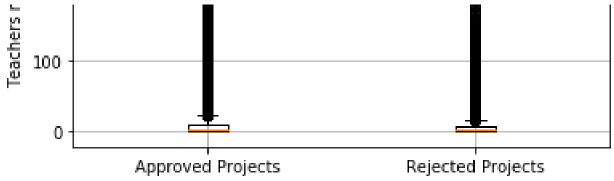


SUMMARY : Considering the above plot teachers having less number of posted projects has a very low chance of approval._

In [44]:

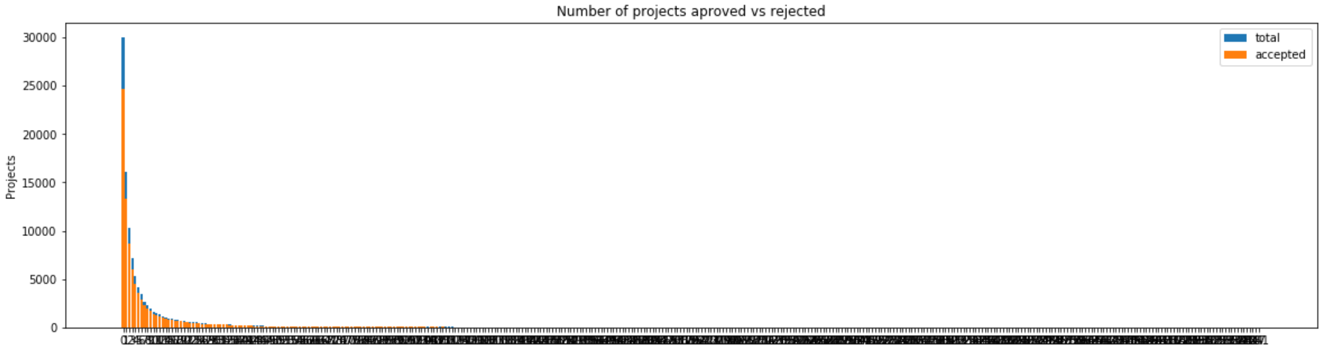
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_tppp, rejected_tppp])
plt.title('Box Plots of teachers no of previously approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Teachers no of prev projects')
plt.grid()
plt.show()
```





In [45]:

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



```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                24652  30014
1                                             1                13329  16058
2                                             2                 8705  10350
3                                             3                 5997   7110
4                                             4                 4452   5266

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

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

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 [46]:

```
import re
```

In [47]:

```
project_data['project_res_sum_digits'] = 2
```

In [48]:

```
project_data['project_res_sum_digits'].tail()
```

Out[48]:

```
109243    2
109244    2
109245    2
109246    2
109247    2
Name: project_res_sum_digits, dtype: int64
```

In [49]:

```
'''
from tqdm import tqdm
count = []
c1 = []
for line in tqdm(range(0,len(project_data['project_resource_summary']))):
    if re.findall('[0-9]',(project_data['project_resource_summary'][line])):
        project_data['project_res_sum_digits'][line] = 1
        count.append(line)
    else:
        project_data['project_res_sum_digits'][line] = 0
        c1.append(line)
'''
```

Out[49]:

```
"\nfrom tqdm import tqdm\ncount = []\nc1 = []\nfor line in
tqdm(range(0,len(project_data['project_resource_summary']))):\n    if re.findall('[0-9]',
(project_data['project_resource_summary'][line])):\n        project_data['project_res_sum_digits'][
line] = 1\n        count.append(line)\n    else:\n        project_data['project_res_sum_digits'][li
ne] = 0\n        c1.append(line)\n"
```

In [50]:

```
project_data['project_res_sum_digits'] = list(map(lambda x : 1 if re.findall('[0-9]',x) else 0,tqdm
(project_data['project_resource_summary'])))
```

```
100%|██████████| 109248/109248 [00:00<00:00, 202103.48it/s]
```

In [51]:

```
from collections import Counter

Counter(project_data['project_res_sum_digits']).keys() # equals to list(set(words))
Counter(project_data['project_res_sum_digits']).values()
```

Out[51]:

```
dict_values([93492, 15756])
```

In [52]:

```
project_data['project_res_sum_digits'].value_counts()
```

Out[52]:

```
0    93492
1    15756
Name: project_res_sum_digits, dtype: int64
```

In [53]:

```
project_data[project_data['project_res_sum_digits']==0]['project_resource_summary'][1]
```

Out[53]:

```
'My students need a projector to help with viewing educational programs'
```

In [54]:

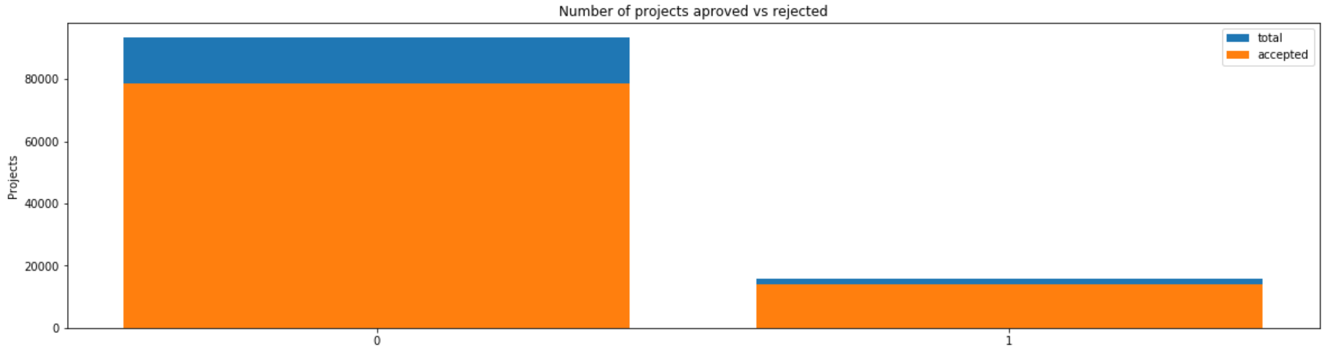
```
project_data['project_res_sum_digits'].unique()
```

Out[54]:

```
array([0, 1], dtype=int64)
```

In [55]:

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



```
   project_res_sum_digits  project_is_approved  total       Avg
0                       0                78616  93492  0.840885
1                       1                14090  15756  0.894263
==================================================
   project_res_sum_digits  project_is_approved  total       Avg
0                       0                78616  93492  0.840885
1                       1                14090  15756  0.894263
```

SUMMARY : In the above count plot, we can see that the projects having digits in summary are in less count but such projects have more approval rate.

In [56]:

```
approved_prs = project_data[project_data['project_is_approved']==1]
['project_res_sum_digits'].values

rejected_prs = project_data[project_data['project_is_approved']==0]
['project_res_sum_digits'].values
```

In [57]:

```
project_data.groupby(['project_is_approved']).count()
```

Out[57]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	pr
project_is_approved								
0	16542	16542	16542	16542	16542	16542	16542	
1	92706	92706	92706	92703	92706	92706	92706	

In [58]:

```
project_data[project_data['project_is_approved']==1]['project_res_sum_digits'].count()
```

Out[58]:

```
92706
```

In [59]:

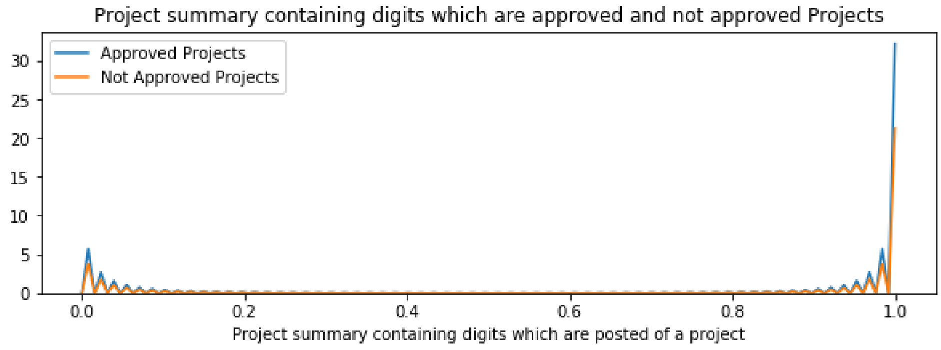
```
len(approved_prs)
```

Out[59]:

```
92706
```

In [60]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_prs, hist=False, label="Approved Projects")
sns.distplot(rejected_prs, hist=False, label="Not Approved Projects")
plt.title('Project summary containing digits which are approved and not approved Projects')
plt.xlabel('Project summary containing digits which are posted of a project')
plt.legend()
plt.show()
```

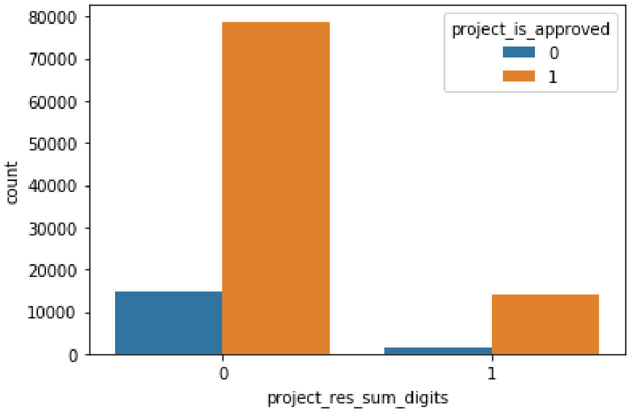


In [61]:

```
sns.countplot(x='project_res_sum_digits',hue='project_is_approved',data=project_data)
```

Out[61]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24004984a20>
```



SUMMARY : Here using the count plot we distingushed about the approval and the not approved projects which contains a digit in project summary.

In [62]:

```
# 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_prs,i), 3), np.round(np.percentile(rejected_prs,i)
, 3)])
print(x)
```

```
+------------+-------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+-------------------+-----------------------+
|     0      |        0.0        |          0.0          |
|     5      |        0.0        |          0.0          |
|     10     |        0.0        |          0.0          |
|     15     |        0.0        |          0.0          |
|     20     |        0.0        |          0.0          |
|     25     |        0.0        |          0.0          |
|     30     |        0.0        |          0.0          |
|     35     |        0.0        |          0.0          |
|     40     |        0.0        |          0.0          |
|     45     |        0.0        |          0.0          |
|     50     |        0.0        |          0.0          |
|     55     |        0.0        |          0.0          |
|     60     |        0.0        |          0.0          |
|     65     |        0.0        |          0.0          |
|     70     |        0.0        |          0.0          |
|     75     |        0.0        |          0.0          |
|     80     |        0.0        |          0.0          |
|     85     |        1.0        |          0.0          |
|     90     |        1.0        |          1.0          |
|     95     |        1.0        |          1.0          |
|    100     |        1.0        |          1.0          |
+------------+-------------------+-----------------------+
```

1.3 Text preprocessing

1.3.1 Essay Text

In [63]:

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

Out[63]:

	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 rows × 21 columns

In [64]:

```
# 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[2000])
```

```python
print("="*50)
print(project_data['essay'].values[3999])
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
==================================================
Describing my students isn't an easy task.  Many would say that they are inspirational, creative,
and hard-working.  They are all unique - unique in their interests, their learning, their
abilities, and so much more.  What they all have in common is their desire to learn each day,
despite difficulties that they encounter.  \r\nOur classroom is amazing - because we understand
that everyone learns at their own pace.  As the teacher, I pride myself in making sure my students
are always engaged, motivated, and inspired to create their own learning! \r\nThis project is to h
elp my students choose seating that is more appropriate for them, developmentally.  Many students
tire of sitting in chairs during lessons, and having different seats available helps to keep them
engaged and learning.\r\nFlexible seating is important in our classroom, as many of our students a
engaged and learning.\r\nFlexible seating is important in our classroom, as many of our students s
truggle with attention, focus, and engagement.  We currently have stability balls for seating, as
well as regular chairs, but these stools will help students who have trouble with balance, or find
it difficult to sit on a stability ball for a long period of time.  We are excited to try these st
ools as a part of our engaging classroom community!nannan
==================================================
I teach a special day class that is filled with awesome students that love learning and coming to
school every day. My students are always seeking a challenge and enjoy the feeling of success when
completing a task or an assignment. I enjoy being an educator which helps shape the minds and
hearts of our future leaders. The learning process is an enjoyable task for my students. My studen
ts enjoy the challenge on a daily basis.\r\n\r\nMy students are happy, motivated, and awesome chil
dren that love doing their best on any given task. Our school is a beautiful campus that has a wel
coming environment for all students, parents, teachers, and staff.My students will use the yoga ba
ll chair as a flexible seating option in the classroom to promote flexible seating. They will use
the chair during small group time and centers throughout the day. The air pump will be used for th
e chair. The treats requested will be used to motivate and reinforce positive behavior in the clas
sroom as well as participation, homework completion, and classwork completion\r\n\r\nThe paint req
uested will be used to make illustrations and drawing of the surroundings to assist in integrating
art and STEAM learning into multiple subject areas such as Science and Mathematics. That will assi
st in integrating art projects connected to multiple subject areas into STEAM.nannan
==================================================
```

In [65]:

```python
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [66]:

```python
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative,
and hard-working.  They are all unique - unique in their interests, their learning, their
abilities, and so much more.  What they all have in common is their desire to learn each day,
despite difficulties that they encounter.  \r\nOur classroom is amazing - because we understand
that everyone learns at their own pace.  As the teacher, I pride myself in making sure my students
are always engaged, motivated, and inspired to create their own learning! \r\nThis project is to h
elp my students choose seating that is more appropriate for them, developmentally.  Many students
tire of sitting in chairs during lessons, and having different seats available helps to keep them
engaged and learning.\r\nFlexible seating is important in our classroom, as many of our students s
truggle with attention, focus, and engagement.  We currently have stability balls for seating, as
well as regular chairs, but these stools will help students who have trouble with balance, or find
it difficult to sit on a stability ball for a long period of time.  We are excited to try these st
ools as a part of our engaging classroom community!nannan
==================================================
```

In [67]:

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

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative,
and hard-working.  They are all unique - unique in their interests, their learning, their
abilities, and so much more.  What they all have in common is their desire to learn each day,
despite difficulties that they encounter.    Our classroom is amazing - because we understand that
everyone learns at their own pace.  As the teacher, I pride myself in making sure my students are
always engaged, motivated, and inspired to create their own learning!   This project is to help my
students choose seating that is more appropriate for them, developmentally.  Many students tire of
sitting in chairs during lessons, and having different seats available helps to keep them engaged
and learning.  Flexible seating is important in our classroom, as many of our students struggle wi
th attention, focus, and engagement.  We currently have stability balls for seating, as well as re
gular chairs, but these stools will help students who have trouble with balance, or find it diffic
ult to sit on a stability ball for a long period of time.  We are excited to try these stools as a
part of our engaging classroom community!nannan
```

In [68]:

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

```
Describing my students is not an easy task Many would say that they are inspirational creative and
hard working They are all unique unique in their interests their learning their abilities and so m
uch more What they all have in common is their desire to learn each day despite difficulties that
they encounter Our classroom is amazing because we understand that everyone learns at their own pa
ce As the teacher I pride myself in making sure my students are always engaged motivated and inspi
red to create their own learning This project is to help my students choose seating that is more a
ppropriate for them developmentally Many students tire of sitting in chairs during lessons and hav
ing different seats available helps to keep them engaged and learning Flexible seating is
important in our classroom as many of our students struggle with attention focus and engagement We
currently have stability balls for seating as well as regular chairs but these stools will help st
udents who have trouble with balance or find it difficult to sit on a stability ball for a long pe
riod of time We are excited to try these stools as a part of our engaging classroom community nann
an
```

In [69]:

```
# 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', "do
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 [70]:

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

```
100%|██████████| 109248/109248 [01:39<00:00, 1092.74it/s]
```

In [71]:

```
# after preprocesing
preprocessed_essays[2000]
```

Out[71]:

```
'describing students not easy task many would say inspirational creative hard working they unique
unique interests learning abilities much what common desire learn day despite difficulties
encounter our classroom amazing understand everyone learns pace as teacher i pride making sure stu
dents always engaged motivated inspired create learning this project help students choose seating
appropriate developmentally many students tire sitting chairs lessons different seats available he
lps keep engaged learning flexible seating important classroom many students struggle attention fo
cus engagement we currently stability balls seating well regular chairs stools help students
trouble balance find difficult sit stability ball long period time we excited try stools part enga
ging classroom community nannan'
```

1.3.2 Project title Text

In [72]:

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

In [73]:

```
pt = project_data['project_title']
```

In [74]:

```
pt.head()
```

Out[74]:

```
0      Educational Support for English Learners at Home
1                 Wanted: Projector for Hungry Learners
2     Soccer Equipment for AWESOME Middle School Stu...
3                                Techie Kindergarteners
4                                Interactive Math Tools
Name: project_title, dtype: object
```

In [75]:

```
pt.nunique()
```

Out[75]:

```
100851
```

In [76]:

```
pt.values[100]
```

Out[76]:

```
'21st Century learners, 21st century technology!'
```

In [77]:

```
# 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 [78]:

```
sent = decontracted(pt.values[2000])
print(sent)
print("="*50)
```

```
Steady Stools for Active Learning
==================================================
```

In [79]:

```
# \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)
```

```
Steady Stools for Active Learning
```

In [80]:

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

```
Steady Stools for Active Learning
```

In [81]:

```
# 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', "dc
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 [82]:

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

```
100%|██████████| 109248/109248 [00:03<00:00, 27994.72it/s]
```

In [83]:

```
preprocessed_titles[2000:2010]
```

Out[83]:

```
['steady stools active learning',
 'classroom supplies',
 'kindergarten students deserve quality books vibrant rug',
 'listen understand',
 'ipads ignite learning',
 'tablets for learning',
 'go p e',
 'making learning fun',
 'empowerment through silk screen designed tee shirts',
 'let play together']
```

1. 4 Preparing data for models

In [84]:

```
project_data.columns
```

Out[84]:

```
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',
       'project_res_sum_digits'],
      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 [85]:

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


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

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

In [86]:

```
# 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', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [87]:

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

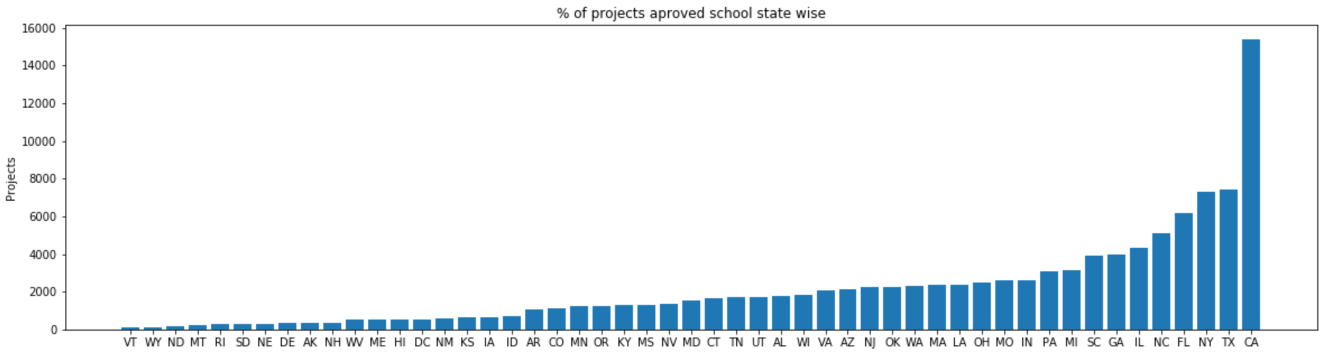
school state

In [88]:

```
# 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['school_state'].values:
    my_counter.update(word.split())
```

In [89]:

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


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

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



In [90]:

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


sub_categories_one_hot_1 = vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_1.shape)
```

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

teacher prefix

In [91]:

```
project_data.groupby(['teacher_prefix'])['teacher_prefix'].count()
```

Out[91]:

```
teacher_prefix
Dr.            13
Mr.         10648
Mrs.        57269
Ms.         38955
Teacher      2360
Name: teacher_prefix, dtype: int64
```

In [92]:

```
project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

Out[92]:

```
7820     NaN
```

```
30368    NaN
57654    NaN
Name: teacher_prefix, dtype: object
```

In [93]:

```
project_data['teacher_prefix'].fillna(project_data['teacher_prefix'].mode()[0],inplace=True)
```

In [94]:

```
project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

Out[94]:

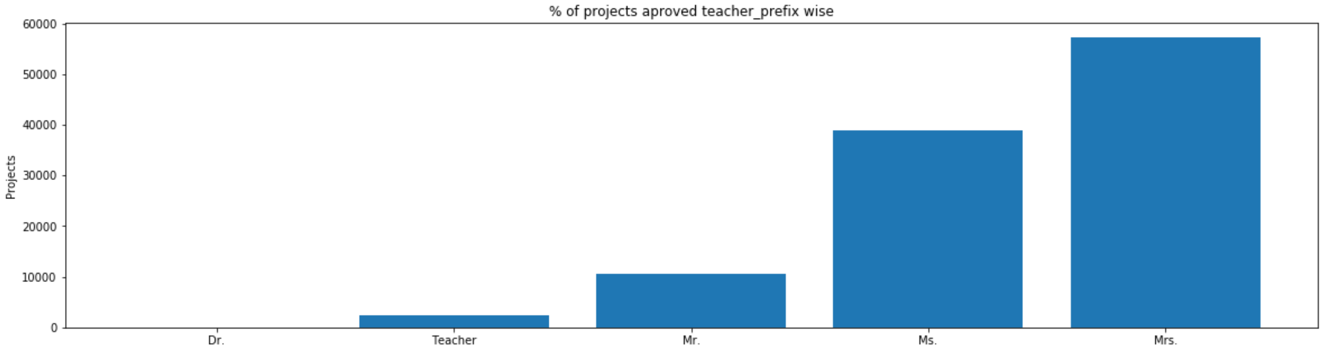
```
Series([], Name: teacher_prefix, dtype: object)
```

In [95]:

```
# 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['teacher_prefix'].values:
    my_counter.update(word.split())
```

In [96]:

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


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

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



In [97]:

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


sub_categories_one_hot_2 = vectorizer.transform(project_data['teacher_prefix'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_2.shape)
```

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

project grade category

In [98]:

```
project_data['project_grade_category'][project_data['project_grade_category'].isnull()==True]
```

Out[98]:

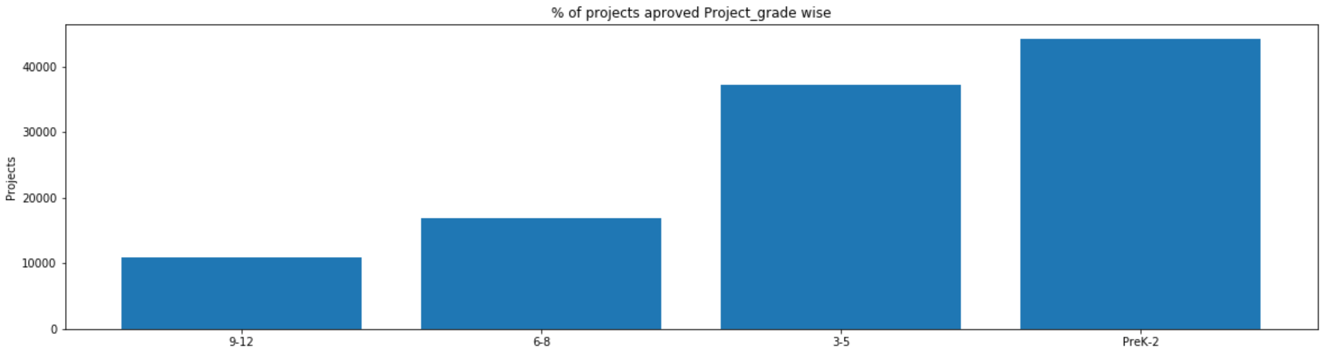
```
Series([], Name: project_grade_category, dtype: object)
```

In [99]:

```
# 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['project_grade_category'].values:
    my_counter.update(word.split())
```

In [100]:

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

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

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



In [101]:

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


sub_categories_one_hot_3 = vectorizer.transform(project_data['project_grade_category'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_3.shape)
```

```
['9-12', '6-8', '3-5', 'PreK-2']
Shape of matrix after one hot encodig  (109248, 4)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [102]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
# 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  (109248, 16623)
```

1.4.2.2 Bag of Words on `project_title`

In [103]:

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

vectorizer1 = CountVectorizer()
text_bow1 = vectorizer1.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",text_bow1.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16867)
```

In [104]:

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

1.4.2.3 TFIDF vectorizer

In [105]:

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

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

1.4.2.4 TFIDF Vectorizer on `project_title`

In [106]:

```
# Similarly you can vectorize for title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer2 = TfidfVectorizer()
text_tfidf2 = vectorizer2.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",text_tfidf2.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16867)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [107]:

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

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

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

# ============================

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

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

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove 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-sa
ve-and-load-variables-in-python/

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


'''
```

Out[107]:

```
'\n# Reading glove vectors in python: 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        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\'
\'))\n\nfor i in preproced_titles:\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 tha
t 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]\n
print("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 pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [110]:

```
# 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:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [111]:

```
# average Word2Vec
# compute average word2vec for each review.
```

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

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

```
100%|██████████| 109248/109248 [00:55<00:00, 1981.92it/s]
```

```
109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [112]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors1.append(vector)

print(len(avg_w2v_vectors1))
print(len(avg_w2v_vectors1[0]))
```

```
100%|██████████| 109248/109248 [00:02<00:00, 38889.88it/s]
```

```
109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [113]:

```
# 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 [114]:

```
# 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%|██████████| 109248/109248 [06:52<00:00, 265.08it/s]
```

```
109248
300
```

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

In [115]:

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

In [116]:

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

In [117]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the 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_vectors1.append(vector)

print(len(tfidf_w2v_vectors1))
print(len(tfidf_w2v_vectors1[0]))
```

```
100%|██████████| 109248/109248 [00:08<00:00, 12415.84it/s]
```

```
109248
300
```

1.4.3 Vectorizing Numerical features

In [118]:

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

```
from sklearn.preprocessing import StandardScaler

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

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

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

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

In [119]:

```
price_standardized
```

Out[119]:

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

In [120]:

```
teacher_scalar = StandardScaler()
teacher_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1
))
print(f"Mean : {teacher_scalar.mean_[0]}, Standard deviation : {np.sqrt(teacher_scalar.var_[0])}")
```

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

In [121]:

```
# Now standardize the data with above maen and variance.
teacher_standardized =
teacher_scalar.transform(project_data['teacher_number_of_previously_posted_projects'].values.reshap
e(-1, 1))
```

In [122]:

```
teacher_standardized
```

Out[122]:

```
array([[-0.40152481],
       [-0.14951799],
       [-0.36552384],
       ...,
       [-0.29352189],
       [-0.40152481],
       [-0.40152481]])
```

1.4.4 Merging all the above features

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

In [123]:

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

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [124]:

```
# 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[124]:

```
(109248, 16663)
```

Assignment 2: Apply TSNE

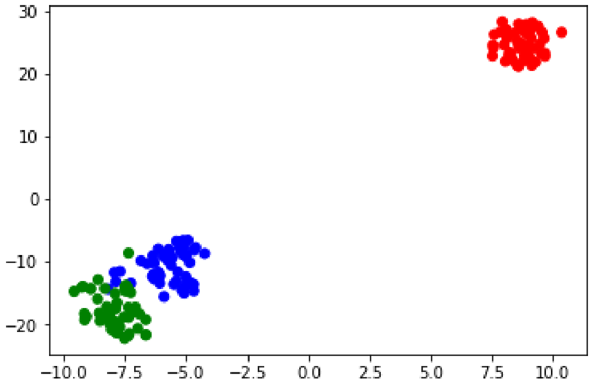
If you are using any code snippet from the internet, you have to provide the reference/citations, as we did in the above cells. Otherwise, it will be treated as plagiarism without citations.

1. In the above cells we have plotted and analyzed many features. Please observe the plots and write the observations in markdown cells below every plot.
2. EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects
3. Build the data matrix using these features
    - school_state : categorical data (one hot encoding)
    - clean_categories : categorical data (one hot encoding)
    - clean_subcategories : categorical data (one hot encoding)
    - teacher_prefix : categorical data (one hot encoding)
    - project_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 [125]:

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

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

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_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(lambda x: colors[x]))
plt.show()
```



In [155]:

```
#hstack with all features including titles text_bow(w2v),text_tfidf,avg_w2v, avg_tfidf

m = hstack((sub_categories_one_hot_1[0:4000],categories_one_hot[0:4000],sub_categories_one_hot[0:4000],sub_categories_one_hot_2[0:4000],sub_categories_one_hot_3[0:4000],text_bow1[0:4000],text_tfidf2[0:4000],avg_w2v_vectors1[0:4000],tfidf_w2v_vectors1[0:4000],teacher_standardized[0:4000]))
m.shape
```

Out[155]:

```
(4000, 34434)
```

In [134]:

```
m.dtype
```

Out[134]:

```
dtype('float64')
```

In [156]:

```
d = hstack((sub_categories_one_hot_1[0:4000],categories_one_hot[0:4000],sub_categories_one_hot[0:4000],sub_categories_one_hot_2[0:4000],sub_categories_one_hot_3[0:4000],text_bow1[0:4000],teacher_standardized[0:4000]))
d.shape
```

Out[156]:

```
(4000, 16967)
```

In [157]:

```
d1=hstack((sub_categories_one_hot_1[0:4000],categories_one_hot[0:4000],sub_categories_one_hot[0:4000],sub_categories_one_hot_2[0:4000],sub_categories_one_hot_3[0:4000],text_tfidf2[0:4000],teacher_standardized[0:4000]))
d1.shape
```

Out[157]:

```
(4000, 16967)
```

In [159]:

In [159]:

```
d2=hstack((sub_categories_one_hot_1[0:4000],categories_one_hot[0:4000],sub_categories_one_hot[0:4000],sub_categories_one_hot_2[0:4000],sub_categories_one_hot_3[0:4000],avg_w2v_vectors1[0:4000],teacher_standardized[0:4000]))
d2.shape
```

Out[159]:

```
(4000, 400)
```

In [160]:

```
d3=hstack((sub_categories_one_hot_1[0:4000],categories_one_hot[0:4000],sub_categories_one_hot[0:4000],sub_categories_one_hot_2[0:4000],sub_categories_one_hot_3[0:4000],tfidf_w2v_vectors1[0:4000],teacher_standardized[0:4000]))
d3.shape
```

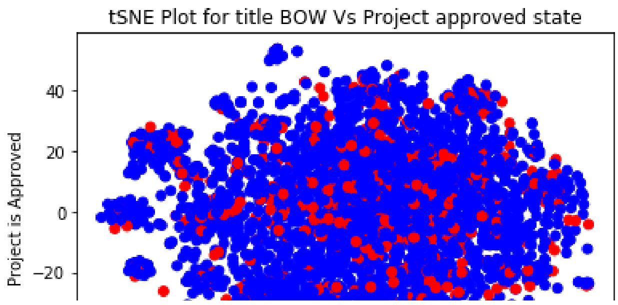
Out[160]:

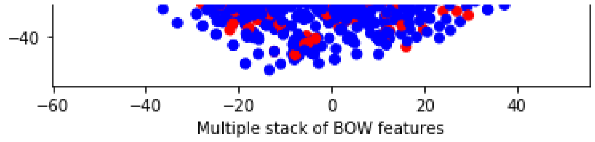
```
(4000, 400)
```

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

In [170]:

```
# 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

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 = d
x=text_bow1[0:4000]
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

X_embedding = tsne.fit_transform(x.toarray())
# 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
plt.title('tSNE Plot for title BOW Vs Project approved state')
plt.xlabel('Multiple stack of BOW features')
plt.ylabel('Project is Approved')

for_tsne = np.hstack((X_embedding, y.as_matrix().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(lambda x: colors[x]))

plt.show()
```

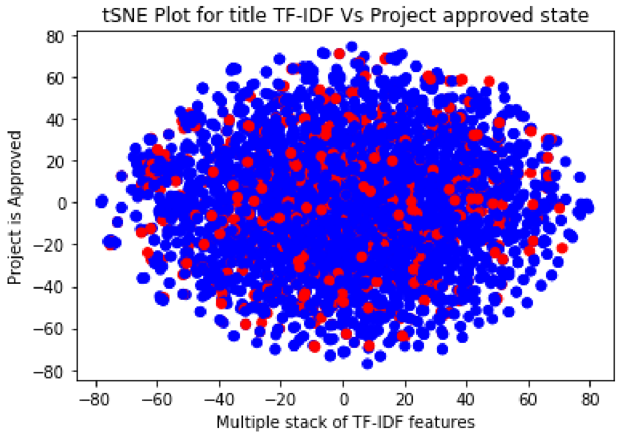




2.2 TSNE with `TFIDF` encoding of `project_title` feature

In [171]:

```
# 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
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 = d1
x= text_tfidf2[0:4000]
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

X_embedding = tsne.fit_transform(x.toarray())
# 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
plt.title('tSNE Plot for title TF-IDF Vs Project approved state')
plt.xlabel('Multiple stack of TF-IDF features')
plt.ylabel('Project is Approved')
for_tsne = np.hstack((X_embedding, y.as_matrix().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(lambda x: colors[x]))
plt.show()
```

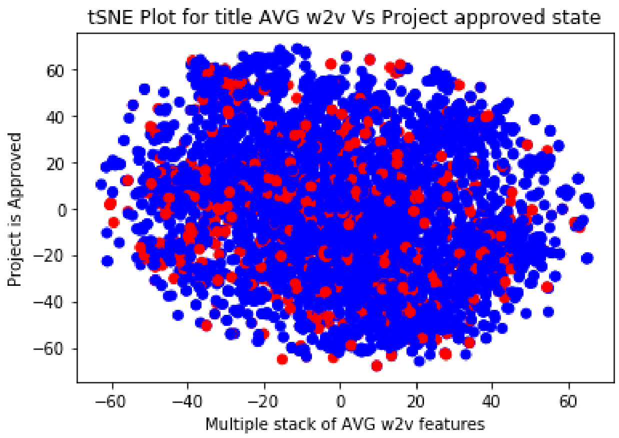


2.3 TSNE with `AVG W2V` encoding of `project_title` feature

In [172]:

```
# 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
```

```
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=avg_w2v_vectors1[0:4000]
#x = d2
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

X_embedding = tsne.fit_transform(np.array(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
plt.title('tSNE Plot for title AVG w2v Vs Project approved state')
plt.xlabel('Multiple stack of AVG w2v features')
plt.ylabel('Project is Approved')
for_tsne = np.hstack((X_embedding, y.as_matrix().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()
```

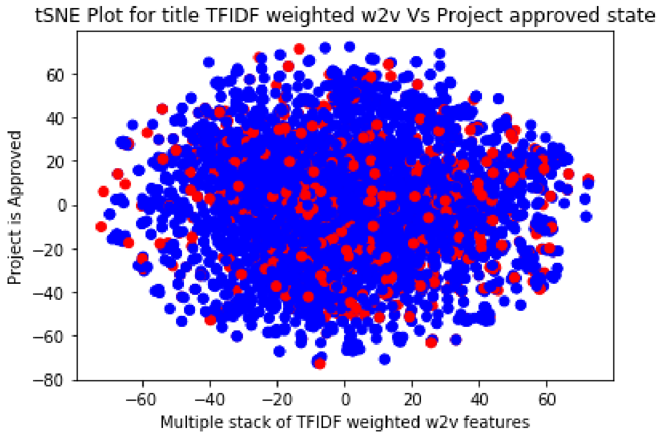


2.4 TSNE with `TFIDF Weighted W2V` encoding of `project_title` feature

In [173]:

```
# 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 labelimport 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 = d3
x=tfidf_w2v_vectors1[0:4000]
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

plt.title('tSNE Plot for title TFIDF weighted w2v Vs Project approved state')
plt.xlabel('Multiple stack of TFIDF weighted w2v features')
plt.ylabel('Project is Approved')

X_embedding = tsne.fit_transform(np.array(x))
```

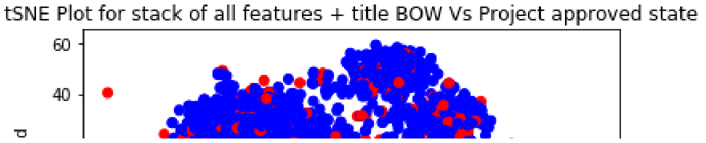
```
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .
toarray() will convert the sparse matrix into dense matrix

for_tsne = np.hstack((X_embedding, y.as_matrix().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()
```

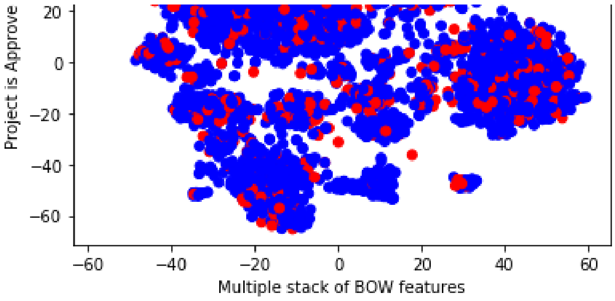


TSNE with all features

In [174]:

```
# 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

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 = d
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

X_embedding = tsne.fit_transform(x.toarray())
# 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
plt.title('tSNE Plot for stack of all features + title BOW Vs Project approved state')
plt.xlabel('Multiple stack of BOW features')
plt.ylabel('Project is Approved')

for_tsne = np.hstack((X_embedding, y.as_matrix().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()
```

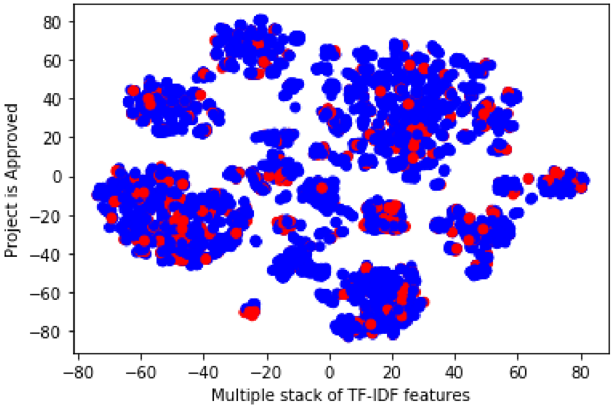




In [175]:

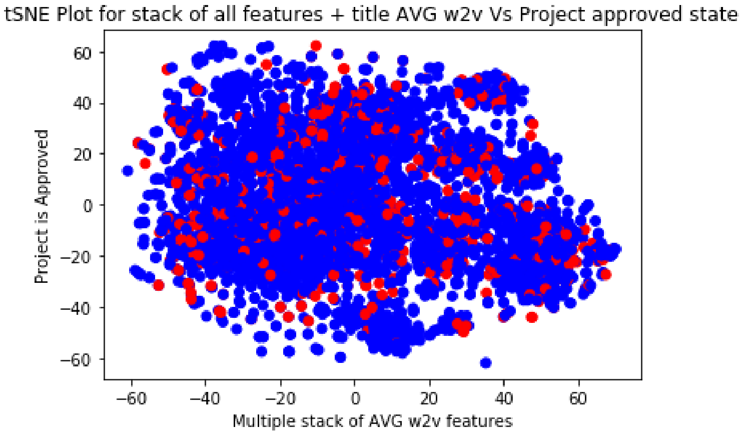
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
import 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 = d1
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

X_embedding = tsne.fit_transform(x.toarray())
# 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
plt.title('tSNE Plot for stack of all features + title TF-IDF Vs Project approved state')
plt.xlabel('Multiple stack of TF-IDF features')
plt.ylabel('Project is Approved')
for_tsne = np.hstack((X_embedding, y.as_matrix().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()
```

tSNE Plot for stack of all features + title TF-IDF Vs Project approved state



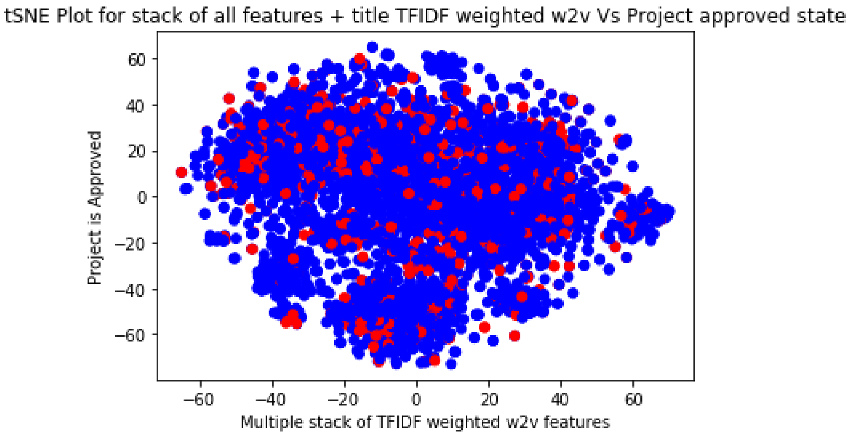
In [176]:

```
# 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
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 = d2
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

X_embedding = tsne.fit_transform(x.toarray())
# 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
plt.title('tSNE Plot for stack of all features + title AVG w2v Vs Project approved state')
plt.xlabel('Multiple stack of AVG w2v features')
plt.ylabel('Project is Approved')
for_tsne = np.hstack((X_embedding, y.as_matrix().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(lambda x: colors[x]))
plt.show()
```



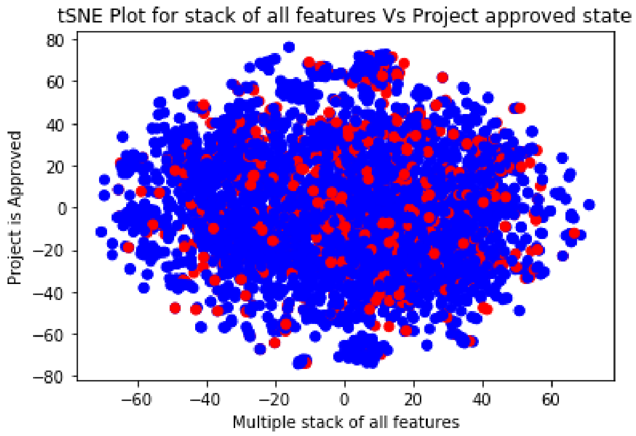
In [177]:

```
# 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 labelimport 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 = d3
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

plt.title('tSNE Plot for stack of all features + title TFIDF weighted w2v Vs Project approved state')
plt.xlabel('Multiple stack of TFIDF weighted w2v features')
plt.ylabel('Project is Approved')

X_embedding = tsne.fit_transform(x.toarray())
# 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.as_matrix().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(lambda x: colors[x]))
plt.show()
```



In [178]:

```
# 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
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 = m
y = project_data['project_is_approved'][0:4000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

X_embedding = tsne.fit_transform(x.toarray())
# 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
plt.title('tSNE Plot for stack of all features Vs Project approved state')
plt.xlabel('Multiple stack of all features')
plt.ylabel('Project is Approved')
for_tsne = np.hstack((X_embedding, y.as_matrix().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(lambda x: colors[x]))

plt.show()
```



2.5 Summary

In [152]:

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

Basic points on TSNE:

1. As we know that TSNE is used for dimensionality reduction and to preserve the neighbourhood of the data.
2. PCA is old and basic type of dimensionality reduction technique.
3. We opt TSNE over PCA because, PCA has a very huge loss of data when data is reduced from high dimensions.
4. There is also a major limitation in TSNE known as Crowding problem.

Summary on results obtained:

1. From the above plotting we can understand that based on the perplexity and learning rate we can obtain a good shape to the data.
2. Even considering the 4000 sample data points BOW and TFIDF including all features has atmost managed to get a proper shape at the end of the compilation when compared to the avg BOW and avg TFIDF.

In [ ]:

```
```