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 (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
	Number of project applications previously submitted by the same teacher

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 [148]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

Only 10000 rows of the total data is used for this whole assignment because of memory constraint and the plots, observations are given based on that.(Data has been modified in both train.csv and resource.csv)

In [149]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [151]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("Number of data points in resource data", resource_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
print(type(project_data))
```

```
Number of data points in train data (10000, 17)
--------------------------------------------------
Number of data points in resource data (10000, 4)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'sch
ool_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']
<class 'pandas.core.frame.DataFrame'>
```

In [152]:
```
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 (10000, 4)
['id' 'description' 'quantity' 'price']
```

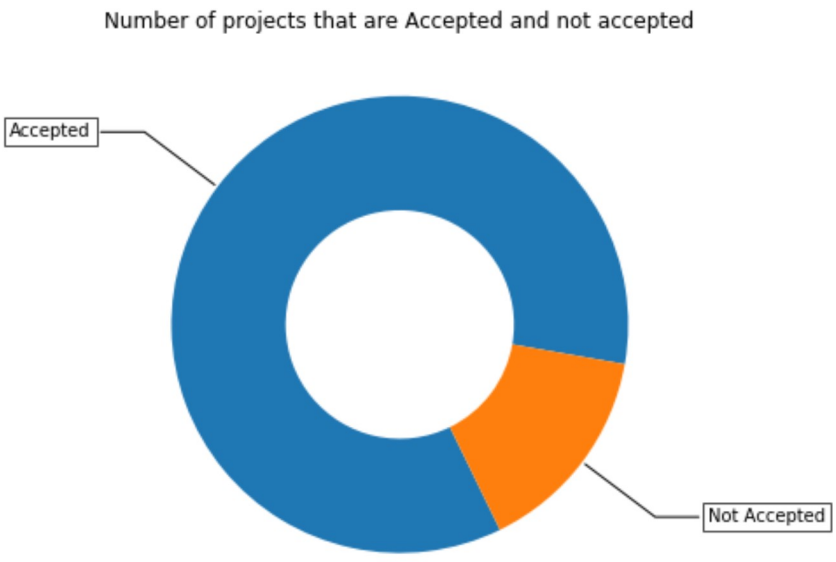
Out[152]:

	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 [153]:
```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.htm
l#sphx-glr-gallery-pie-and-polar-charts-pie-and-donut-labels-py


y_value_counts = project_data['project_is_approved'].value_counts()
print("Number of projects that 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 that 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=-10)

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("Number of projects that are Accepted and not accepted")

plt.show()
```

```
Number of projects that are approved for funding  8500 , ( 85.0 %)
Number of projects that are not approved for funding  1500 , ( 15.0 %)
```



1.2.1 Univariate Analysis: School State

In [154]:
```
# 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[154]:
```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n\nscl = [[0.0, \'rgb(242,240,247)\'],[0.2, \'rgb(218,218,235)\'],[0.4, \'rgb(188,189,220)\'],            [0.6, \'rgb(158,154,200)\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\ndata = [ dict(\n        type=\'choropleth\',\n        colorscale = 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 = dict(\n        title = \'Project Proposals % of Acceptance Rate by US States\',\n        geo = dict(\n            scope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n            showlakes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n        ),\n    )\n\nfig = go.Figure(data=data, layout=layout)\noffline.iplot(fig, filename=\'us-map-heat-map\')\n'
```

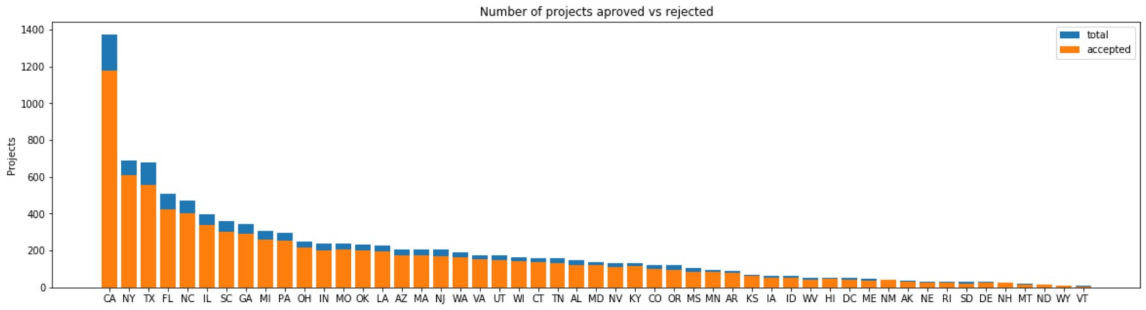
In [155]:
```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.pdf
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
print('='*50)
print("States with highest % approvals")
print(temp.tail(5))
```

```
States with lowest % approvals
   state_code  num_proposals
50         WY       0.727273
41         SD       0.733333
26         MT       0.736842
7          DC       0.750000
37         OR       0.793388
==================================================
States with highest % approvals
   state_code  num_proposals
8          DE       0.896552
17         KY       0.899225
32         NM       0.906977
16         KS       0.954545
28         ND       1.000000
```

In [156]:
```
#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 [157]:
```
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 [158]:
```
univariate_barplots(project_data, 'school_state', 'project_is_approved', False)
```



	school_state	project_is_approved	total	Avg
4	CA	1177	1374	0.856623
34	NY	608	689	0.882438
43	TX	558	680	0.820588
9	FL	422	508	0.830709
27	NC	400	469	0.852878

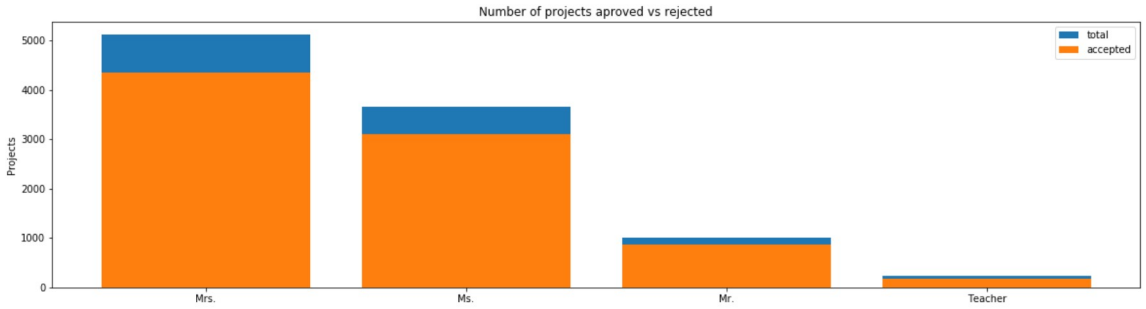
==========

	school_state	project_is_approved	total	Avg
30	NH	24	28	0.857143
26	MT	14	19	0.736842
28	ND	14	14	1.000000
50	WY	8	11	0.727273
46	VT	6	7	0.857143

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

1.2.2 Univariate Analysis: teacher_prefix

In [159]:
```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)
```



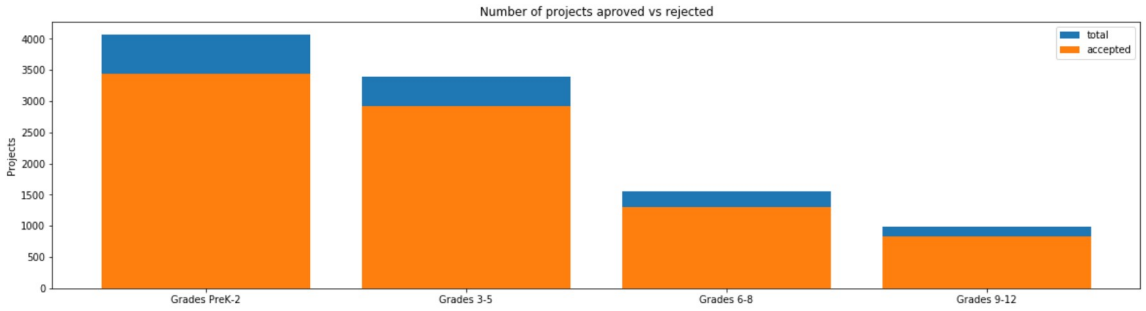
	teacher_prefix	project_is_approved	total	Avg
1	Mrs.	4354	5120	0.850391
2	Ms.	3110	3647	0.852756
0	Mr.	861	1006	0.855865
3	Teacher	174	226	0.769912

==========

	teacher_prefix	project_is_approved	total	Avg
1	Mrs.	4354	5120	0.850391
2	Ms.	3110	3647	0.852756
0	Mr.	861	1006	0.855865
3	Teacher	174	226	0.769912

1.2.3 Univariate Analysis: project_grade_category

In [160]:
```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', top=False)
```




```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                3441   4069  0.845662
0             Grades 3-5                2920   3390  0.861357
1             Grades 6-8                1305   1546  0.844114
2            Grades 9-12                 834    995  0.838191
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                3441   4069  0.845662
0             Grades 3-5                2920   3390  0.861357
1             Grades 6-8                1305   1546  0.844114
2            Grades 9-12                 834    995  0.838191
```

1.2.4 Univariate Analysis: project_subject_categories

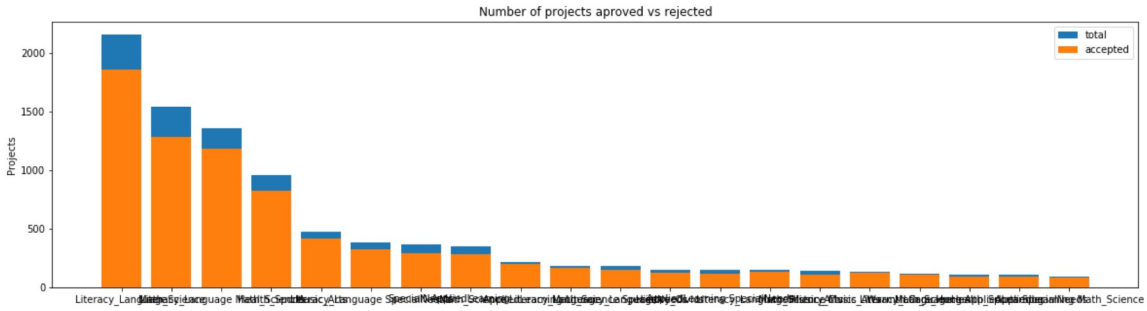
In [161]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

In [162]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[162]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	05-12-20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	25-10-20

In [163]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=20)
```



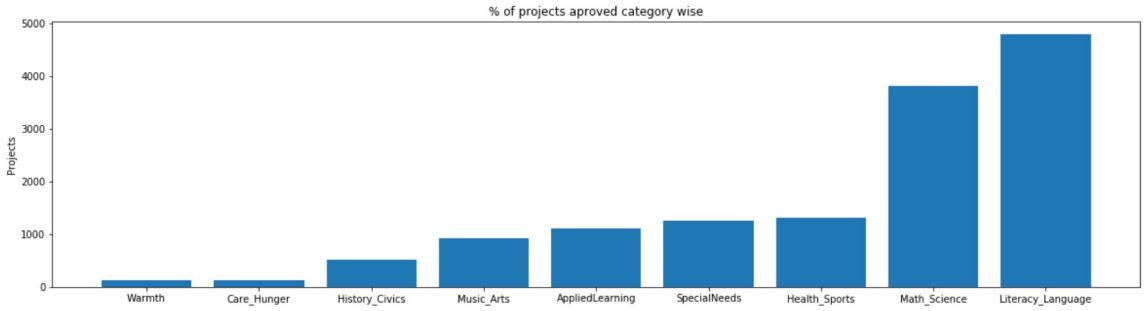
	clean_categories	project_is_approved	total	Avg
23	Literacy_Language	1856	2160	0.859259
31	Math_Science	1284	1546	0.830530
27	Literacy_Language Math_Science	1183	1360	0.869853
8	Health_Sports	826	961	0.859521
39	Music_Arts	415	477	0.870021

==========

	clean_categories	project_is_approved	total	Avg
19	History_Civics Literacy_Language	127	135	0.940741
49	Warmth Care_Hunger	105	115	0.913043
32	Math_Science AppliedLearning	93	111	0.837838
14	Health_Sports SpecialNeeds	93	110	0.845455
4	AppliedLearning Math_Science	79	95	0.831579

In [164]:
```
# 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 [165]:
```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))


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

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



In [166]:
```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Warmth               :       127
Care_Hunger          :       127
History_Civics       :       522
Music_Arts           :       930
AppliedLearning      :      1116
SpecialNeeds         :      1260
Health_Sports        :      1317
Math_Science         :      3812
Literacy_Language    :      4787
```

1.2.5 Univariate Analysis: project_subject_subcategories

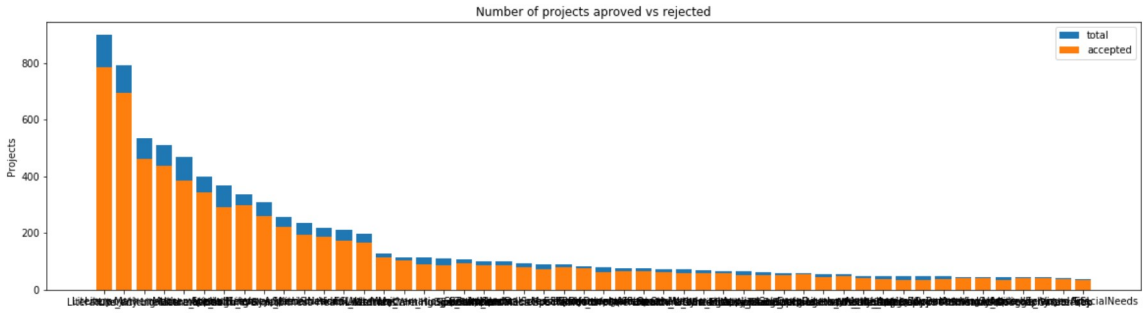
In [167]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.
com/a/47301924/4084039

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

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

In [168]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[168]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	05-12-20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	25-10-20

In [169]:
```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved',
top=50)
```



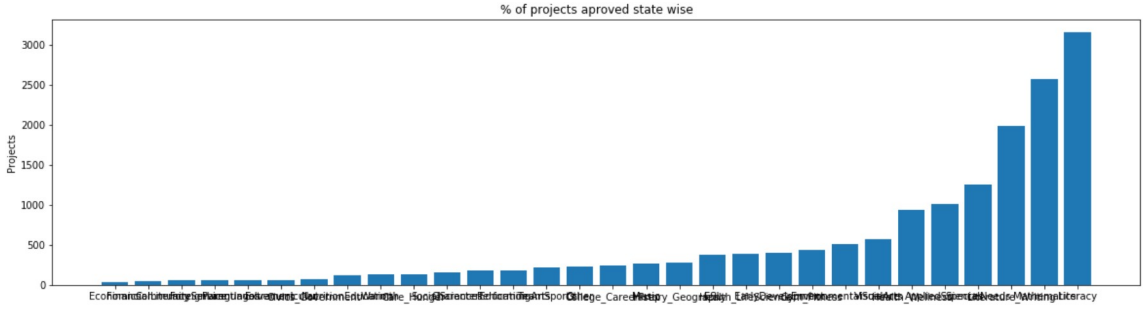
	clean_subcategories	project_is_approved	total	Avg
231	Literacy	784	899	0.872080
233	Literacy Mathematics	694	790	0.878481
244	Literature_Writing Mathematics	461	535	0.861682
232	Literacy Literature_Writing	436	510	0.854902
254	Mathematics	386	470	0.821277

==========

	clean_subcategories	project_is_approved	total	Avg
3	AppliedSciences College_CareerPrep	36	44	0.818182
222	History_Geography Literacy	42	44	0.954545
241	Literacy VisualArts	41	44	0.931818
23	AppliedSciences SpecialNeeds	37	42	0.880952
101	ESL	34	38	0.894737

In [170]:
```
# 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 [171]:
```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))


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

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

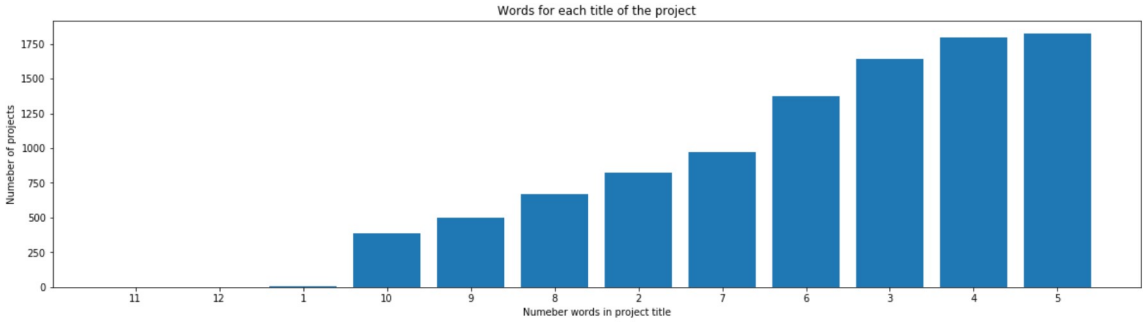


In [172]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :        33
FinancialLiteracy    :        47
CommunityService     :        56
ForeignLanguages     :        63
ParentInvolvement    :        65
Extracurricular      :        65
Civics_Government    :        72
NutritionEducation   :       115
Warmth               :       127
Care_Hunger          :       127
SocialSciences       :       157
CharacterEducation   :       178
PerformingArts       :       183
TeamSports           :       218
Other                :       229
College_CareerPrep   :       244
Music                :       266
History_Geography    :       276
ESL                  :       378
Health_LifeScience   :       391
EarlyDevelopment     :       396
Gym_Fitness          :       443
EnvironmentalScience :       507
VisualArts           :       575
Health_Wellness      :       934
AppliedSciences      :      1009
SpecialNeeds         :      1260
Literature_Writing   :      1988
Mathematics          :      2573
Literacy             :      3156
```

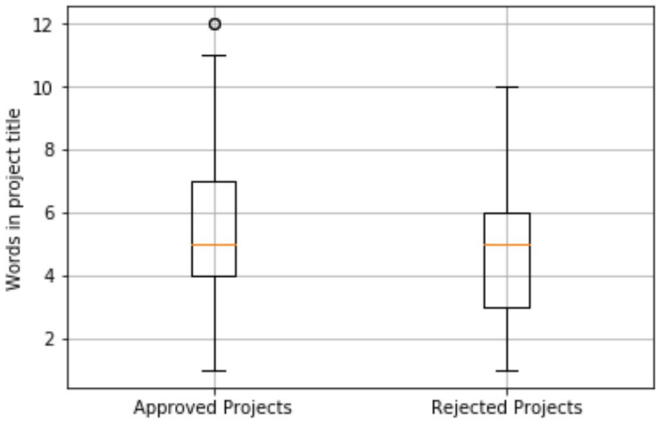
1.2.6 Univariate Analysis: Text features (Title)

In [173]:
```
#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()
```

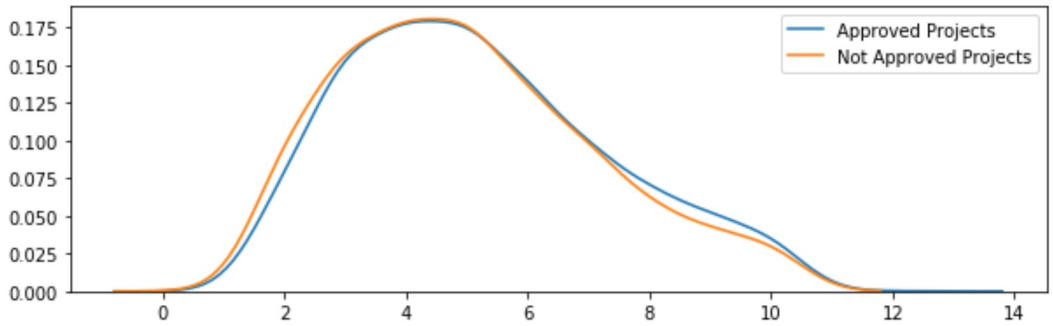


In [174]:
```
approved_title_word_count = project_data[project_data['project_is_approved']==
1]['project_title'].str.split().apply(len)
approved_title_word_count = approved_title_word_count.values

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 [175]:
```
# 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 [176]:
```
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()
```



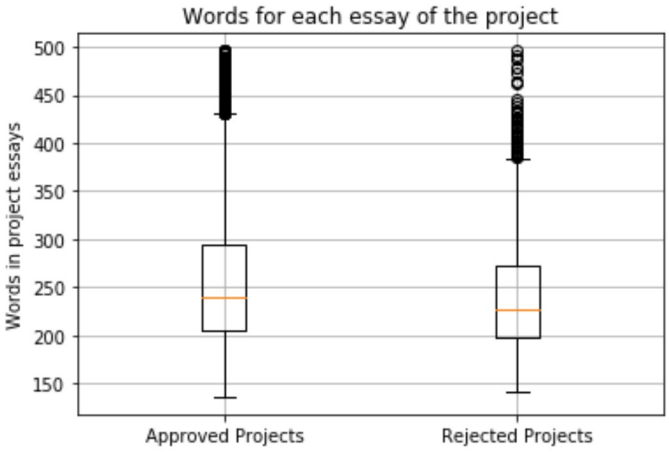
1.2.7 Univariate Analysis: Text features (Project Essay's)

In [177]:
```
# 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)
print(type(project_data["essay"]))
```

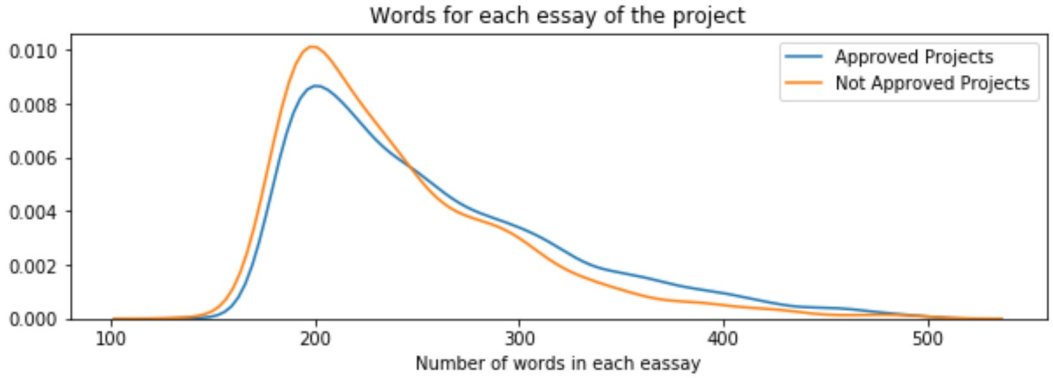
```
<class 'pandas.core.series.Series'>
```

In [178]:
```
approved_word_count = project_data[project_data['project_is_approved']==1]['ess
ay'].str.split().apply(len)
# print(approved_word_count)
approved_word_count = approved_word_count.values
# print(approved_word_count)
rejected_word_count = project_data[project_data['project_is_approved']==0]['ess
ay'].str.split().apply(len)
rejected_word_count = rejected_word_count.values
```

In [179]:
```
# 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 [180]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```



1.2.8 Univariate Analysis: Cost per project

In [181]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[181]:

	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 [182]:
```
# 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)
```

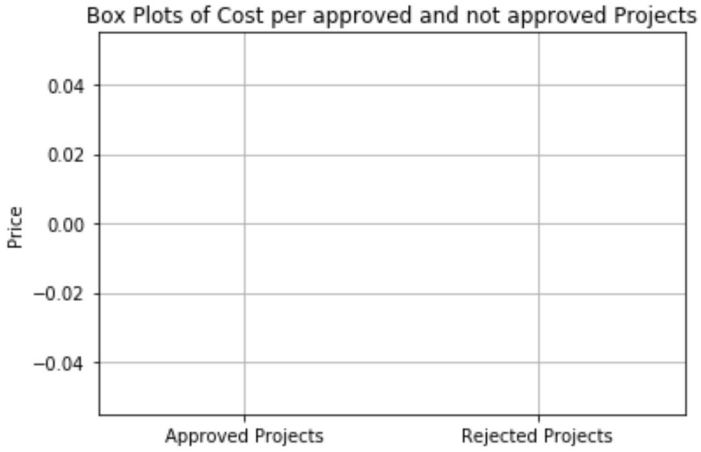
Out[182]:

	id	price	quantity
0	p000341	1295.23	12
1	p000426	117.45	24

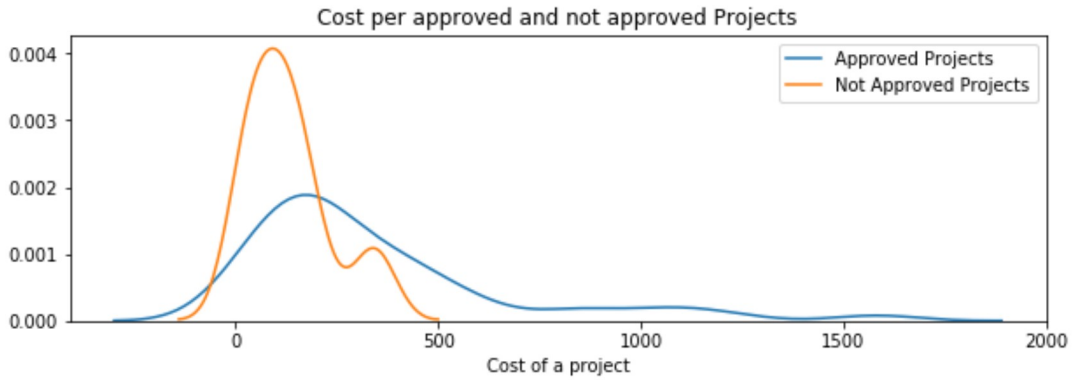
In [183]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [184]:
```
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 [185]:
```
# 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 [186]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_price, hist=False, label="Approved Projects")
sns.distplot(rejected_price, hist=False, label="Not Approved Projects")
plt.title('Cost per approved and not approved Projects')
plt.xlabel('Cost of a project')
plt.legend()
plt.show()
```



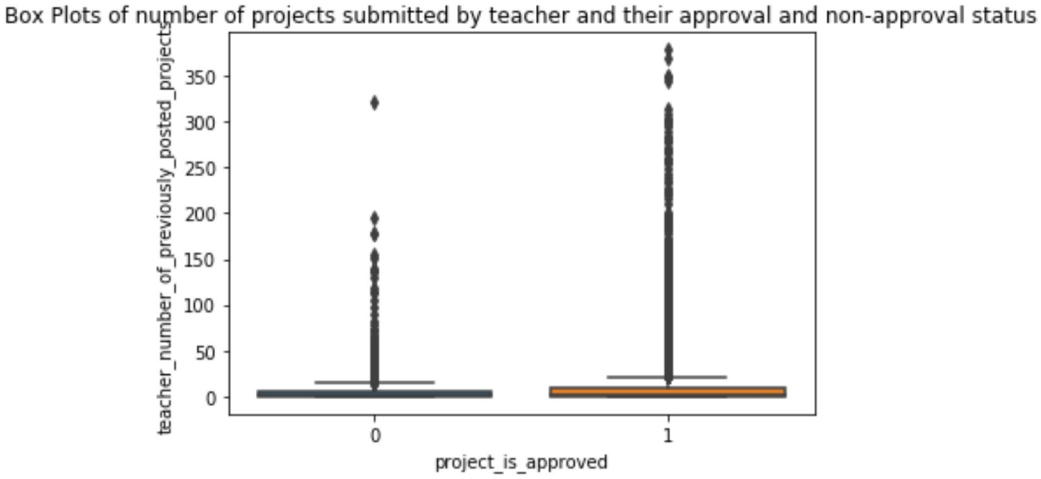
In [187]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 insta
ll 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.perc
entile(rejected_price,i), 3)])
print(x)
```

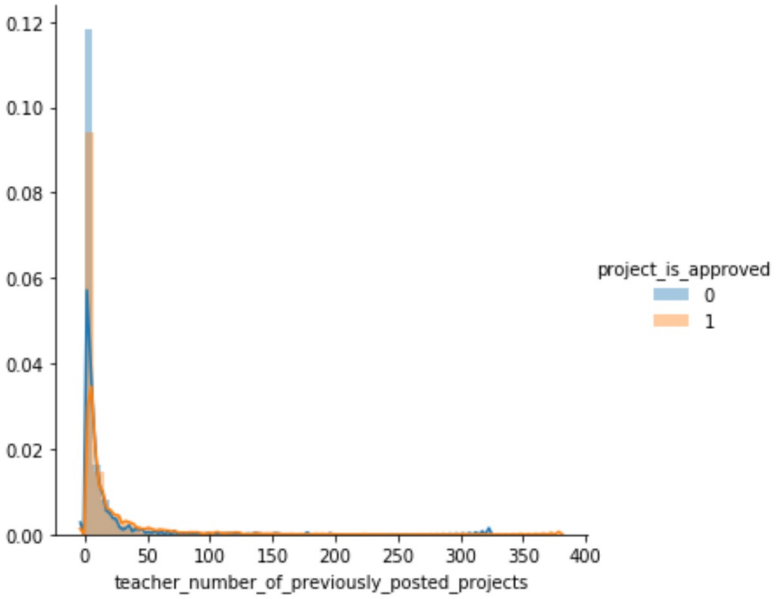
Percentile	Approved Projects	Not Approved Projects
0	nan	nan
5	nan	nan
10	nan	nan
15	nan	nan
20	nan	nan
25	nan	nan
30	nan	nan
35	nan	nan
40	nan	nan
45	nan	nan
50	nan	nan
55	nan	nan
60	nan	nan
65	nan	nan
70	nan	nan
75	nan	nan
80	nan	nan
85	nan	nan
90	nan	nan
95	nan	nan
100	nan	nan

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 [188]:
```
# we get the teacher's previously posted project data using train_data.csv file
f1 = project_data[['teacher_number_of_previously_posted_projects','project_is_approved']]
a1 = f1['teacher_number_of_previously_posted_projects']
a2 = f1['project_is_approved']
a1=a1.values
a2=a2.values

#print(a1)
#print(a2)

# https://seaborn.pydata.org/generated/seaborn.boxplot.html
# http://cmdlinetips.com/2018/03/how-to-make-boxplots-in-python-with-pandas-and-seaborn/
sns.boxplot(x='project_is_approved',y='teacher_number_of_previously_posted_projects', data=f1)
plt.title('Box Plots of number of projects submitted by teacher and their approval and non-approval status')
plt.show()

# https://seaborn.pydata.org/generated/seaborn.FacetGrid.html
# https://www.tutorialspoint.com/seaborn/seaborn_facet_grid.htm
sns.FacetGrid(f1, hue="project_is_approved", height=5).map(sns.distplot, "teacher_number_of_previously_posted_projects").add_legend();
plt.show();

x = PrettyTable()
x.field_names = ["Number of projects submitted by teacher", "Approved Projects", "Not Approved Projects"]
for i in range(0,101,5):
    x.add_row([i,np.round(np.percentile(a1,i), 3), np.round(np.percentile(a2,i), 3)])
print(x)
```


Box Plots of number of projects submitted by teacher and their approval and non-approval status
teacher_number_of_previously_posted_projects
0
50
100
150
200
250
300
350
0
1
project_is_approved


0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250
300
350
400
teacher_number_of_previously_posted_projects
project_is_approved
0
1

Number of projects submitted by teacher	Approved Projects	Not Approved Projects
0	0.0	0.0
5	0.0	0.0
10	0.0	0.0
15	0.0	0.85
20	0.0	1.0
25	0.0	1.0
30	1.0	1.0
35	1.0	1.0
40	1.0	1.0
45	2.0	1.0
50	2.0	1.0
55	3.0	1.0
60	4.0	1.0
65	5.0	1.0
70	7.0	1.0
75	9.0	1.0
80	12.0	1.0
85	18.0	1.0
90	28.0	1.0
95	53.0	1.0
100	378.0	1.0

Observation: According to the above analysis the box plot and the PDF doesnot give much information but from the pretty table(percentile values) we can gather that as the number of Teacher's previously posted projects increase their approval rate is also getting increased, the rate of not getting approved is lower.

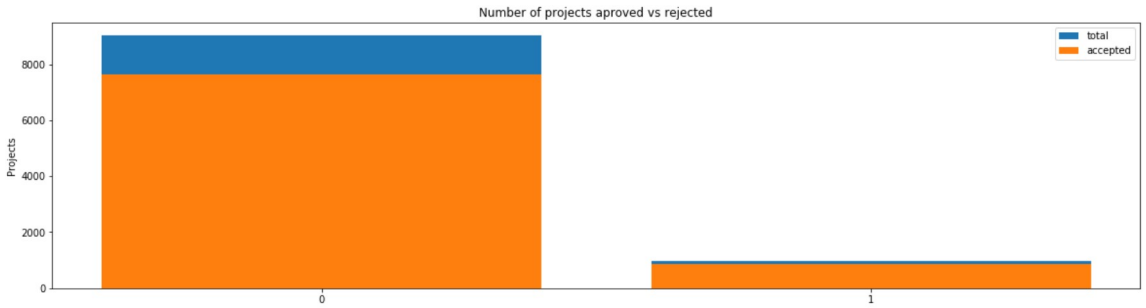
1.2.10 Univariate Analysis: project_resource_summary

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 [189]:
```
# https://stackoverflow.com/questions/19859282/check-if-a-string-contains-a-num
ber
# https://stackoverflow.com/questions/29517072/add-column-to-dataframe-with-def
ault-value
import pdb
aa1 = {}
aa2 = {}
aa3 = []
v = range(len(a1))
def hasNumbers(a1):
    for j in tqdm(v):
        for k in a1[j].split():
            # pdb.set_trace()
            if k.isdigit():
                aa1[j] = int(k)


def form_list_num():
    for x in v:
        if x in aa1.keys():
            aa2[x] = aa1[x]
        else:
            aa2[x] = 0

def num_conversion():
    for r in aa2.values():
        if r == 0:
            aa3.append(0)
        else:
            aa3.append(1)

a1 = []
for i in project_data['project_resource_summary']:
    a1.append(i)

hasNumbers(a1)
form_list_num()
num_conversion()

# print(aa3)
# for l in tqdm(range (30)):
#    pdb.set_trace()
#    print(aa2.values()[l])

# len(aa2)
project_data['number_in_summary'] = aa3
univariate_barplots(project_data, 'number_in_summary', 'project_is_approved', t
op=2)
```

```
100%|██████████████████████████████████████████████████████████████████████████████
| 10000/10000 [00:00<00:00, 86288.35it/s]
```




```
   number_in_summary  project_is_approved  total       Avg
0                  0                 7622   9035  0.843608
1                  1                  878    965  0.909845
==================================================
   number_in_summary  project_is_approved  total       Avg
0                  0                 7622   9035  0.843608
1                  1                  878    965  0.909845
```

Observation: The presence of numerical digits in the Project Resource Summary column of the DonorsChoose(train) data appears to be affecting the acceptance or rejectance of a project since from the above univariate bar plot analysis it can be seen that 90% of the total projects with numerical digits in them got approved whereas only 84% of the total projects without numerical digits in them got approved. Hence it can be concluded that the presence of numerical digits in the Project Resource Summary column of the DonorsChoose(train) data is useful in the classification of whether a project will be approved or not and thus it will be used further.

1.3 Text preprocessing

1.3.1 Essay Text

In [190]:
```
project_data.head(2)
```

Out[190]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	05-12-2(
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	25-10-2(

2 rows × 21 columns

In [191]:
```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
#print("="*50)
#print(project_data['essay'].values[20000])
#print("="*50)
#print(project_data['essay'].values[99999])
#print("="*50)
```

My students are English learners that are working on English as their second o
r third languages. We are a melting pot of refugees, immigrants, and native-bo
rn Americans bringing the gift of language to our school. \r\n\r\n We have ove
r 24 languages represented in our English Learner program with students at eve
ry level of mastery. We also have over 40 countries represented with the fami
lies within our school. Each student brings a wealth of knowledge and experie
nces to us that open our eyes to new cultures, beliefs, and respect.\"The limi
ts of your language are the limits of your world.\"-Ludwig Wittgenstein Our E
nglish learner's have a strong support system at home that begs for more resou
rces. Many times our parents are learning to read and speak English along sid
e of their children. Sometimes this creates barriers for parents to be able t
o help their child learn phonetics, letter recognition, and other reading skil
ls.\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 home is able to assis
t. 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 spec
ially chosen by the English Learner Teacher and will be sent home regularly to
watch. The videos are to help the child develop early reading skills.\r\n\r\n
Parents that do not have access to a dvd player will have the opportunity to c
heck out a dvd player to use for the year. The plan is to use these videos an
d educational dvd's for the years to come for other EL students.\r\nnannan
==
The 51 fifth grade students that will cycle through my classroom this year all
love learning, at least most of the time. At our school, 97.3% of the students
receive free or reduced price lunch. Of the 560 students, 97.3% are minority s
tudents. \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 th
e school hosts a carnival to celebrate the hard work put in during the school
year, with a dunk tank being the most popular activity.My students will use th
ese five brightly colored Hokki stools in place of regular, stationary, 4-legg
ed chairs. As I will only have a total of ten in the classroom and not enough
for each student to have an individual one, they will be used in a variety of
ways. During independent reading time they will be used as special chairs stud
ents will each use on occasion. I will utilize them in place of chairs at my s
mall group tables during math and reading times. The rest of the day they will
be used by the students who need the highest amount of movement in their life
in order to stay focused on school.\r\n\r\nWhenever asked what the classroom i
s 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 in group with m
e on the Hokki Stools, they are always moving, but at the same time doing thei
r work. Anytime the students get to pick where they can sit, the Hokki Stools
are the first to be taken. There are always students who head over to the kidn
ey 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
the 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 tha
t 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 ma
de up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nT
hey attend a Title I school, which means there is a high enough percentage of
free and reduced-price lunch to qualify. Our school is an \"open classroom\" c
oncept, which is very unique as there are no walls separating the classrooms.
These 9 and 10 year-old students are very eager learners; they are like sponge
s, absorbing all the information and experiences and keep on wanting more.With
these resources such as the comfy red throw pillows and the whimsical nautical
hanging decor and the blue fish nets, I will be able to help create the mood i
n 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 st
ep foot into our classroom for the first time on Meet the Teacher evening. I'l
l take pictures of each child with them, have them developed, and then hung in

In [192]:
```
# 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 [193]:
```
sent = decontracted(project_data['essay'].values[49])
print(sent)
print("="*50)
```

```
Located in the Bay Area, our city is a melting pot of diversity and culture an
d our school is home to amazing students who enjoy and embrace each other. Wit
h a variety of excited and motivated learners, our class is dedicated to imple
menting more science and technology into the classroom. \r\n\r\nStudents are e
ager to learn new science curriculum as well as develop skills and strategies
to make them better readers. In the classroom, students are full of energy, po
sitivity, and confidence. They enjoy learning new things, especially topics re
lated to science. They are inquisitive and determined, and as an educator, I i
ntend to continue nurturing this drive. As an enthusiastic and energetic educa
tor, I am looking forward to helping my students reach for the stars, theoreti
cally and literally!My students enjoy being active outside as well as inside t
he classroom, and Rainy Day Recess will not stop them from being active! They
have requested to participate in regular activities that get them up and movin
g throughout the day. In addition to Rainy Day Recess activities, they will be
nefit from regular \"body breaks\" as well as \"brain breaks\" every day. This
allows for them to get up and move and get their wiggles out. They will need a
bright and colorful carpet with a design which allows for them to have individ
ual space. \r\nThe requested carpet provides them with ample room to participa
te in fun and invigorating indoor activities, like hot potato, \"Farmer and th
e Pig\", speed ball, and many others, all while respecting the space each chil
d needs to participate. Additionally, the carpet provides them with space for
daily yoga, dancing, and indoor PE. They will also benefit from books on nutri
tion, and maintaining a healthy and active lifestyle. On rainy days, they will
benefit from an indoor playground, which can be created with the use of balanc
ing pods. The use of fitness dice will allow students to take ownership of the
ir activity, as each child will have a turn in rolling the dice to determine h
ow many of each activity we should do as a group. \r\nThese supplies will help
encourage students to be active and to have fun while doing so. Even on rainy
days when students are stuck inside, they will be able to stimulate their mind
s and bodies by balancing on the indoor obstacle course, practicing stretching
on the carpet, rolling the fitness dice to stay active, or construct a game of
\"Farmer and the Pig\". Health, fitness, and nutrition are important to me and
my students!!nannan
==================================================
```

In [194]:
```
# \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)
```

```
Located in the Bay Area, our city is a melting pot of diversity and culture an
d our school is home to amazing students who enjoy and embrace each other. Wit
h a variety of excited and motivated learners, our class is dedicated to imple
menting more science and technology into the classroom.     Students are eager
to learn new science curriculum as well as develop skills and strategies to ma
ke them better readers. In the classroom, students are full of energy, positiv
ity, and confidence. They enjoy learning new things, especially topics related
to science. They are inquisitive and determined, and as an educator, I intend
to continue nurturing this drive. As an enthusiastic and energetic educator, I
am looking forward to helping my students reach for the stars, theoretically a
nd literally!My students enjoy being active outside as well as inside the clas
sroom, and Rainy Day Recess will not stop them from being active! They have re
quested to participate in regular activities that get them up and moving throu
ghout the day. In addition to Rainy Day Recess activities, they will benefit f
rom regular  body breaks  as well as  brain breaks  every day. This allows for
them to get up and move and get their wiggles out. They will need a bright and
colorful carpet with a design which allows for them to have individual space.
The requested carpet provides them with ample room to participate in fun and i
nvigorating indoor activities, like hot potato,  Farmer and the Pig , speed ba
ll, and many others, all while respecting the space each child needs to partic
ipate. Additionally, the carpet provides them with space for daily yoga, danci
ng, and indoor PE. They will also benefit from books on nutrition, and maintai
ning a healthy and active lifestyle. On rainy days, they will benefit from an
indoor playground, which can be created with the use of balancing pods. The us
e of fitness dice will allow students to take ownership of their activity, as
each child will have a turn in rolling the dice to determine how many of each
activity we should do as a group.   These supplies will help encourage student
s to be active and to have fun while doing so. Even on rainy days when student
s are stuck inside, they will be able to stimulate their minds and bodies by b
alancing on the indoor obstacle course, practicing stretching on the carpet, r
olling the fitness dice to stay active, or construct a game of  Farmer and the
Pig . Health, fitness, and nutrition are important to me and my students!!nann
an
```

In [195]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Located in the Bay Area our city is a melting pot of diversity and culture and
our school is home to amazing students who enjoy and embrace each other With a
variety of excited and motivated learners our class is dedicated to implementi
ng more science and technology into the classroom Students are eager to learn
new science curriculum as well as develop skills and strategies to make them b
etter readers In the classroom students are full of energy positivity and conf
idence They enjoy learning new things especially topics related to science The
y are inquisitive and determined and as an educator I intend to continue nurtu
ring this drive As an enthusiastic and energetic educator I am looking forward
to helping my students reach for the stars theoretically and literally My stud
ents enjoy being active outside as well as inside the classroom and Rainy Day
Recess will not stop them from being active They have requested to participate
in regular activities that get them up and moving throughout the day In additi
on to Rainy Day Recess activities they will benefit from regular body breaks a
s well as brain breaks every day This allows for them to get up and move and g
et their wiggles out They will need a bright and colorful carpet with a design
which allows for them to have individual space The requested carpet provides t
hem with ample room to participate in fun and invigorating indoor activities l
ike hot potato Farmer and the Pig speed ball and many others all while respect
ing the space each child needs to participate Additionally the carpet provides
them with space for daily yoga dancing and indoor PE They will also benefit fr
om books on nutrition and maintaining a healthy and active lifestyle On rainy
days they will benefit from an indoor playground which can be created with the
use of balancing pods The use of fitness dice will allow students to take owne
rship of their activity as each child will have a turn in rolling the dice to
determine how many of each activity we should do as a group These supplies wil
l help encourage students to be active and to have fun while doing so Even on
rainy days when students are stuck inside they will be able to stimulate their
minds and bodies by balancing on the indoor obstacle course practicing stretch
ing on the carpet rolling the fitness dice to stay active or construct a game
of Farmer and the Pig Health fitness and nutrition are important to me and my
students nannan
```

In [196]:
```
# 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', 'its
elf', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'th
at', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'h
as', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becaus
e', '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', 'o
ff', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al
l', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than
', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should'
ve", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "
didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma
', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [197]:
```
# 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%|██████████████████████████████████████████████████████████████████████████
██| 10000/10000 [00:06<00:00, 1541.89it/s]
```

In [198]:
```
# after preprocesing
preprocessed_essays[4999]
```

Out[198]:
```
'loud proud we special basketball family like no our school great community va
st diverseness we surrounded colleges low income housing we pride preparing at
hletes great court our students recite every day we destined greatness i belie
ve wholeheartedly i forming winners life basketball a great kids coming way we
need socks add two uniforms every basketball season girls basketball team stri
ves play best not i push give court i also teach take pride look team we want
look like team head toe girls feel good play ball look good court i seen lime
green socks purple socks crazy mismatched socks we need uniformity way around
nannan'
```

1.3.2 Project title Text

In [199]:
```
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())

preprocessed_titles[4998]
```

```
100%|██████████████████████████████████████████████████████████████████████████
| 10000/10000 [00:00<00:00, 30115.01it/s]
```

Out[199]: 'read baby read favorite titles'

Observation: The 'project_title' column of the DonorsChoose (train.csv) data has text data and has been preprocessed the same way as it was done for Project Essay. Thus yielding text with words in lowercase and without any stopwords, special characters, tags, short forms of words in 'preprocessed_titles'.

1. 4 Preparing data for models

In [200]:
```
project_data.columns
```

Out[200]:
```
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
',
       'number_in_summary'],
      dtype='object')
```

we are going to consider

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

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

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

1.4.1 Vectorizing Categorical data

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

In [201]:
```
# 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)
```

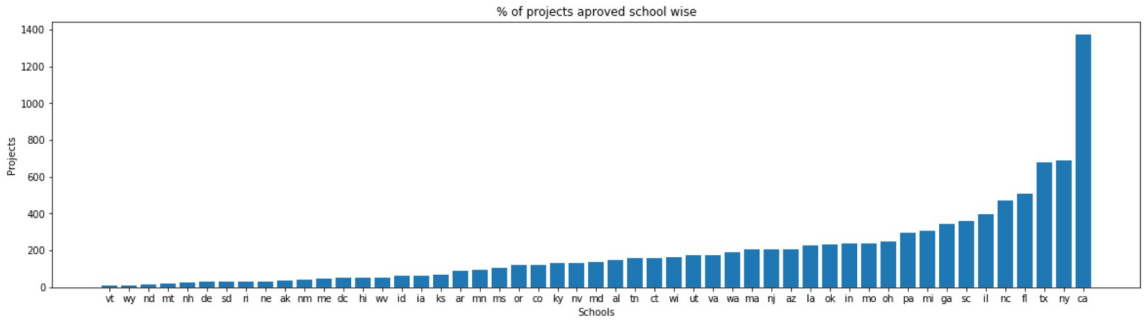
```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (10000, 9)
```

In [202]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


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

```
['Economics', 'FinancialLiteracy', 'CommunityService', 'ForeignLanguages', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'CharacterEducation', 'PerformingArts', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'ESL', 'Health_LifeScience', 'EarlyDevelopment', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (10000, 30)
```

In [203]:
```
project_data = pd.read_csv('train_data.csv')
sch1_catogories = list(project_data['school_state'].values)
# print(sch1_catogories)
school_list = []
for sent in sch1_catogories:
    school_list.append(sent.lower().strip())
project_data['school_categories'] = school_list
project_data.drop(['school_state'], axis=1, inplace=True)
print(project_data.head(2))

# count of all the words in corpus python: https://stackoverflow.com/a/22898595
/4084039
my_counter_sch = Counter()
for word in project_data['school_categories'].values:
    my_counter_sch.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sch_dict = dict(my_counter_sch)
sorted_sch_dict = dict(sorted(sch_dict.items(), key=lambda kv: kv[1]))


ind1 = np.arange(len(sorted_sch_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind1, list(sorted_sch_dict.values()))
plt.xlabel('Schools')
plt.ylabel('Projects')
plt.title('% of projects aproved school wise')
plt.xticks(ind1, list(sorted_sch_dict.keys()))
plt.show()

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

vectorizer = CountVectorizer(vocabulary=list(sorted_sch_dict.keys()), lowercas
e=False, binary=True)
vectorizer.fit(project_data['school_categories'].values)
print(vectorizer.get_feature_names())

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

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

  project_submitted_datetime project_grade_category  \
0           05-12-2016 13:43          Grades PreK-2
1           25-10-2016 09:22             Grades 6-8

         project_subject_categories     project_subject_subcategories  \
0                Literacy & Language                       ESL, Literacy
1  History & Civics, Health & Sports  Civics & Government, Team Sports

                                  project_title  \
0  Educational Support for English Learners at Home
1            Wanted: Projector for Hungry Learners

                                  project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...

                                  project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...           NaN
1  The projector we need for our school is very c...           NaN

  project_essay_4                         project_resource_summary  \
0             NaN  My students need opportunities to practice beg...
1             NaN  My students need a projector to help with view...

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0
1                                             7                    1

  school_categories
0                in
1                fl
```



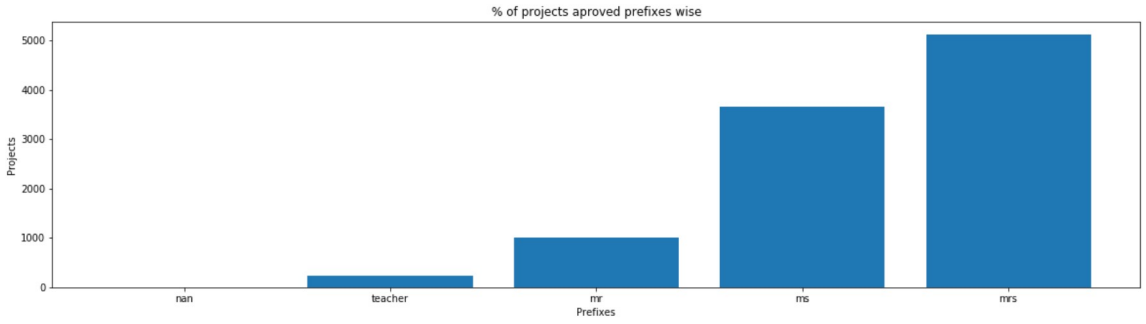
```
vt                    :         7
wy                    :        11
nd                    :        14
mt                    :        19
nh                    :        28
de                    :        29
sd                    :        30
ri                    :        31
ne                    :        32
ak                    :        35
nm                    :        43
me                    :        45
dc                    :        52
hi                    :        53
wv                    :        54
id                    :        60
ia                    :        64
ks                    :        66
ar                    :        91
mn                    :        94
ms                    :       103
or                    :       121
co                    :       123
ky                    :       129
nv                    :       130
md                    :       135
al                    :       146
tn                    :       158
ct                    :       160
wi                    :       165
ut                    :       174
va                    :       176
wa                    :       191
ma                    :       207
nj                    :       207
az                    :       208
la                    :       229
ok                    :       230
in                    :       240
mo                    :       240
oh                    :       246
pa                    :       298
mi                    :       308
ga                    :       345
sc                    :       358
il                    :       395
nc                    :       469
fl                    :       508
tx                    :       680
ny                    :       689
ca                    :      1374
['vt', 'wy', 'nd', 'mt', 'nh', 'de', 'sd', 'ri', 'ne', 'ak', 'nm', 'me', 'dc',
'hi', 'wv', 'id', 'ia', 'ks', 'ar', 'mn', 'ms', 'or', 'co', 'ky', 'nv', 'md',
'al', 'tn', 'ct', 'wi', 'ut', 'va', 'wa', 'ma', 'nj', 'az', 'la', 'ok', 'in',
'mo', 'oh', 'pa', 'mi', 'ga', 'sc', 'il', 'nc', 'fl', 'tx', 'ny', 'ca']
Shape of matrix after one hot encodig  (10000, 51)
```

Observation: The 'school_state' column of the DonorsChoose (train.csv) data has categorical data and has been preprocessed to yield words in lowercase under the new column 'school_categories'.

In [204]:
```
project_data = pd.read_csv('train_data.csv')
# 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-fro
m-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string
-in-python
prefix_catogories = list(project_data['teacher_prefix'].values)
# print(type(prefix_catogories))
prefix_list = []
for sent in prefix_catogories:
    sent = re.sub('[^A-Za-z0-9]+', ' ', str(sent))
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split())
    prefix_list.append(sent.lower().strip())
# print(prefix_list)
project_data['prefix_catogories'] = prefix_list
```

In [205]:
```
project_data.drop(['teacher_prefix'], axis=1, inplace=True)
print(project_data.head(2))

# count of all the words in corpus python: https://stackoverflow.com/a/22898595
/4084039
my_counter_prefix = Counter()
for word in project_data['prefix_catogories'].values:
    my_counter_prefix.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
prefix_dict = dict(my_counter_prefix)
sorted_prefix_dict = dict(sorted(prefix_dict.items(), key=lambda kv: kv[1]))


ind2 = np.arange(len(sorted_prefix_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind2, list(sorted_prefix_dict.values()))
plt.xlabel('Prefixes')
plt.ylabel('Projects')
plt.title('% of projects aproved prefixes wise')
plt.xticks(ind2, list(sorted_prefix_dict.keys()))
plt.show()

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

vectorizer = CountVectorizer(vocabulary=list(sorted_prefix_dict.keys()), lowerc
ase=False, binary=True)
vectorizer.fit(project_data['prefix_catogories'].values)
print(vectorizer.get_feature_names())

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

```
   Unnamed: 0       id                         teacher_id school_state  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           IN
1      140945  p258326  897464ce9ddc600bced1151f324dd63a           FL

  project_submitted_datetime project_grade_category  \
0           05-12-2016 13:43          Grades PreK-2
1           25-10-2016 09:22             Grades 6-8

         project_subject_categories     project_subject_subcategories  \
0                Literacy & Language                       ESL, Literacy
1  History & Civics, Health & Sports  Civics & Government, Team Sports

                                  project_title  \
0  Educational Support for English Learners at Home
1            Wanted: Projector for Hungry Learners

                                  project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...

                                  project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...             NaN
1  The projector we need for our school is very c...             NaN

  project_essay_4                         project_resource_summary  \
0             NaN  My students need opportunities to practice beg...
1             NaN  My students need a projector to help with view...

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0
1                                             7                    1

  prefix_catogories
0               mrs
1                mr
```

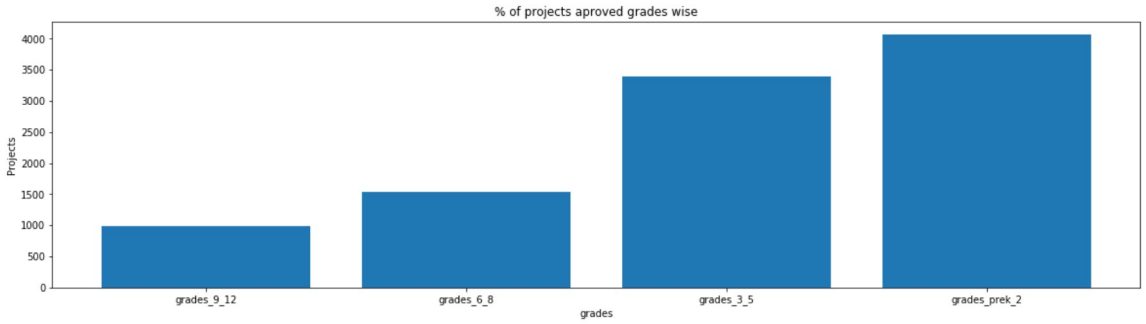



```
nan                 :         1
teacher             :       226
mr                  :      1006
ms                  :      3647
mrs                 :      5120
['nan', 'teacher', 'mr', 'ms', 'mrs']
Shape of matrix after one hot encodig  (10000, 5)
```

Observation: The 'teacher_prefix' column of the DonorsChoose (train.csv) data has categorical data and has been preprocessed to yield words in lowercase and without special characters under the new column 'prefix_catogories'.

In [206]:
```
project_data = pd.read_csv('train_data.csv')
grade_catogories = list(project_data['project_grade_category'].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-fro
m-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string
-in-python
grade_list = []
for sent in grade_catogories:
    sent = sent.replace('-','_')
    sent = sent.replace(' ','_')
    # sent = re.sub('[^A-Za-z0-9]+', ' ', str(sent))
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split())
    grade_list.append(sent.lower().strip())

# temp = temp.replace('-','_')
project_data['new_grade_category'] = grade_list

project_data.drop(['project_grade_category'], axis=1, inplace=True)
print(project_data.head(2))

# count of all the words in corpus python: https://stackoverflow.com/a/22898595
/4084039
my_counter_grade = Counter()
for word in project_data['new_grade_category'].values:
    my_counter_grade.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
grade_dict = dict(my_counter_grade)
sorted_grade_dict = dict(sorted(grade_dict.items(), key=lambda kv: kv[1]))


ind3 = np.arange(len(sorted_grade_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind3, list(sorted_grade_dict.values()))
plt.xlabel('grades')
plt.ylabel('Projects')
plt.title('% of projects aproved grades wise')
plt.xticks(ind3, list(sorted_grade_dict.keys()))
plt.show()

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

vectorizer = CountVectorizer(vocabulary=list(sorted_grade_dict.keys()), lowerca
se=False, binary=True)
vectorizer.fit(project_data['new_grade_category'].values)
print(vectorizer.get_feature_names())

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

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

  school_state project_submitted_datetime         project_subject_categories
\
0           IN           05-12-2016 13:43                Literacy & Language
1           FL           25-10-2016 09:22  History & Civics, Health & Sports

      project_subject_subcategories  \
0                       ESL, Literacy
1  Civics & Government, Team Sports

                                      project_title  \
0  Educational Support for English Learners at Home
1            Wanted: Projector for Hungry Learners

                                      project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...

                                      project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...             NaN
1  The projector we need for our school is very c...             NaN

  project_essay_4                          project_resource_summary  \
0             NaN  My students need opportunities to practice beg...
1             NaN  My students need a projector to help with view...

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0
1                                             7                    1

  new_grade_category
0      grades_prek_2
1         grades_6_8
```




```
grades_9_12          :       995
grades_6_8           :      1546
grades_3_5           :      3390
grades_prek_2        :      4069
['grades_9_12', 'grades_6_8', 'grades_3_5', 'grades_prek_2']
Shape of matrix after one hot encodig  (10000, 4)
```

Observation: The 'project_grade*category' column of the DonorsChoose (train.csv) data has categorical data and has been preprocessed to yield words in lowercase and without special characters(' ' and '-' replaced with '*) under the new column 'new_grade_category' making every grade a whole word.

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [207]:
```
# We are considering only the words which appeared in at least 10 documents(row
s or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encoding ",text_bow.shape)
```

```
Shape of matrix after one hot encoding  (10000, 6213)
```

1.4.2.2 Bag of Words on `project_title`

In [208]:
```
# Similarly you can vectorize for title also
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix (title) after one hot encoding ",title_bow.shape)
```

```
Shape of matrix (title) after one hot encoding  (10000, 671)
```

Observation: The earlier created 'preprocessed_titles' column of the DonorsChoose (train.csv) data has preprocessed text data and has been vectorized with 'Bag of Words' model.

1.4.2.3 TFIDF vectorizer

In [209]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encoding ",text_tfidf.shape)
```

```
Shape of matrix after one hot encoding  (10000, 6213)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [210]:
```
# Similarly you can vectorize for title also
vectorizer = TfidfVectorizer(min_df=10)
title_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix(title) after one hot encoding ",title_tfidf.shape)
```

```
Shape of matrix(title) after one hot encoding  (10000, 671)
```

Observation: The earlier created 'preprocessed_titles' column of the DonorsChoose (train.csv) data with preprocessed text data has been vectorized with 'TfidfVectorizer' model.

1.4.2.5 Using Pretrained Models: Avg W2V

In [211]:
```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
from tqdm import tqdm

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



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

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

# ============================
'''

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

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

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

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


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

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

```
Loading Glove Model

1917495it [06:46, 4718.74it/s]

Done. 1917495  words loaded!
all the words in the corpus 1556208
the unique words in the corpus 23219
The number of words that are present in both glove vectors and our corpus 2232
4 ( 96.145 %)
word 2 vec length 22324
```

In [212]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-
to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [213]:
```
# 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%|██████████████████████████████████████████████████████████████████████████
██| 10000/10000 [00:08<00:00, 1138.39it/s]

10000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [214]:
```
# Similarly you can vectorize for title also
# compute average word2vec for each title.
avg_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector_title = np.zeros(300) # as word vectors are of zero length
    cnt_title_words =0; # num of words with a valid vector in the sentence/revi
ew
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector_title += model[word]
            cnt_title_words += 1
    if cnt_words != 0:
        vector_title /= cnt_title_words
    avg_w2v_vectors_title.append(vector_title)

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

```
100%|██████████████████████████████████████████████████████████████████████████
| 10000/10000 [00:00<00:00, 17375.97it/s]

10000
300
```

Observation: The earlier created 'preprocessed_titles' column of the DonorsChoose (train.csv) data with preprocessed text data has been vectorized using pretrained - glove_words, 'Average Word to Vector/AVGW2V' model.

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [215]:
```
# 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 [216]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in thi
s 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 va
lue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

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

```
100%|██████████████████████████████████████████████████████████████████████
██| 10000/10000 [00:49<00:00, 200.21it/s]

10000
300
```

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

In [217]:
```
# Similarly you can vectorize for title also
tfidf_model_title = TfidfVectorizer()
tfidf_model_title.fit(preprocessed_titles)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_title.get_feature_names(), list(tfidf_model_t
itle.idf_)))
tfidf_words_title = set(tfidf_model_title.get_feature_names())

# compute tfidf word2vec for each title.
tfidf_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored
in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector_title = 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 va
lue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector_title += (vector_title * tf_idf) # calculating tfidf weighte
d w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector_title /= tf_idf_weight
    tfidf_w2v_vectors_title.append(vector_title)

print(len(tfidf_w2v_vectors_title))
print(len(tfidf_w2v_vectors_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████
| 10000/10000 [00:00<00:00, 13685.97it/s]

10000
300
```

Observation: The earlier created 'preprocessed_titles' column of the DonorsChoose (train.csv) data with preprocessed text data has been vectorized using pretrained - glove_words, 'TFIDF weighted Word to Vector/TFIDF weighted W2V' model.

1.4.3 Vectorizing Numerical features

In [218]:
```
# 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_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
approved_price = project_data[project_data['project_is_approved']==1]['price'].values
rejected_price = project_data[project_data['project_is_approved']==0]['price'].values

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 : 317.3365454545455, Standard deviation : 321.254046099561
```

In [219]:
```
price_standardized
```

Out[219]:
```
array([[nan],
       [nan],
       [nan],
       ...,
       [nan],
       [nan],
       [nan]])
```

In [220]:
```
# 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
previously_posted_projects_scalar = StandardScaler()
previously_posted_projects_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
print(f"Mean : {previously_posted_projects_scalar.mean_[0]}, Standard deviation : {np.sqrt(previously_posted_projects_scalar.var_[0])}")

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

```
Mean : 11.2092, Standard deviation : 27.9321398278041
```

In [221]:
```
previously_posted_projects_standardized
```

Out[221]:
```
array([[-0.40130116],
       [-0.15069379],
       [-0.3655001 ],
       ...,
       [-0.25809695],
       [-0.18649484],
       [-0.22229589]])
```

Observation: The 'teacher_number_of_previously_posted_projects' column of the DonorsChoose (train.csv) data is a numerical feature and has been vectorized using StandardScalar, thus removing the mean value of the column and scaling it to unit variance.

1.4.4 Merging all the above features

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

In [222]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(10000, 9)
(10000, 30)
(10000, 6213)
(10000, 1)
```

In [223]:
```
print(title_bow.shape)
print(grade_one_hot.shape)
print(prefix_one_hot.shape)
print(sch_one_hot.shape)
print(previously_posted_projects_standardized.shape)
```

```
(10000, 671)
(10000, 4)
(10000, 5)
(10000, 51)
(10000, 1)
```

In [224]:
```
# 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 dens
e matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standar
dized))
X.shape
```

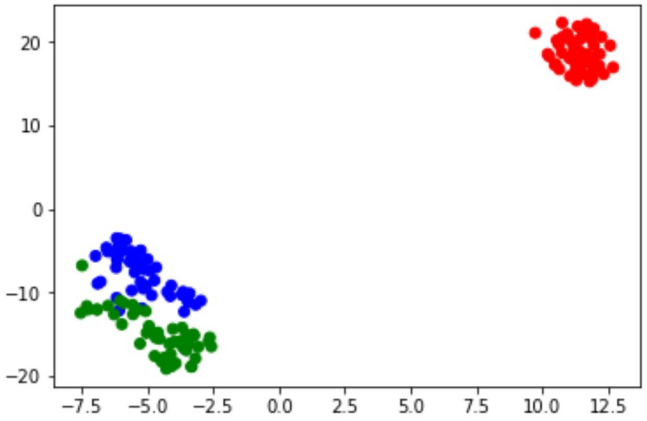
Out[224]: (10000, 6253)

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 [225]:
```
# 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_transfo
rm(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()
```

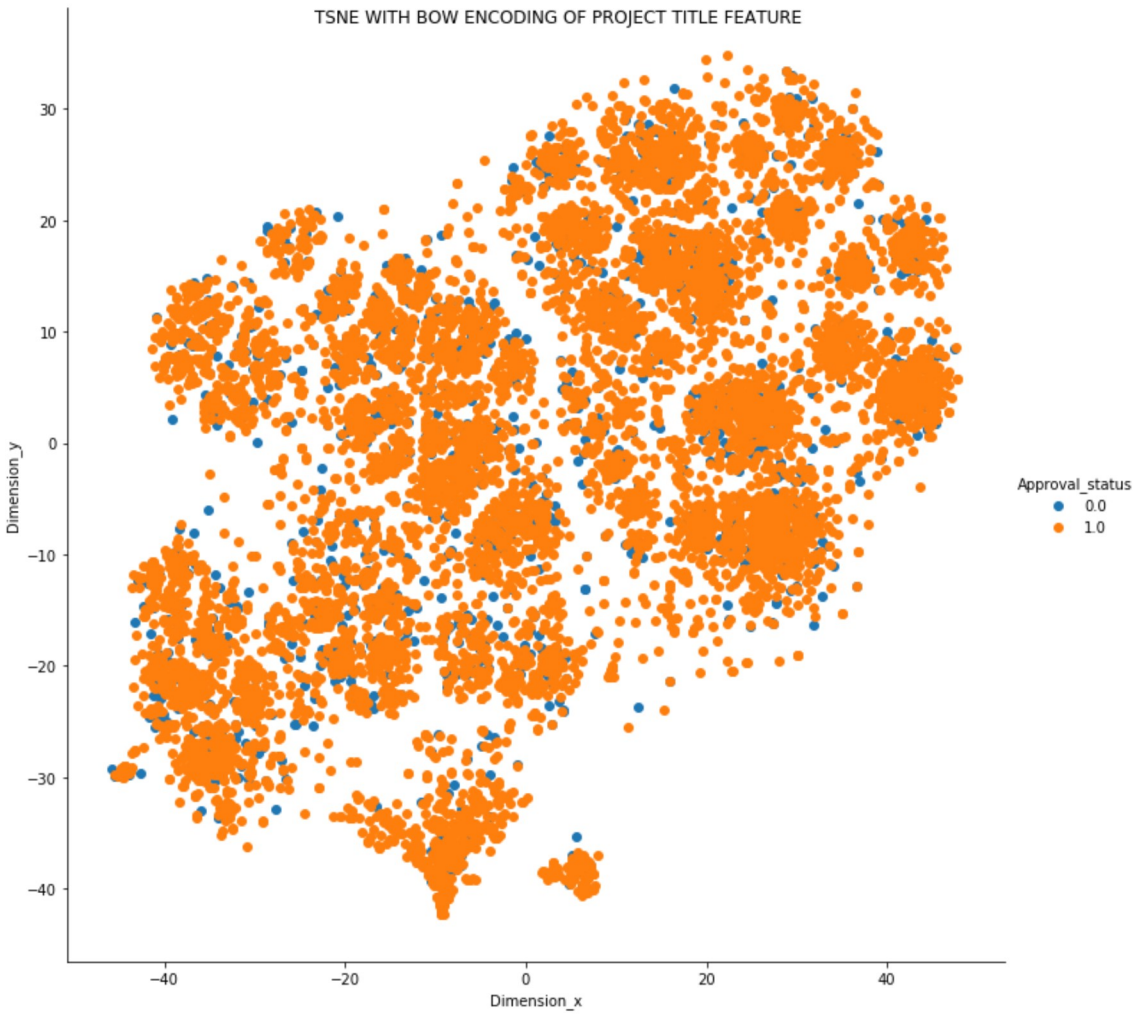


2.1 TSNE with `BOW` encoding of `project_title` feature

In [226]:
```
### please write all of the code with proper documentation and proper titles fo
r each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the read
er
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

# 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 dens
e matirx :)
Y1 = hstack((categories_one_hot, sub_categories_one_hot, price_standardized,tit
le_bow, grade_one_hot, prefix_one_hot, sch_one_hot, previously_posted_projects_
standardized))
Y1.shape
```

Out[226]:
```
(10000, 772)
```

In [227]:
```
# https://scikit-learn.org/stable/modules/impute.html
import scipy.sparse as sp
from sklearn.impute import SimpleImputer
# X = sp.csc_matrix([[1, 2], [0, -1], [8, 4]])
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(Y1)
SimpleImputer(copy=True, fill_value=None, missing_values=np.nan, strategy='medi
an', verbose=0)
Y1 = imp.transform(Y1)
Y1 = Y1.toarray()
# print(type(Y1))
# print(Y1)

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200, random_state =
0)
Y_embedding = tsne.fit_transform(Y1)

# print(Y_embedding)

# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transfo
rm(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix
t = project_data['project_is_approved']
t1=t[0:]
# print(type(t1))


for_tsne1 = np.vstack((Y_embedding.T, t1)).T
for_tsne1_df = pd.DataFrame(data=for_tsne1, columns=['Dimension_x','Dimension_y
','Approval_status'])
sns.FacetGrid(for_tsne1_df, hue = "Approval_status", height = 10).map(plt.scatt
er, "Dimension_x", "Dimension_y").add_legend().fig.suptitle("TSNE WITH BOW ENCO
DING OF PROJECT TITLE FEATURE ")
plt.show()
```



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

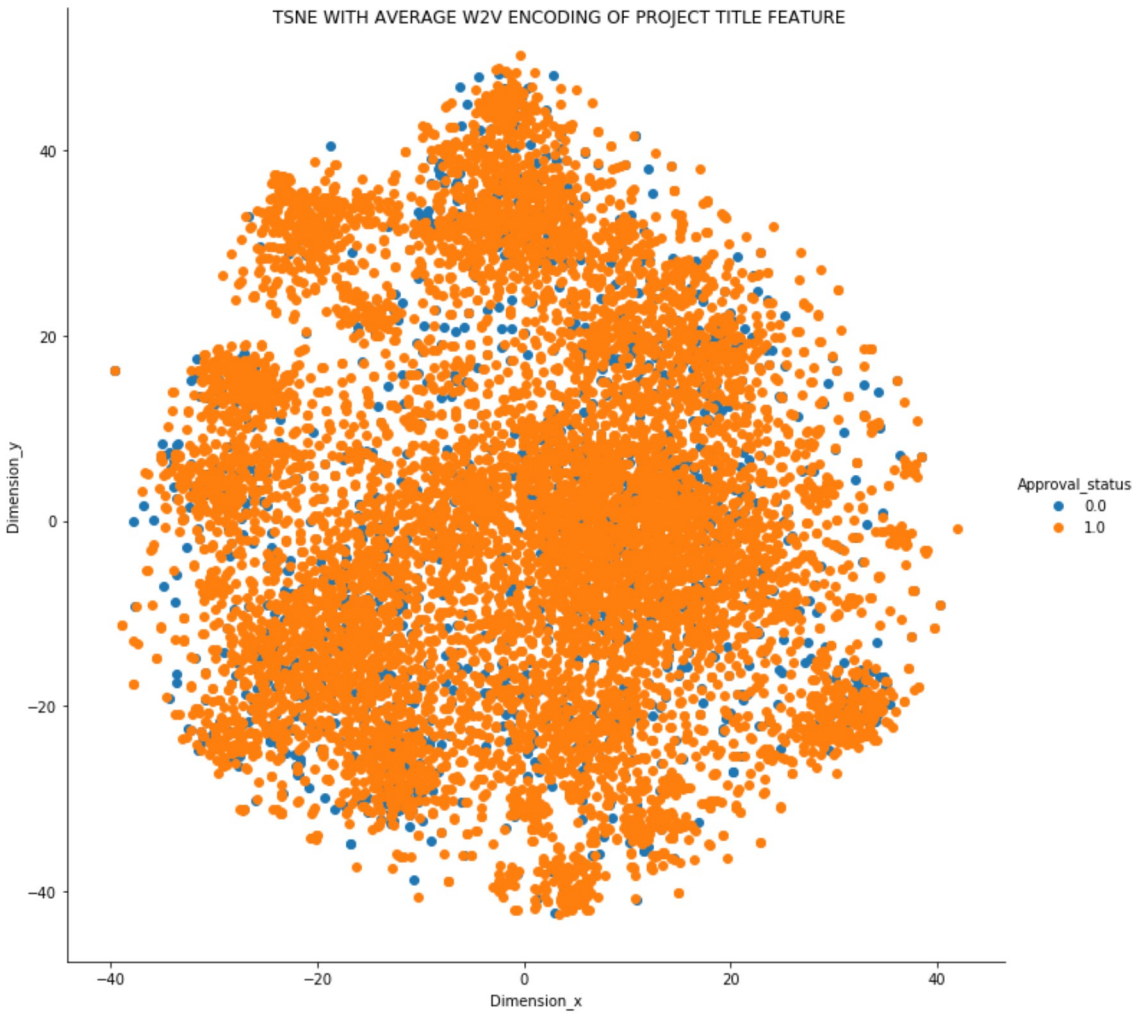
In [228]:

```
# https://scikit-learn.org/stable/modules/impute.html
import scipy.sparse as sp
from sklearn.impute import SimpleImputer
# X = sp.csc_matrix([[1, 2], [0, -1], [8, 4]])

Y_tf = hstack((categories_one_hot, sub_categories_one_hot, price_standardized,t
itle_tfidf, grade_one_hot, prefix_one_hot, sch_one_hot, previously_posted_proje
cts_standardized))
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(Y_tf)
SimpleImputer(copy=True, fill_value=None, missing_values=np.nan, strategy='medi
an', verbose=0)
Y_tf = imp.transform(Y_tf)
Y_tf = Y_tf.toarray()
# print(type(Y1))
# print(Y1)

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200, random_state =
0)
Y_embedding = tsne.fit_transform(Y_tf)

# print(Y_embedding)

# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transfo
rm(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix
t = project_data['project_is_approved']
t1=t[0:]
# print(type(t1))


for_tsne1 = np.vstack((Y_embedding.T, t1)).T
for_tsne1_df = pd.DataFrame(data=for_tsne1, columns=['Dimension_x','Dimension_y
','Approval_status'])
sns.FacetGrid(for_tsne1_df, hue = "Approval_status", height = 10).map(plt.scatt
er, "Dimension_x", "Dimension_y").add_legend().fig.suptitle("TSNE WITH TFIDF EN
CODING OF PROJECT TITLE FEATURE ")
plt.show()
```

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

In [229]:

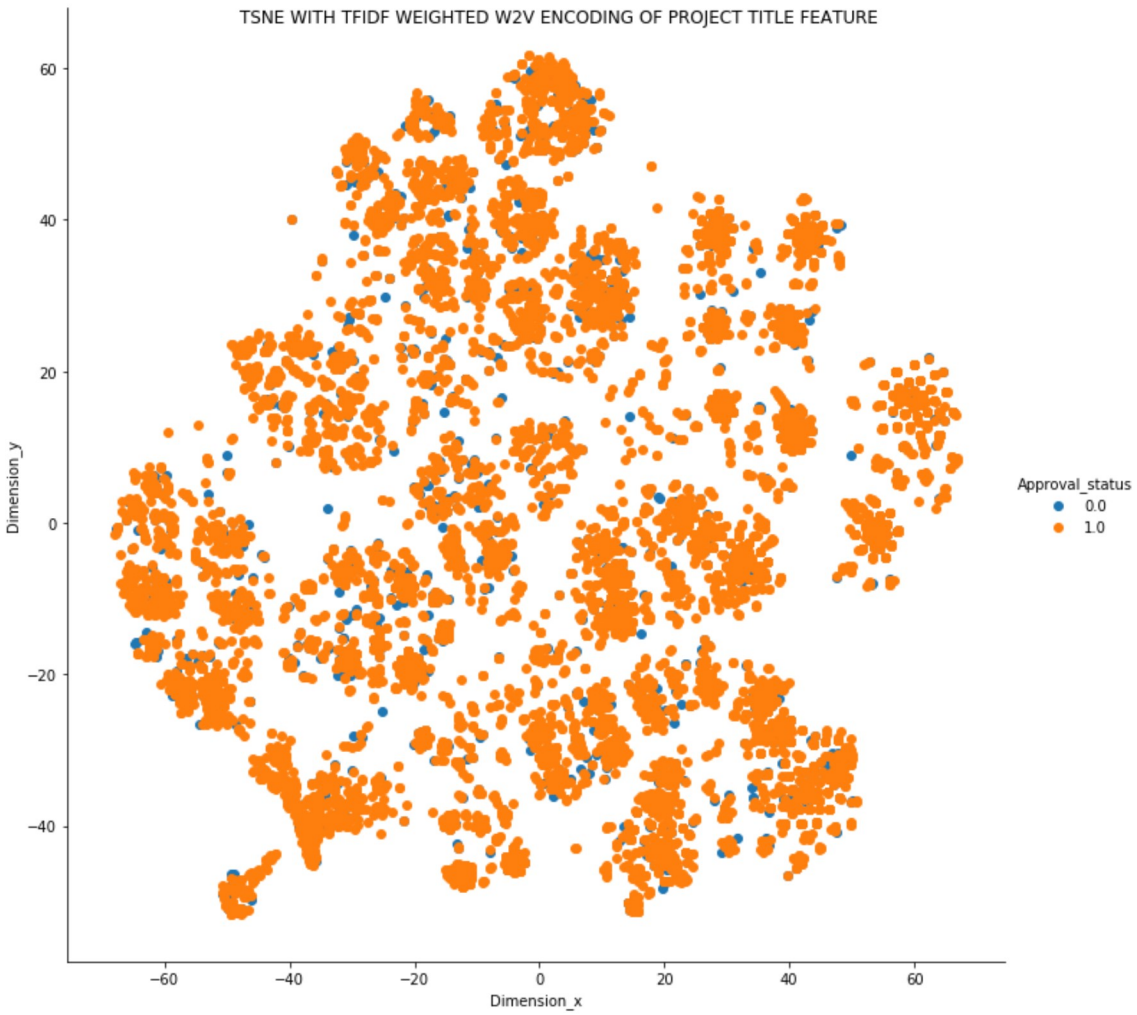
```
# https://scikit-learn.org/stable/modules/impute.html
import scipy.sparse as sp
from sklearn.impute import SimpleImputer
# X = sp.csc_matrix([[1, 2], [0, -1], [8, 4]])

Y_avg_w2v = hstack((categories_one_hot, sub_categories_one_hot, price_standardi
zed,avg_w2v_vectors_title, grade_one_hot, prefix_one_hot, sch_one_hot, previous
ly_posted_projects_standardized))
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(Y_avg_w2v)
SimpleImputer(copy=True, fill_value=None, missing_values=np.nan, strategy='medi
an', verbose=0)
Y_avg_w2v = imp.transform(Y_avg_w2v)
Y_avg_w2v = Y_avg_w2v.toarray()
# print(type(Y1))
# print(Y1)

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200, random_state =
0)
Y_embedding = tsne.fit_transform(Y_avg_w2v)

# print(Y_embedding)

# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transfo
rm(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix
t = project_data['project_is_approved']
t1=t[0:]
# print(type(t1))


for_tsne1 = np.vstack((Y_embedding.T, t1)).T
for_tsne1_df = pd.DataFrame(data=for_tsne1, columns=['Dimension_x','Dimension_y
','Approval_status'])
sns.FacetGrid(for_tsne1_df, hue = "Approval_status", height = 10).map(plt.scatt
er, "Dimension_x", "Dimension_y").add_legend().fig.suptitle("TSNE WITH AVERAGE
W2V ENCODING OF PROJECT TITLE FEATURE ")
plt.show()
```



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

In [230]:

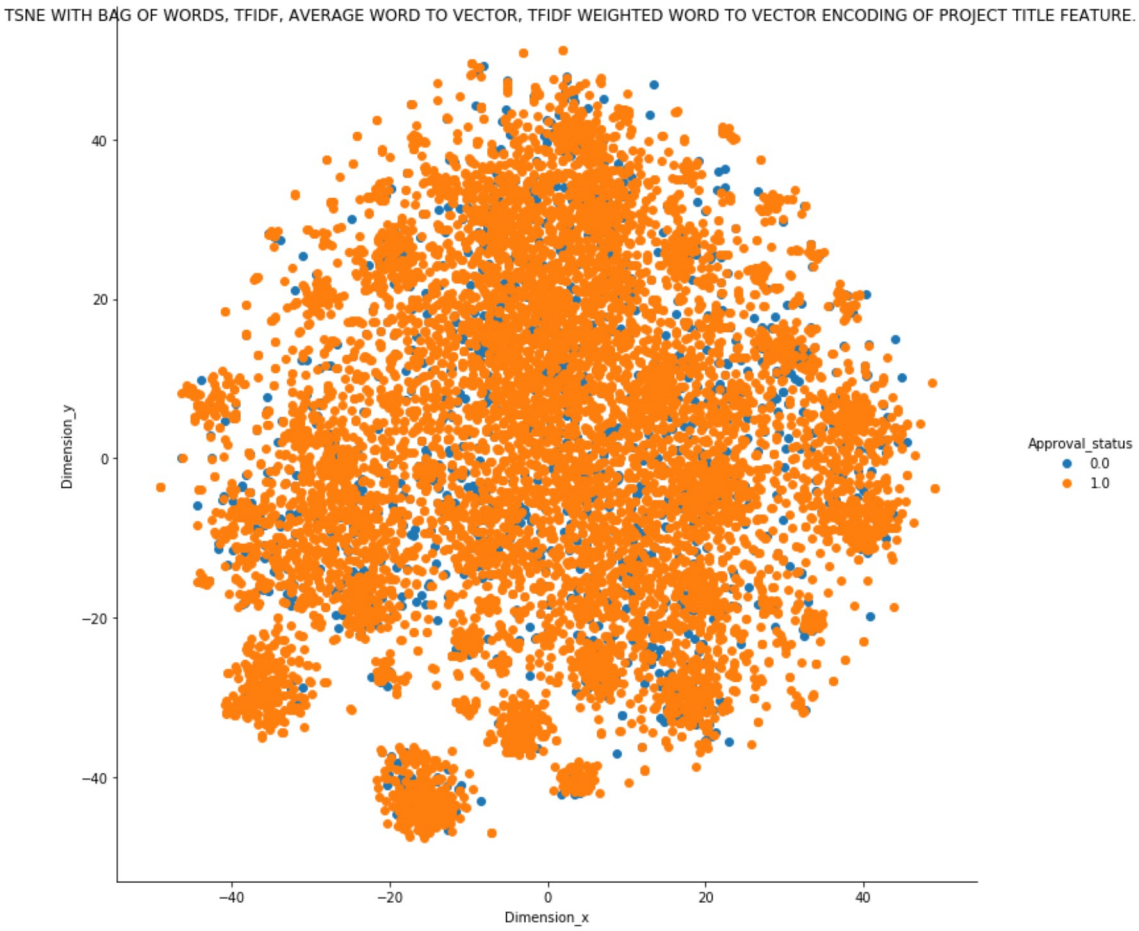
```
# https://scikit-learn.org/stable/modules/impute.html
import scipy.sparse as sp
from sklearn.impute import SimpleImputer
# X = sp.csc_matrix([[1, 2], [0, -1], [8, 4]])

Y_tfidf_w2v = hstack((categories_one_hot, sub_categories_one_hot, price_standar
dized,tfidf_w2v_vectors_title, grade_one_hot, prefix_one_hot, sch_one_hot, prev
iously_posted_projects_standardized))
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(Y_tfidf_w2v)
SimpleImputer(copy=True, fill_value=None, missing_values=np.nan, strategy='medi
an', verbose=0)
Y_tfidf_w2v = imp.transform(Y_tfidf_w2v)
Y_tfidf_w2v = Y_tfidf_w2v.toarray()
# print(type(Y1))
# print(Y1)

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200, random_state =
0)
Y_embedding = tsne.fit_transform(Y_tfidf_w2v)

# print(Y_embedding)

# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transfo
rm(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix
t = project_data['project_is_approved']
t1=t[0:]
# print(type(t1))


for_tsne1 = np.vstack((Y_embedding.T, t1)).T
for_tsne1_df = pd.DataFrame(data=for_tsne1, columns=['Dimension_x','Dimension_y
','Approval_status'])
sns.FacetGrid(for_tsne1_df, hue = "Approval_status", height = 10).map(plt.scatt
er, "Dimension_x", "Dimension_y").add_legend().fig.suptitle("TSNE WITH TFIDF WE
IGHTED W2V ENCODING OF PROJECT TITLE FEATURE ")
plt.show()
```



TSNE WITH BAG OF WORDS, TFIDF, AVERAGE WORD TO VECTOR, TFIDF WEIGHTED WORD TO VECTOR ENCODING OF PROJECT TITLE FEATURE.

In [231]:

```
# https://scikit-learn.org/stable/modules/impute.html
import scipy.sparse as sp
from sklearn.impute import SimpleImputer
# X = sp.csc_matrix([[1, 2], [0, -1], [8, 4]])

Y_all = hstack((categories_one_hot, sub_categories_one_hot, price_standardized,
title_bow, title_tfidf, avg_w2v_vectors_title, tfidf_w2v_vectors_title, grade_o
ne_hot, prefix_one_hot, sch_one_hot, previously_posted_projects_standardized))
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(Y_all)
SimpleImputer(copy=True, fill_value=None, missing_values=np.nan, strategy='medi
an', verbose=0)
Y_all = imp.transform(Y_all)
Y_all = Y_all.toarray()
# print(type(Y1))
# print(Y1)

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200, random_state =
0)
Y_embedding = tsne.fit_transform(Y_all)

# print(Y_embedding)

# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transfo
rm(x.toarray()) , .toarray() will convert the sparse matrix into dense matrix
t = project_data['project_is_approved']
t1=t[0:]
# print(type(t1))


for_tsne1 = np.vstack((Y_embedding.T, t1)).T
for_tsne1_df = pd.DataFrame(data=for_tsne1, columns=['Dimension_x','Dimension_y
','Approval_status'])
sns.FacetGrid(for_tsne1_df, hue = "Approval_status", height = 10).map(plt.scatt
er, "Dimension_x", "Dimension_y").add_legend().fig.suptitle("TSNE WITH BAG OF W
ORDS, TFIDF, AVERAGE WORD TO VECTOR, TFIDF WEIGHTED WORD TO VECTOR ENCODING OF
PROJECT TITLE FEATURE.")
plt.show()
```



2.5 Summary

TSNE plot is plotted for the categorical, numerical features obtained after preprocessing with project_title(BOW), project_title(TFIDF), project_title(AVG W2V), project_title(TFIDF W2V) individually and all 4 combined. Overall since the approved and rejected values are scattered and overlapping over eachother, there is no distinction/pattern to be observed to separate the approved and rejected status. In no plots the approved and rejected status are clustered separately, thus making it impossible to find the acceptance/rejectance pattern.