DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

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

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

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

In [4]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

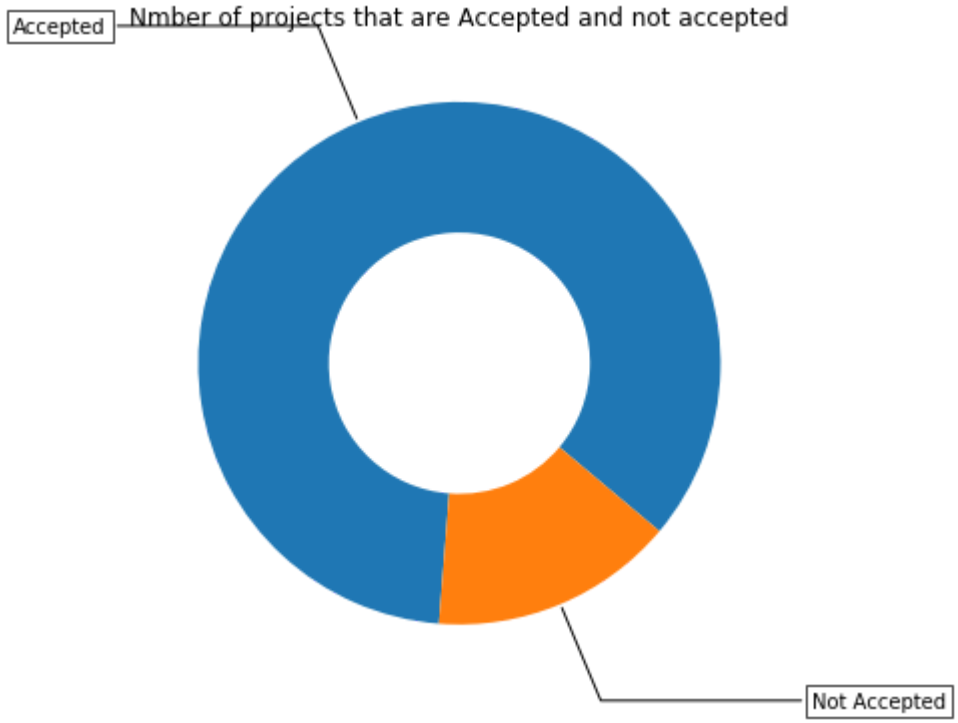
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 Data Analysis

In [5]:
```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html
#sphx-glr-gallery-pie-and-polar-charts-pie-and-donut-labels-py


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

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

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

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

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

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

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

plt.show()
```

```
Number of projects thar are approved for funding  92706 , ( 84.85830404217927
%)
Number of projects thar are not approved for funding  16542 , ( 15.1416959578
20739 %)
```



Summary

From the past data we can say that, almost 85% projects are getting accepted and remaining 15% projects are rejected. 17:3 ratio for accepted and not accepted.

In [6]:
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/40
84039

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 (thin
k 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(188,189,2
20)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,3
9,143)']]

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[6]:
```
'\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'
```

1.2.1 Univariate Analysis: School State

In [7]:
```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.pdf
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
print('='*50)
print("States with highest % approvals")
print(temp.tail(5))
```

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.800000
7          DC       0.802326
43         TX       0.813142
26         MT       0.816327
18         LA       0.831245
==================================================
States with highest % approvals
   state_code  num_proposals
30         NH       0.873563
35         OH       0.875152
47         WA       0.876178
28         ND       0.888112
8          DE       0.897959
```

Summary

Above data represent highest and lowest % of approval from number of requested projects done every states in USA. From this data we can say that there is not much difference between highest

and lowest approval, **10% approx**. Delaware has highest approval of 89% and Vermont has least 80% approval.

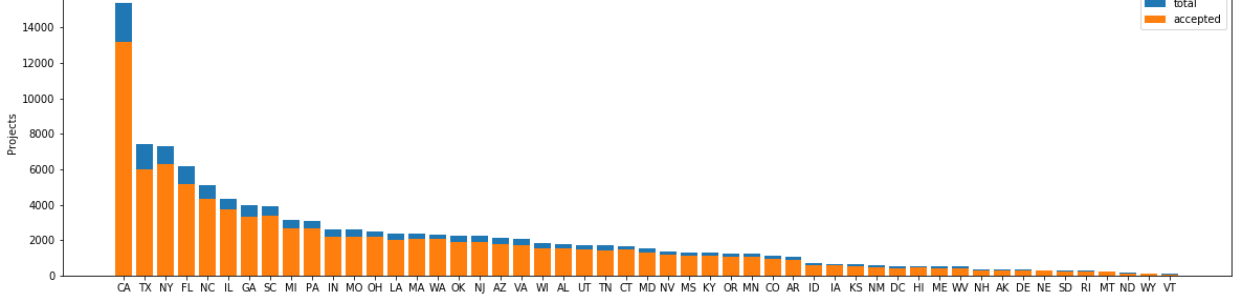
In [8]:
```
 # try to plot PDF to get idea about % approvals and states but fail to determi
ne some ooutcomes
    # code taken by my EDA assignment.
"""
sns.FacetGrid(project_data, hue="school_state", height=6)\
.map(sns.distplot, "project_is_approved")\
.add_legend();
plt.show();"""
```

Out[8]:
```
'\nsns.FacetGrid(project_data, hue="school_state", height=6).map(sns.distplo
t, "project_is_approved").add_legend();\nplt.show();'
```

In [9]:
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_ma
rkers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    plt.figure(figsize=(20,5))
    p1 = plt.bar(ind, data[col3].values)
    p2 = plt.bar(ind, data[col2].values)

    plt.ylabel('Projects')
    plt.title('Number of projects aproved vs rejected')
    plt.xticks(ind, list(data[xtick].values))
    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()
```

In [10]:
```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51
540521/4084039
    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/19385591/40840
39
    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':'mea
n'})).reset_index()['Avg']

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

    if top:
        temp = temp[0:top]

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

In [11]:
```
univariate_barplots(project_data, 'school_state', 'project_is_approved', False)
```



	school_state	project_is_approved	total	Avg
4	CA	13205	15388	0.858136
43	TX	6014	7396	0.813142
34	NY	6291	7318	0.859661
9	FL	5144	6185	0.831690
27	NC	4353	5091	0.855038

==========

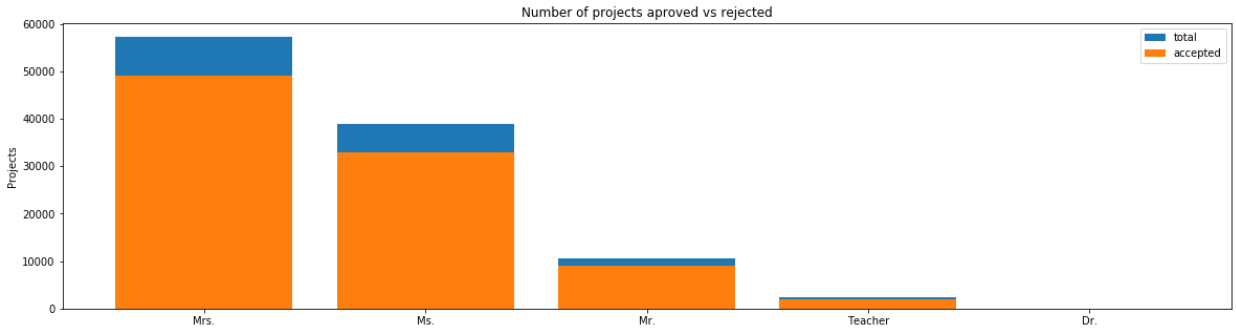
	school_state	project_is_approved	total	Avg
39	RI	243	285	0.852632
26	MT	200	245	0.816327
28	ND	127	143	0.888112
50	WY	82	98	0.836735
46	VT	64	80	0.800000

SUMMARY

Above data illustrate total number of project applied for approval and approved projects by all states. Every state has greater than 80% success rate in approval. With 15388 number of project applications California states is top applicant, and almost double then second most state Texas with (7400 number of application) . Vermont and Wyoming are last on the list with 80 and 98 number of application. There is no reletion with total number application and % of application accepted.

1.2.2 Univariate Analysis: teacher_prefix

In [12]:
```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top
=False)
```



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

==========

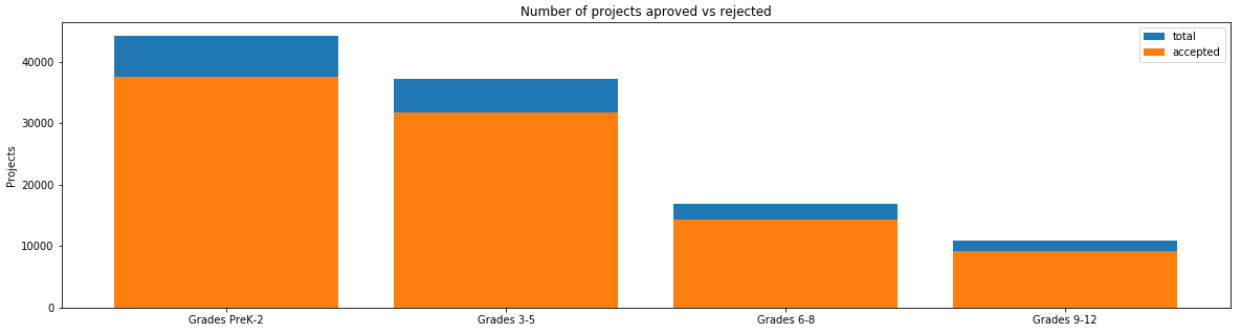
	teacher_prefix	project_is_approved	total	Avg
2	Mrs.	48997	57269	0.855559
3	Ms.	32860	38955	0.843537
1	Mr.	8960	10648	0.841473
4	Teacher	1877	2360	0.795339
0	Dr.	9	13	0.692308

SUMMARY

From above data we can say that women tend to applied more than men almost 82% of total applications. Teacher prefix Mrs has highest number of applications and % of project accepted, 57000 and 85% respectively. Doctorates faculty has applied for very few projects and get low acceptance rate, so as teachers.

1.2.3 Univariate Analysis: project_grade_category

In [13]:
```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', top=False)
```




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

SAMMARY

From above data we can say that PreK - 2 has applied for project the most with 44000 number of projects applied. Second highest application is form grades 3-5 with 37000 projects. Grads 9-12 has applied for least number of projects compare to all other groups of grades. If you divide data in groups of grades then % of acceptance in every groups nearly equal to 84%.

1.2.4 Univariate Analysis: project_subject_categories

In [14]:
```
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 ''(empt
y) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trai
ling spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [15]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(7)
```

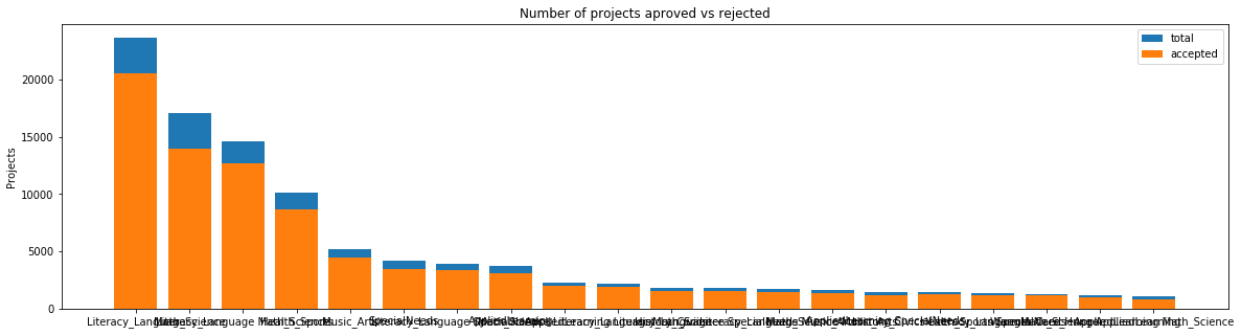
Out[15]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_s
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2
5	141660	p154343	a50a390e8327a95b77b9e495b58b9a6e	Mrs.	FL	2
6	21147	p099819	9b40170bfa65e399981717ee8731efc3	Mrs.	CT	2

Summary

We have did text cleaning here on the **'project_subject_categories'** column. We have replacing **'&'** with **'_'**, replacing **'the'** with **''** (removing 'the'), remove empty space and remove trailing space. Afert cleaning store data in to new column man as **'clean__categories'**.

In [16]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', to
p=20)
"""univariate_barplots(project_data, 'clean_categories', 'project_is_approved',
False)"""
```



	clean_categories	project_is_approved	total	Avg
24	Literacy_Language	20520	23655	0.867470
32	Math_Science	13991	17072	0.819529
28	Literacy_Language Math_Science	12725	14636	0.869432
8	Health_Sports	8640	10177	0.848973
40	Music_Arts	4429	5180	0.855019

==========

	clean_categories	project_is_approved	total	Avg
19	History_Civics Literacy_Language	1271	1421	0.894441
14	Health_Sports SpecialNeeds	1215	1391	0.873472
50	Warmth Care_Hunger	1212	1309	0.925898
33	Math_Science AppliedLearning	1019	1220	0.835246
4	AppliedLearning Math_Science	855	1052	0.812738

Out[16]: "univariate_barplots(project_data, 'clean_categories', 'project_is_approved', False)"

In [17]:
```
"""Code is copied from above. To visualize is there any relation with the proje
ct approval and combine category. """

temp = pd.DataFrame(project_data.groupby("clean_categories")["project_is_approv
ed"].apply(np.mean)).reset_index()
temp.columns = ['clean_categories', 'num_proposals']
temp.sort_values(by=['num_proposals'], inplace=True)
print("clean_categories with lowest % approvals")
print(temp.head(15))
print('='*50)
print("clean_categories with highest % approvals")
print(temp.tail(15))
```

```
clean_categories with lowest % approvals
                        clean_categories  num_proposals
23     History_Civics Warmth Care_Hunger       0.000000
45         Music_Arts Warmth Care_Hunger       0.500000
39       Math_Science Warmth Care_Hunger       0.545455
42              Music_Arts Health_Sports       0.684211
41           Music_Arts AppliedLearning       0.700000
43            Music_Arts History_Civics       0.722222
31  Literacy_Language Warmth Care_Hunger       0.777778
49       SpecialNeeds Warmth Care_Hunger       0.782609
17        History_Civics AppliedLearning       0.785714
47            SpecialNeeds Health_Sports       0.785714
34            Math_Science Health_Sports       0.787440
7     AppliedLearning Warmth Care_Hunger       0.800000
26       Literacy_Language Health_Sports       0.805556
13              Health_Sports Music_Arts       0.806452
5            AppliedLearning Music_Arts       0.807388
==================================================
clean_categories with highest % approvals
                        clean_categories  num_proposals
30     Literacy_Language SpecialNeeds       0.855592
35        Math_Science History_Civics       0.855828
20        History_Civics Math_Science       0.857143
36     Math_Science Literacy_Language       0.859764
3   AppliedLearning Literacy_Language       0.861251
```

```
24                  Literacy_Language       0.867470
28     Literacy_Language Math_Science       0.869432
14         Health_Sports SpecialNeeds       0.873472
44            Music_Arts SpecialNeeds       0.876812
27   Literacy_Language History_Civics       0.877627
10       Health_Sports History_Civics       0.883721
19   History_Civics Literacy_Language       0.894441
18       History_Civics Health_Sports       0.923077
50                 Warmth Care_Hunger       0.925898
15   Health_Sports Warmth Care_Hunger       0.956522
```

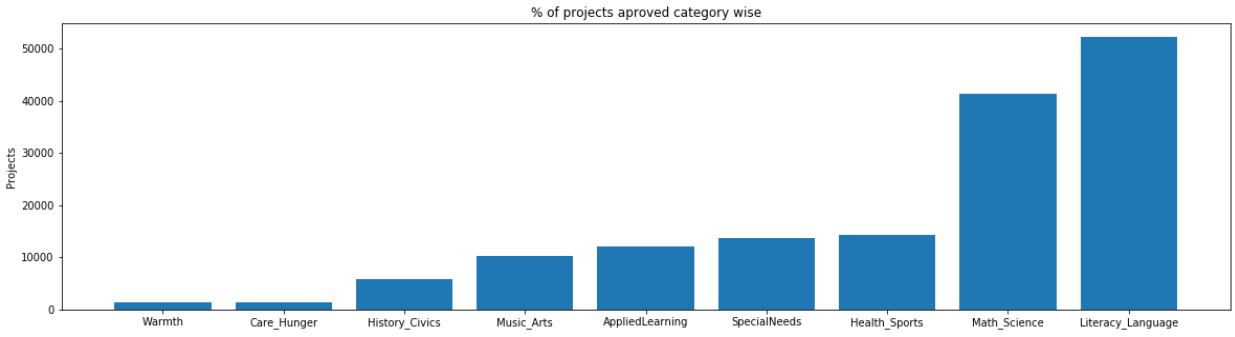
Summary

From the above data we can say that donors are more taking interest if category are **Health_Sports, Warmth, Care_Hunger and Literacy_Language**.

We can't say anything about what category is more acceptable on the base on its combination, for example if Warmth, Care_Hunger category is combine and with Health_Sports has height approvals 92% and 95% respectively, but other than this then Warmth, Care_Hunger has lowest approvals with combination of other categories.

Heights number of projects applied form the category of Literacy_Language , Math_Science and combination of both. Category of Literacy_Language and its combination with other category have approximately 85% of approvals.

In [18]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

# 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 [19]:
```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Warmth               :      1388
Care_Hunger          :      1388
History_Civics       :      5914
```

```
Music_Arts           :     10293
AppliedLearning      :     12135
SpecialNeeds         :     13642
Health_Sports        :     14223
Math_Science         :     41421
Literacy_Language    :     52239
```

Summary

As we have seen above Math_Science, Literacy_Language is the most popular category for the projects and Warmth, Care_Hunger is the least popular category.

1.2.5 Univariate Analysis: project_subject_subcategories

In [20]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.
com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-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 [21]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

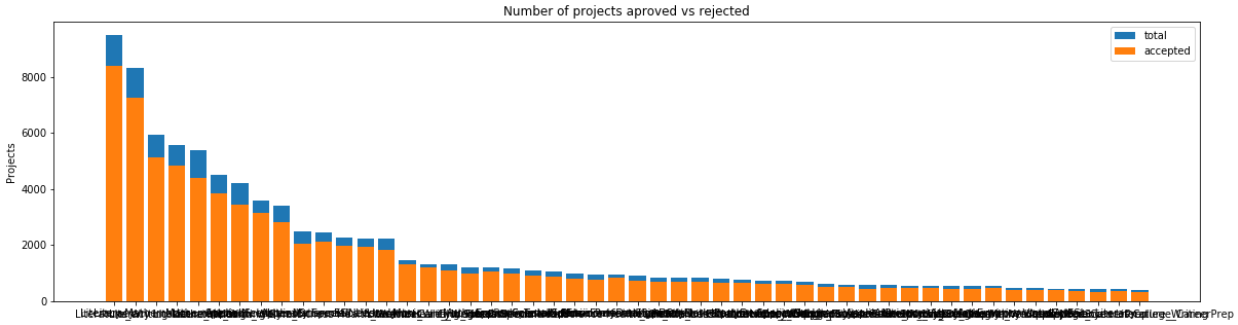
Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_su
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

Summary

Same text pre-processing is performed on project sub-category like removing ‘the’ and trailing space, replacing ‘&’ with ‘_’ and etc.

In [22]:
```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved',
top=50)
```



	clean_subcategories	project_is_approved	total	Avg
317	Literacy	8371	9486	0.882458
319	Literacy Mathematics	7260	8325	0.872072
331	Literature_Writing Mathematics	5140	5923	0.867803
318	Literacy Literature_Writing	4823	5571	0.865733
342	Mathematics	4385	5379	0.815207

==========

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

Summary

Literacy is one of the most popular subcategory of projects along with Literacy in mathematics.

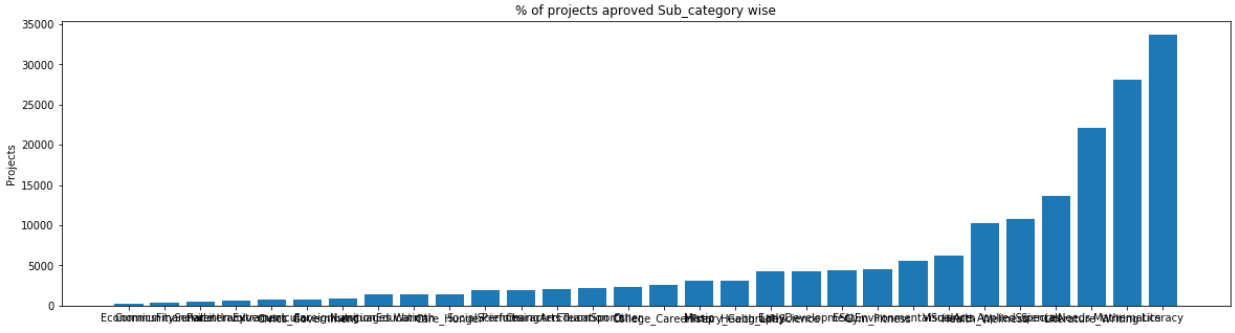
In [23]:
```
"""Code is copied from above. To visualize is there any relation with the proje
ct approval and sub category. """

temp = pd.DataFrame(project_data.groupby("clean_subcategories")["project_is_app
roved"].apply(np.mean)).reset_index()
temp.columns = ['clean_subcategories', 'num_proposals']
temp.sort_values(by=['num_proposals'], inplace=True)
print("clean_subcategories with lowest % approvals")
print(temp.head(15))
print('='*50)
print("clean_subcategories with highest % approvals")
print(temp.tail(15))
```

```
clean_subcategories with lowest % approvals
                       clean_subcategories  num_proposals
316   History_Geography Warmth Care_Hunger       0.000000
129                          ESL Economics       0.000000
120    CommunityService NutritionEducation       0.250000
370                  Other PerformingArts       0.333333
352        Mathematics Warmth Care_Hunger       0.333333
121                CommunityService Other       0.333333
213     Extracurricular Health_LifeScience       0.400000
132                 ESL FinancialLiteracy       0.500000
220     Extracurricular NutritionEducation       0.500000
35            CharacterEducation Economics       0.500000

63       Civics_Government Extracurricular       0.500000
56   CharacterEducation Warmth Care_Hunger       0.500000
106                  CommunityService ESL       0.500000
```

```
28      AppliedSciences Warmth Care_Hunger       0.500000
151             EarlyDevelopment Economics       0.500000
==================================================
clean_subcategories with highest % approvals
                      clean_subcategories  num_proposals
244    ForeignLanguages Health_LifeScience            1.0
119                CommunityService Music             1.0
112           CommunityService Gym_Fitness            1.0
72    Civics_Government NutritionEducation            1.0
73     Civics_Government ParentInvolvement            1.0
215      Extracurricular History_Geography            1.0
211       Extracurricular ForeignLanguages            1.0
111     CommunityService FinancialLiteracy            1.0
363      NutritionEducation SocialSciences            1.0
210      Extracurricular FinancialLiteracy            1.0
142                ESL NutritionEducation             1.0
367  NutritionEducation Warmth Care_Hunger            1.0
237    FinancialLiteracy ParentInvolvement            1.0
123        CommunityService PerformingArts            1.0
206        EnvironmentalScience TeamSports            1.0
```

In [24]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [25]:
```
# 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 Sub_category wise')
plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
plt.show()
```



In [26]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
NutritionEducation   :      1355
```

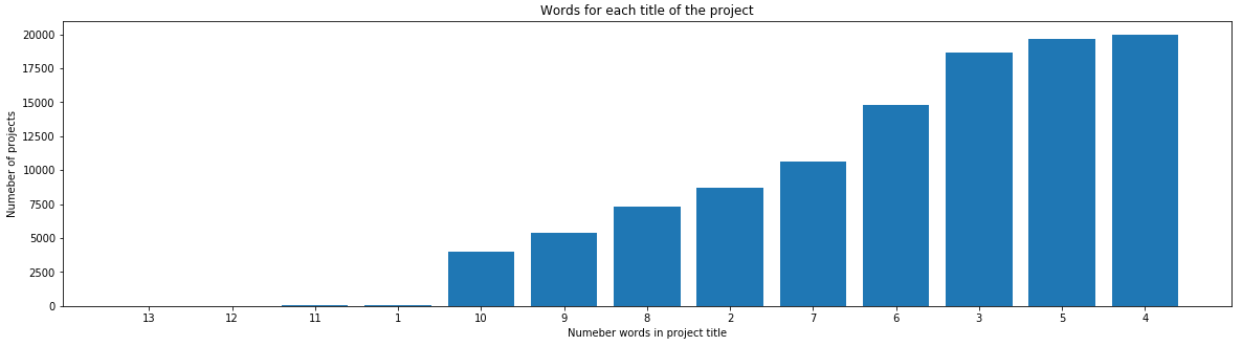
```
Warmth                :      1388
Care_Hunger           :      1388
SocialSciences        :      1920
PerformingArts        :      1961
CharacterEducation    :      2065
TeamSports            :      2192
Other                 :      2372
College_CareerPrep    :      2568
Music                 :      3145
History_Geography     :      3171
Health_LifeScience    :      4235
EarlyDevelopment      :      4254
ESL                   :      4367
Gym_Fitness           :      4509
EnvironmentalScience  :      5591
VisualArts            :      6278
Health_Wellness       :     10234
AppliedSciences       :     10816
SpecialNeeds          :     13642
Literature_Writing    :     22179
Mathematics           :     28074
Literacy              :     33700
```

Summary

From the above data we can say that Literacy, mathematics and literature writing is highest approval sub-category (almost 35% to 40% of total).

1.2.6 Univariate Analysis: Text features (Title)

In [27]:
```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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

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

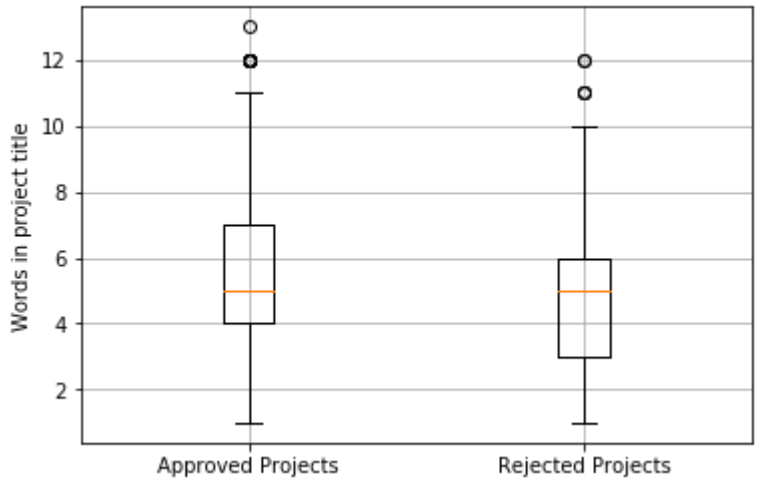


Summary

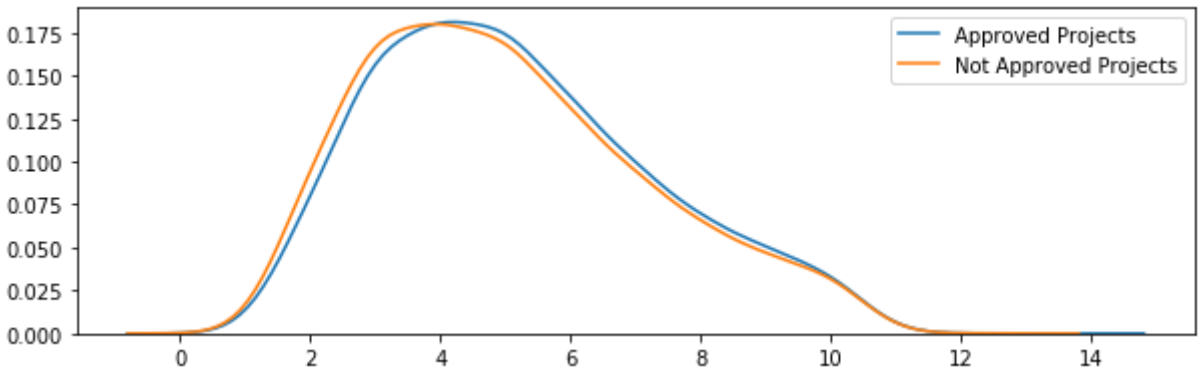
Here we have applied univariate analysis on number of words in each project title. From the graph we can say that majority of project title contain 3 to 5 words. And there are many few projects which have word count from 9 to 13.

In [28]:
```
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 [29]:
```
# 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 [30]:
```
plt.figure(figsize=(10,3))
sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)
sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6)
plt.legend()
plt.show()
```



Summary

From the above box plot data we can say that mean of approved and non-approved project is nearly same but approved projects have slightly more number of words in title. And same thing we can conclude from above PDF that slop of approved projects is slightly ahead of non-approved projects.

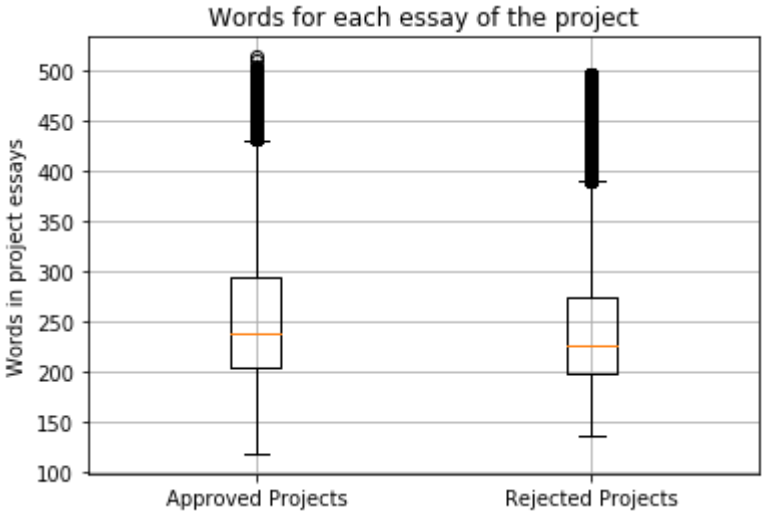
1.2.7 Univariate Analysis: Text features (Project Essay's)

In [31]:
```
# 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)
```

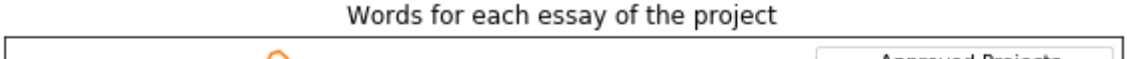
Some project have 4 essays but after May 17, 2016, the number of essays was reduced from 4 to 2, so in above we combine all the essay into one field.

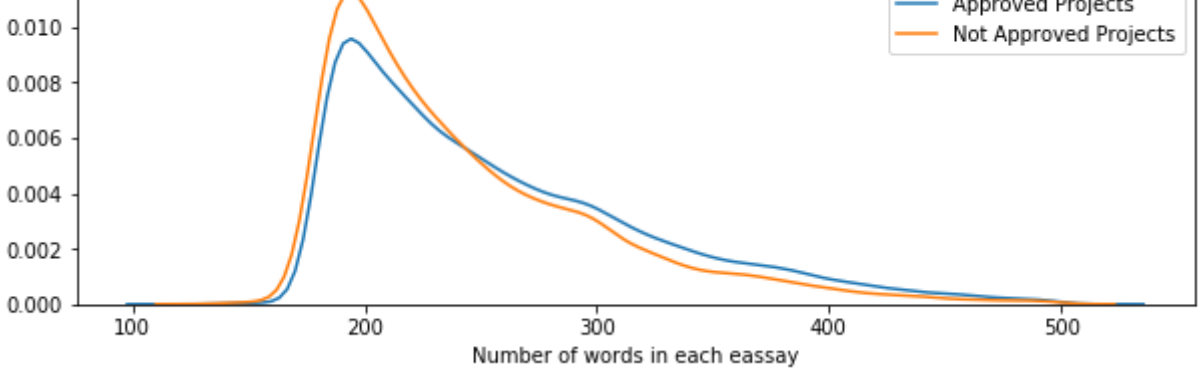
In [32]:
```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.split().apply(len)
approved_word_count = approved_word_count.values

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

In [33]:
```
# 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 [34]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```





Summary

We have preform univariate analysis on the essay of projects. From the above boxplot and PDF we can say that approved project tend to have slightly more number of words in essay then non-approved projects.

1.2.8 Univariate Analysis: Cost per project

In [35]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[35]:

	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 [36]:
```
# 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(5)
```

Out[36]:

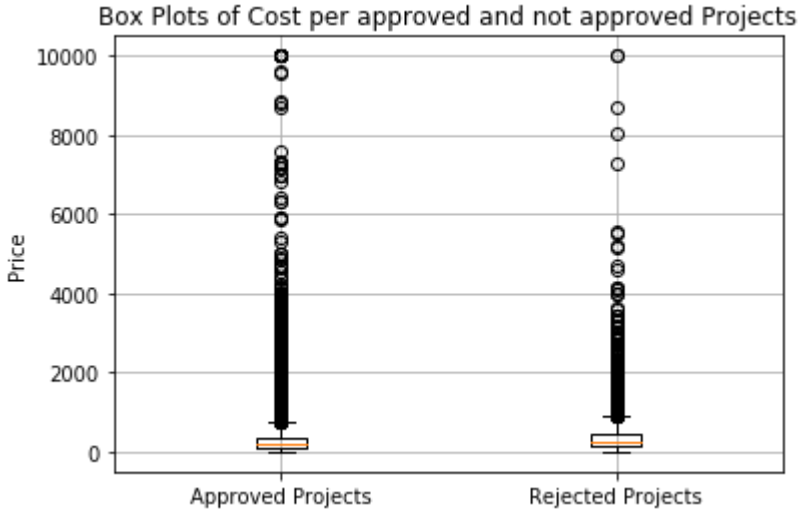
	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98
4	p000005	485.99	8

In [37]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

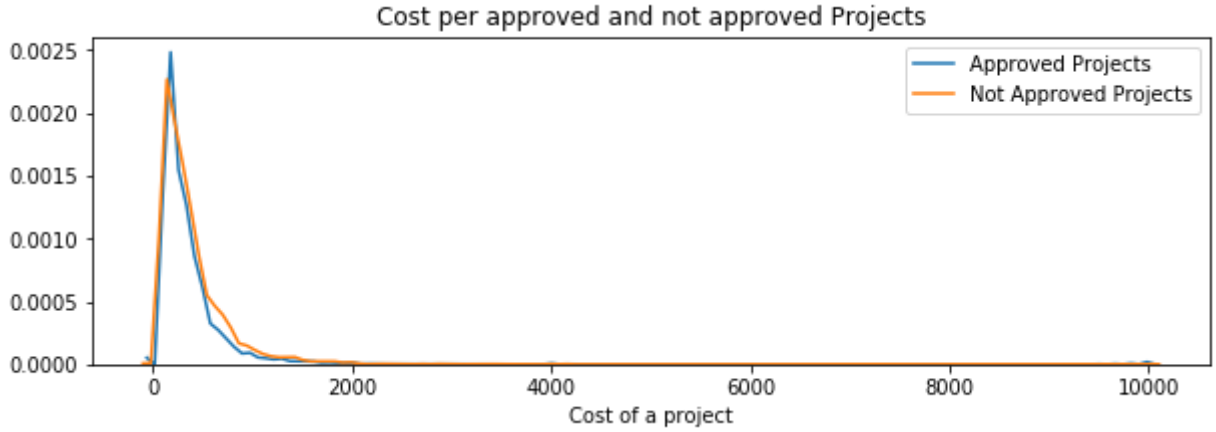
In [38]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].
values

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

In [39]:
```
# 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 [40]:
```
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()
```



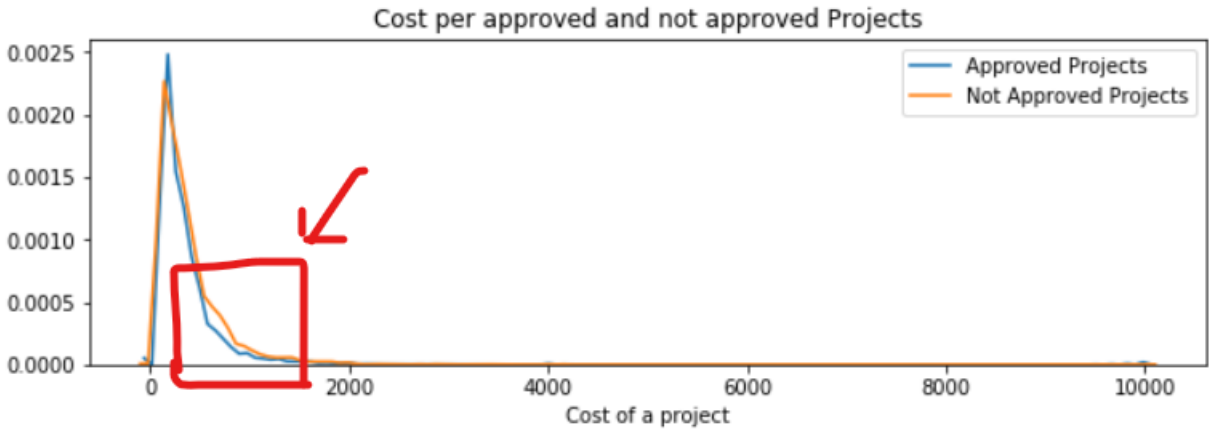
In [41]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

#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      |       0.66       |         1.97         |
|     5      |      13.59       |         41.9         |
|     10     |      33.88       |        73.67         |
|     15     |       58.0       |        99.109        |
|     20     |      77.38       |        118.56        |
|     25     |      99.95       |       140.892        |
|     30     |      116.68      |        162.23        |
```

```
|     35     |      137.232      |         184.014         |
|     40     |       157.0       |         208.632         |
|     45     |      178.265      |         235.106         |
|     50     |       198.99      |         263.145         |
|     55     |       223.99      |          292.61         |
|     60     |       255.63      |         325.144         |
|     65     |      285.412      |          362.39         |
|     70     |      321.225      |          399.99         |
|     75     |      366.075      |         449.945         |
|     80     |       411.67      |         519.282         |
|     85     |       479.0       |         618.276         |
|     90     |       593.11      |         739.356         |
|     95     |      801.598      |         992.486         |
|    100     |       9999.0      |          9999.0         |
+------------+-------------------+-------------------------+
```

Summary

From boxplot we can't derive any conclusions but from looking at PDF we can say that cost of non-approved projects is just higher then approved projects. The portion of PDF is indicated in figure also



But if you see percentile table, you will notice that at every stage (percentile) project which is not approved has more cost then project which is approved.

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [42]:
```
# How to Get Unique Values from a Column in Pandas Data Frame  https://cmdlinetips.com/2018/01/how-to-get-unique-values-from-a-column-in-pandas-data-frame/
# load the data with pd.read_csv
"""Try to get unique # of tows for each colums but gepminder count every rows individually """
gapminder = pd.read_csv('train_data.csv')
print(gapminder.info())
```

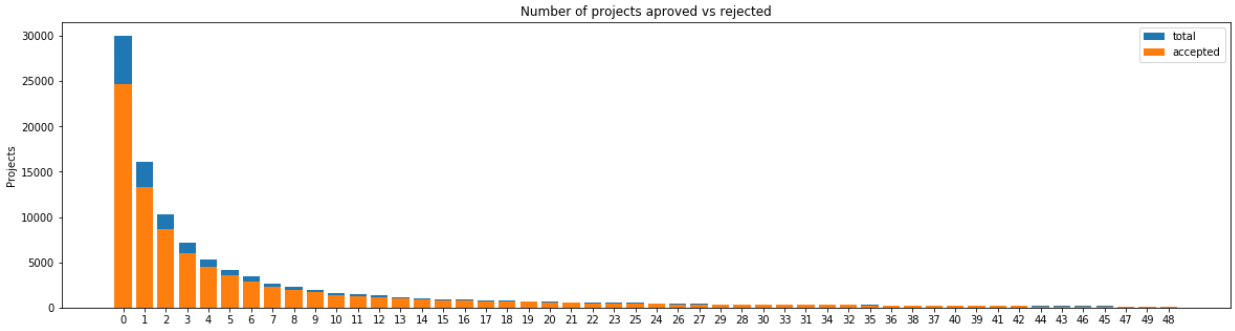
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109248 entries, 0 to 109247
Data columns (total 17 columns):
Unnamed: 0                                    109248 non-null int64
id                                            109248 non-null object
teacher_id                                    109248 non-null object
teacher_prefix                                109245 non-null object
school_state                                  109248 non-null object
project_submitted_datetime                    109248 non-null object
project_grade_category                        109248 non-null object
project_subject_categories                    109248 non-null object
project_subject_subcategories                 109248 non-null object
project_title                                 109248 non-null object
project_essay_1                               109248 non-null object

project_essay_2                               109248 non-null object
```

```
project_essay_3                                         3758 non-null object
project_essay_4                                         3758 non-null object
project_resource_summary                                109248 non-null object
teacher_number_of_previously_posted_projects    109248 non-null int64
project_is_approved                                     109248 non-null int64
dtypes: int64(3), object(14)
memory usage: 14.2+ MB
None
```

In [43]:
```
# To fine number of teachers
a= project_data.teacher_id
b=np.unique(a)
len(b)
```

Out[43]:
```
72168
```

In [44]:
```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_project
s', 'project_is_approved', top=50)
```




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

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved  total
\
46                                            46                 149    164
45                                            45                 141    153
47                                            47                 129    144
49                                            49                 128    143
48                                            48                 135    140

         Avg
46  0.908537
45  0.921569
47  0.895833
49  0.895105
48  0.964286
```

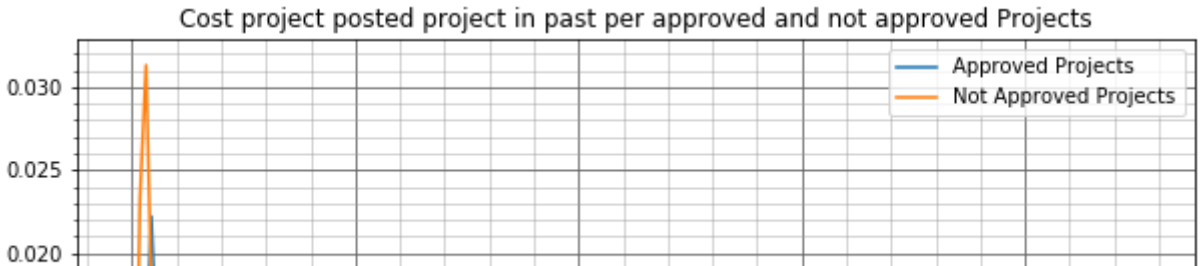
In [45]:
```
"""Code is copied from above. To visualize is there any relation with the proje
ct approval and
teacher_number_of_previously_posted_projects. """

temp = pd.DataFrame(project_data.groupby("teacher_number_of_previously_posted_p
```

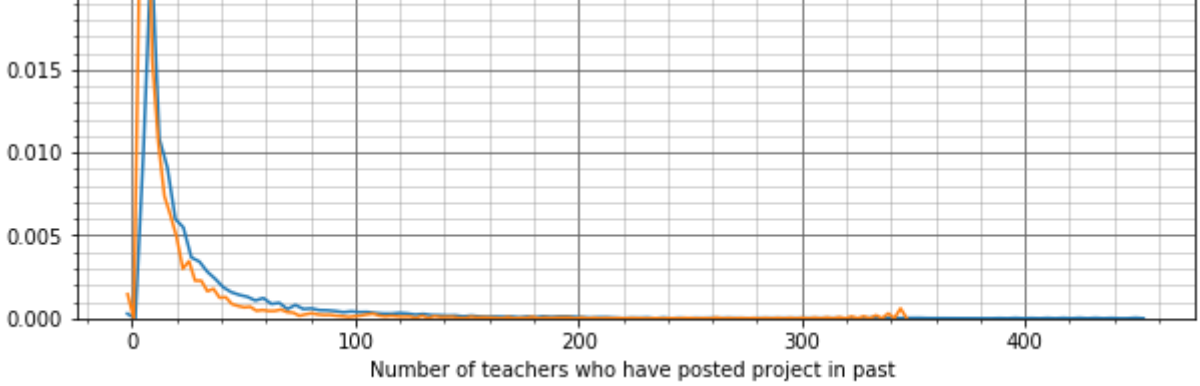
```
rojects")["project_is_approved"].apply(np.mean)).reset_index()
temp.columns = ['teacher_number_of_previously_posted_projects', 'num_proposals'
]
temp.sort_values(by=['num_proposals'], inplace=True)
print("teacher_number_of_previously_posted_projects lowest % approvals")
print(temp.head(5))
print('='*50)
print("teacher_number_of_previously_posted_projects with highest % approvals")
print(temp.tail(5))
```

```
teacher_number_of_previously_posted_projects lowest % approvals
     teacher_number_of_previously_posted_projects  num_proposals
315                                           322       0.000000
247                                           248       0.500000
331                                           343       0.500000
169                                           169       0.666667
225                                           225       0.666667
==================================================
teacher_number_of_previously_posted_projects with highest % approvals
     teacher_number_of_previously_posted_projects  num_proposals
242                                           242            1.0
243                                           243            1.0
244                                           244            1.0
246                                           246            1.0
373                                           451            1.0
```

In [46]:

```
# copy above code and modified according to my need
approved_t = project_data[project_data['project_is_approved']==1]['teacher_numb
er_of_previously_posted_projects'].values

rejected_t = project_data[project_data['project_is_approved']==0]['teacher_numb
er_of_previously_posted_projects'].values
"""sns.FacetGrid(project_data, hue="teacher_number_of_previously_posted_project
s", height=6)
sns.distplot(project_is_approved, hist=False, label="Approved Projects")"""
plt.figure(figsize=(10,5))
sns.distplot(approved_t, hist=False, label="Approved Projects")
sns.distplot(rejected_t, hist=False, label="Not Approved Projects")
plt.title('Cost project posted project in past per approved and not approved Pr
ojects')
plt.xlabel('Number of teachers who have posted project in past ')
plt.legend()
# plot with mejorand minor grid  https://riptutorial.com/matplotlib/example/140
63/plot-with-gridlines
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=.5)
plt.show()
```





In [48]:
```
# Try to row boxplot but it dos not give any infromations
"""plt.figure(figsize=(10,5))
plt.boxplot([approved_t, rejected_t])
plt.title('Cost project posted project in past per approved and not approved Pr
ojects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Number of teachers who have posted project in past')
plt.grid()
plt.show()"""
```

Out[48]:
```
"plt.figure(figsize=(10,5))\nplt.boxplot([approved_t, rejected_t])\nplt.title
('Cost project posted project in past per approved and not approved Project
s')\nplt.xticks([1,2],('Approved Projects','Rejected Projects'))\nplt.ylabel
('Number of teachers who have posted project in past')\nplt.grid()\nplt.show
()"
```

Summary of Univariate Analysis: teacher_number_of_previously_posted_projects

There are around 72,000 teachers who have applied for one or more projects. There are very large amount (40% of total) of teachers who haven't applied for the project before. You can't say anything about project approval and disapproval on the basis of number of projects posted by a teacher in past. From the PDF we can say that there is more possibility for teachers who haven't posted for project or posted for less than 10 projects in past rather than posted more than 100.

The PDF shows that if teacher posted form 20 to 80 projects before then it have more chances to get accepted then others.

From the PDF we can see that if teacher is applying for more project than there is more chances that project get rejected.

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

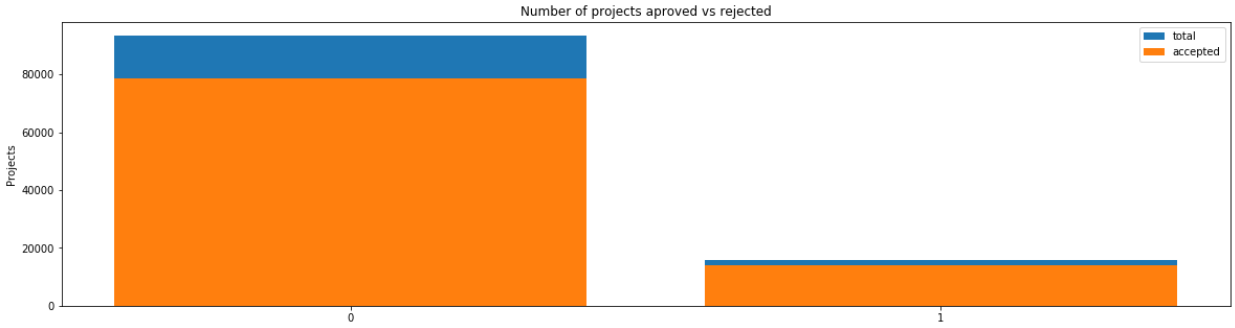
Check if the `presence of the numerical digits` in the `project_resource_summary` effects the acceptance of the project or not. If you observe that `presence of the numerical digits` is helpful in the classification, please include it for further process or you can ignore it.

In [47]:
```
prs = project_data.project_resource_summary
# https://stackoverflow.com/questions/19859282/check-if-a-string-contains-a-num
ber
# https://stackoverflow.com/questions/8234641/how-do-i-find-one-number-in-a-str
ing-in-python
# https://docs.python.org/2/library/stdtypes.html#str.isalpha
# https://github.com/RogerD044/DonorsChoose_Analysis/blob/master/2_DonorsChoose
```

```
_EDA_TSNE.ipynb
# https://github.com/akashsingh93/donors_choose/blob/master/2_DonorsChoose_EDA_TSNE.ipynb
"""new_prs = re.search(r'\d+',prs)"""
project_data["PRS_has_number"] = project_data["project_resource_summary"].apply(lambda row: re.search(r'\d',row)).map(bool).map(int)
"""We have created new column in dataset (PRS_has_number). If PRS contain number then data in PRS_has_number will be 1 otherwise 0."""
```

Out[47]:
```
'We have created new column in dataset (PRS_has_number). If PRS contain number then data in PRS_has_number will be 1 otherwise 0.'
```

In [48]:
```
# copy above code and modified according to my need
univariate_barplots(project_data, 'PRS_has_number', 'project_is_approved', False)
```




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

Summary of Univariate Analysis: project_resource_summary

From the above graph we can say that 90% of project report summary does not contain numerical values. Acceptance percentage of PRS who don't have numerical value is 84. And if teacher numeric values in PRS then acceptance percentage will be increased by 5%.

1.3 Text preprocessing

1.3.1 Essay Text

In [49]:
```
project_data.head(2)
```

Out[49]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_su
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

2 rows × 21 columns

In [50]:
```
# 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
or third languages. We are a melting pot of refugees, immigrants, and native-
born Americans bringing the gift of language to our school. \r\n\r\n We have
over 24 languages represented in our English Learner program with students at
every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and ex
periences to us that open our eyes to new cultures, beliefs, and respect.\"Th
e limits of your language are the limits of your world.\"-Ludwig Wittgenstein
Our English learner's have a strong support system at home that begs for more
resources.  Many times our parents are learning to read and speak English alo
ng side of their children.  Sometimes this creates barriers for parents to be
able to help their child learn phonetics, letter recognition, and other readi
ng skills.\r\n\r\nBy providing these dvd's and players, students are able to
continue their mastery of the English language even if no one at home is able
to assist.  All families with students within the Level 1 proficiency status,
will be a offered to be a part of this program.  These educational videos wil
l be specially chosen by the English Learner Teacher and will be sent home re
gularly to watch.  The videos are to help the child develop early reading ski
lls.\r\n\r\nParents that do not have access to a dvd player will have the opp
ortunity to check out a dvd player to use for the year.  The plan is to use t
hese videos and educational dvd's for the years to come for other EL student
s.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year al
l love learning, at least most of the time. At our school, 97.3% of the stude
nts receive free or reduced price lunch. Of the 560 students, 97.3% are minor
ity students. \r\nThe school has a vibrant community that loves to get togeth
er and celebrate. Around Halloween there is a whole school parade to show off
the beautiful costumes that students wear. On Cinco de Mayo we put on a big f
estival with crafts made by the students, dances, and games. At the end of th
e year the school hosts a carnival to celebrate the hard work put in during t
he school year, with a dunk tank being the most popular activity.My students
will use these five brightly colored Hokki stools in place of regular, statio
nary, 4-legged chairs. As I will only have a total of ten in the classroom an
d 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 speci
al chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of th
e day they will be used by the students who need the highest amount of moveme
nt in their life in order to stay focused on school.\r\n\r\nWhenever asked wh
at the classroom is missing, my students always say more Hokki Stools. They c
an't get their fill of the 5 stools we already have. When the students are si
tting in group with me on the Hokki Stools, they are always moving, but at th
e same time doing their work. Anytime the students get to pick where they can
sit, the Hokki Stools are the first to be taken. There are always students wh
o head over to the kidney table to get one of the stools who are disappointed
as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for
```

7 hours a day. The Hokki stools will be a compromise that allow my students t
o 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 co
re muscles for balance while they sit. For many of my students, these chairs
will take away the barrier that exists in schools for a child who can't sit s
till.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 theme
d room for my students look forward to coming to each day.\r\n\r\nMy class is
made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r
\nThey attend a Title I school, which means there is a high enough percentage
of free and reduced-price lunch to qualify. Our school is an \"open classroom
\" concept, which is very unique as there are no walls separating the classro
oms. These 9 and 10 year-old students are very eager learners; they are like
sponges, absorbing all the information and experiences and keep on wanting mo
re.With these resources such as the comfy red throw pillows and the whimsical
nautical hanging decor and the blue fish nets, I will be able to help create
the mood in our classroom setting to be one of a themed nautical environment.
Creating a classroom environment is very important in the success in each and
every child's education. The nautical photo props will be used with each chil
d as they step foot into our classroom for the first time on Meet the Teacher
evening. I'll take pictures of each child with them, have them developed, and
then hung in our classroom ready for their first day of 4th grade. This kind
gesture will set the tone before even the first day of school! The nautical t
hank you cards will be used throughout the year by the students as they creat
e thank you cards to their team groups.\r\n\r\nYour generous donations will h
elp me to help make our classroom a fun, inviting, learning environment from
day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to g
et our classroom ready. Please consider helping with this project to make our
new school year a very successful one. Thank you!nannan
==========
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations. \r\n\r\nThe materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or red
uced price lunch. Despite their disabilities and limitations, my students lo
ve coming to school and come eager to learn and explore.Have you ever felt li
ke you had ants in your pants and you needed to groove and move as you were i
n a meeting? This is how my kids feel all the time. The want to be able to mo
ve as they learn or so they say.Wobble chairs are the answer and I love then
because they develop their core, which enhances gross motor and in Turn fine
motor skills. \r\nThey also want to learn through games, my kids don't want t
o sit and do worksheets. They want to learn to count by jumping and playing.
Physical engagement is the key to our success. The number toss and color and
shape mats can make that happen. My students will forget they are doing work
and just have the fun a 6 year old deserves.nannan
==========
The mediocre teacher tells. The good teacher explains. The superior teacher d
emonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school ha
s 803 students which is makeup is 97.6% African-American, making up the large
st segment of the student body. A typical school in Dallas is made up of 23.
2% African-American students. Most of the students are on free or reduced lun
ch. We aren't receiving doctors, lawyers, or engineers children from rich bac
kgrounds or neighborhoods. As an educator I am inspiring minds of young child
ren and we focus not only on academics but one smart, effective, efficient, a
nd disciplined students with good character.In our classroom we can utilize t
he Bluetooth for swift transitions during class. I use a speaker which does
n't amplify the sound enough to receive the message. Due to the volume of my
speaker my students can't hear videos or books clearly and it isn't making th
e lessons as meaningful. But with the bluetooth speaker my students will be a
ble to hear and I can stop, pause and replay it at any time.\r\nThe cart will
allow me to have more room for storage of things that are needed for the day
and has an extra part to it I can use. The table top chart has all of the le

```
tter, words and pictures for students to learn about different letters and it
is more accessible.nannan
==================================================
```

In [51]:
```
# 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 [52]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations. \r\n\r\nThe materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or red
uced price lunch.  Despite their disabilities and limitations, my students lo
ve coming to school and come eager to learn and explore.Have you ever felt li
ke you had ants in your pants and you needed to groove and move as you were i
n a meeting? This is how my kids feel all the time. The want to be able to mo
ve as they learn or so they say.Wobble chairs are the answer and I love then
because they develop their core, which enhances gross motor and in Turn fine
motor skills. \r\nThey also want to learn through games, my kids do not want
to sit and do worksheets. They want to learn to count by jumping and playing.
Physical engagement is the key to our success. The number toss and color and
shape mats can make that happen. My students will forget they are doing work
and just have the fun a 6 year old deserves.nannan
==================================================
```

In [53]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-b
reaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations.      The materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced
price lunch.  Despite their disabilities and limitations, my students love co
ming to school and come eager to learn and explore.Have you ever felt like yo
u had ants in your pants and you needed to groove and move as you were in a m
eeting? This is how my kids feel all the time. The want to be able to move as
they learn or so they say.Wobble chairs are the answer and I love then becaus
e they develop their core, which enhances gross motor and in Turn fine motor
skills.   They also want to learn through games, my kids do not want to sit a
```

```
nd do worksheets. They want to learn to count by jumping and playing. Physica
l engagement is the key to our success. The number toss and color and shape m
ats can make that happen. My students will forget they are doing work and jus
t have the fun a 6 year old deserves.nannan
```

In [54]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays cognitive delays gross fine motor delays to autism They are eage
r beavers and always strive to work their hardest working past their limitati
ons The materials we have are the ones I seek out for my students I teach in
a Title I school where most of the students receive free or reduced price lun
ch Despite their disabilities and limitations my students love coming to scho
ol and come eager to learn and explore Have you ever felt like you had ants i
n your pants and you needed to groove and move as you were in a meeting This
is how my kids feel all the time The want to be able to move as they learn or
so they say Wobble chairs are the answer and I love then because they develop
their core which enhances gross motor and in Turn fine motor skills They also
want to learn through games my kids do not want to sit and do worksheets They
want to learn to count by jumping and playing Physical engagement is the key
to our success The number toss and color and shape mats can make that happen
My students will forget they are doing work and just have the fun a 6 year ol
d deserves nannan
```

In [55]:
```
# 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', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d'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", 'm
a', '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 [56]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
```

```
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████
| 109248/109248 [02:40<00:00, 682.68it/s]
```

In [57]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[57]:
```
'my kindergarten students varied disabilities ranging speech language delays
cognitive delays gross fine motor delays autism they eager beavers always str
ive work hardest working past limitations the materials ones i seek students
i teach title i school students receive free reduced price lunch despite disa
bilities limitations students love coming school come eager learn explore hav
e ever felt like ants pants needed groove move meeting this kids feel time th
e want able move learn say wobble chairs answer i love develop core enhances
gross motor turn fine motor skills they also want learn games kids not want s
it worksheets they want learn count jumping playing physical engagement key s
uccess the number toss color shape mats make happen my students forget work f
un 6 year old deserves nannan'
```

1.3.2 Project title Text

In [58]:
```
# similarly you can preprocess the titles also
# printing some random essays.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [59]:
```
PT = decontracted(project_data['project_title'].values[20000])
print(PT)
print("="*50)
```

```
We Need To Move It While We Input It!
==================================================
```

In [60]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-b
reaks-python/
PT = PT.replace('\\r', ' ')
PT = PT.replace('\\"', ' ')
```

```
PT = PT.replace('\\n', ' ')
print(PT)
```

```
We Need To Move It While We Input It!
```

In [61]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
PT = re.sub('[^A-Za-z0-9]+', ' ', PT)
print(PT)
```

```
We Need To Move It While We Input It
```

In [62]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    PT = decontracted(sentance)
    PT = PT.replace('\\r', ' ')
    PT = PT.replace('\\"', ' ')
    PT = PT.replace('\\n', ' ')
    PT = re.sub('[^A-Za-z0-9]+', ' ', PT)
    # https://gist.github.com/sebleier/554280
    PT = ' '.join(e for e in PT.split() if e not in stopwords)
    preprocessed_project_title.append(PT.lower().strip())
```

```
100%|██████████| 109248/109248 [00:07<00:00, 14214.78it/s]
```

In [63]:
```
# after preprocesing
preprocessed_project_title[20000]
```

Out[63]:
```
'we need to move it while we input it'
```

In [64]:
```
import re

"""def contracted(phrase):
    # specific
    phrase = re.sub(r".", "", phrase)

    return phrase"""
from tqdm import tqdm
preprocessed_teacher_prefix = []
for sentance in tqdm(project_data['teacher_prefix'].values):
    PT = decontracted(str(sentance))
    PT = PT.replace('\\r', ' ')
    PT = PT.replace('\\"', ' ')
    PT = PT.replace('\\n', ' ')
    PT = re.sub('[^A-Za-z0-9]+', ' ', PT)
        # https://gist.github.com/sebleier/554280
    PT = ' '.join(e for e in PT.split() if e not in stopwords)
    preprocessed_teacher_prefix.append(PT.lower().strip())


"""Try to replace teacher_prefix with preporcessed_teacher_prefix"""
```

```
100%|██████████| 109248/109248 [00:05<00:00, 21821.37it/s]
```

Out[64]:
```
'Try to replace teacher_prefix with preporcessed_teacher_prefix'
```

In [65]:
```
preprocessed_teacher_prefix[3]
```

Out[65]:
```
'mrs'
```

1. 4 Preparing data for models

In [66]:
```
project_data.columns
```

Out[66]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       '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', 'quantit
y',
       'PRS_has_number'],
      dtype='object')
```

we are going to consider

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

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

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

1.4.1 Vectorizing Categorical data

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

In [67]:
```
# 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'].valu
es)
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

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

In [68]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lower
case=False, binary=True)
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
```

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

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

In [69]:
```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category also
```

In [70]:
```
# for state
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())

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


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


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

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

In [71]:
```
# for teacher_prefix https://github.com/RogerD044/DonorsChoose_Analysis/blob/master/2_DonorsChoose_EDA_TSNE.ipynb
# https://stackoverflow.com/questions/46543996/blast-parsing-attributeerror-float-object-has-no-attribute-split
# np.nan is an invalid document, expected byte or unicode string.
# https://blog.jerrysha.com/2016/11/machine-learning-introduction.html
# https://github.com/mprem1204/Supervised-Learning-models-on-Donors-choose-dataset/blob/master/2_DonorsChoose_EDA_TSNE.ipynb
# https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is-an-invalid-document
"""my_counter = Counter()"""
t_pre = project_data['teacher_prefix'].unique()
"""for word in t_pre.values:
    my_counter.update(word.split())"""

"""teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv: kv[1]))"""

vectorizer = CountVectorizer(vocabulary = t_pre, lowercase=False, binary=True)
```

```
vectorizer.fit(project_data['teacher_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())

teacher_prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].va
lues.astype('U'))
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
```

```
['Mrs.', 'Mr.', 'Ms.', 'Teacher', nan, 'Dr.']
Shape of matrix after one hot encodig  (109248, 6)
```

In [72]:
```
# for project_grade_category

my_counter = Counter()
# how to remove word in string python : -> https://www.tutorialspoint.com/How-t
o-remove-specific-characters-from-a-string-in-Python
# https://stackoverflow.com/questions/25900332/find-last-word-in-a-string-withi
n-a-list-pandas-python-3
no_grade = project_data['project_grade_category'].str.split().str[-1]
for word in no_grade.values:
    my_counter.update(word.split())

grade_category_dict = dict(my_counter)
sorted_grade_category_dict = dict(sorted(grade_category_dict.items(), key=lambd
a kv: kv[1]))

vectorizer = CountVectorizer(vocabulary=list(sorted_grade_category_dict.keys
()), lowercase=False, binary=True)
vectorizer.fit(no_grade.values)
print(vectorizer.get_feature_names())

project_grade_category_one_hot = vectorizer.transform(no_grade.values)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot.s
hape)
```

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

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [73]:
```
# 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 encodig ",text_bow.shape)
```

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

1.4.2.2 Bag of Words on `project_title`

In [74]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_project_title)
print("Shape of matrix after one hot encodig ",title_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3329)
```

1.4.2.3 TFIDF vectorizer

In [75]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

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

1.4.2.4 TFIDF Vectorizer on `project_title`

In [76]:
```
# Similarly you can vectorize for title also

vectorizer = TfidfVectorizer(min_df=10)
title_tfidf = vectorizer.fit_transform(preprocessed_project_title)
print("Shape of matrix after one hot encodig ",title_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3329)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [77]:
```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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

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

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

words_courpus = {}
words_glove = set(model.keys())
for i in words:
```

```
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


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

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


'''
```

Out[77]:
```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ==========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words that are present in both glove vectors and our coupus",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n\n# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n\'
```

In [78]:
```
# 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 [79]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_essay = []; # 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_essay.append(vector)

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

```
100%|██████████| 109248/109248 [02:00<00:00, 906.26it/s]
```

```
109248
```

```
109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [80]:
```
# Similarly you can vectorize for title also

avg_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(preprocessed_project_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title.append(vector)

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

```
100%|██████████████████████████████████████████████████████████████████████████|
109248/109248 [00:05<00:00, 20142.41it/s]

109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [81]:
```
# 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 [82]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_essay = []; # the avg-w2v for each sentence/review is stored
 in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf 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_essay.append(vector)

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

```
100%|██████████████████████████████████████████████████████████████████████████
| 109248/109248 [12:03<00:00, 151.06it/s]
```

```
109248
300
```

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

In [83]:
```
# Similarly you can vectorize for title also

tfidf_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored
 in this list
for sentence in tqdm(preprocessed_project_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf 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_title.append(vector)

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

```
100%|██████████████████████████████████████████████████████████████████████████|
109248/109248 [00:08<00:00, 13547.92it/s]

109248
300
```

1.4.3 Vectorizing Numerical features

In [84]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sk
learn.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 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

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

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

In [85]:
```
price_standardized
```

Out[85]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

In [86]:
```
# for quantity

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

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

```
C:\Users\prasa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

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

```
C:\Users\prasa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

Out[86]:
```
array([[ 0.23047132],
       [-0.60977424],
       [ 0.19227834],
       ...,
       [-0.4951953 ],
       [-0.03687954],
       [-0.45700232]])
```

In [87]:
```
# for teacher_number_of_previously_posted_projects

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

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized = tmppp_scalar.transf
orm(project_data['teacher_number_of_previously_posted_projects'].values.reshape
(-1, 1))
teacher_number_of_previously_posted_projects_standardized
```

```
C:\Users\prasa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

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

```
C:\Users\prasa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: D
```

```
ataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

Out[87]:
```
array([[-0.40152481],
       [-0.14951799],
       [-0.36552384],
       ...,
       [-0.29352189],
       [-0.40152481],
       [-0.40152481]])
```

1.4.4 Merging all the above features

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

In [88]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(school_state_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(project_grade_category_one_hot.shape)
print(text_bow.shape)
print(title_bow.shape)
print(price_standardized.shape)
print(quantity_standardized.shape)
print(teacher_number_of_previously_posted_projects_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 51)
(109248, 6)
(109248, 4)
(109248, 16623)
(109248, 3329)
(109248, 1)
(109248, 1)
(109248, 1)
```

In [89]:
```
# 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, teacher_prefix_one_hot, sub_categories_one_hot,
school_state_one_hot, project_grade_category_one_hot,  text_bow, title_bow, pri
ce_standardized, quantity_standardized, teacher_number_of_previously_posted_pro
jects_standardized))
X.shape
```

Out[89]:
```
(109248, 20055)
```

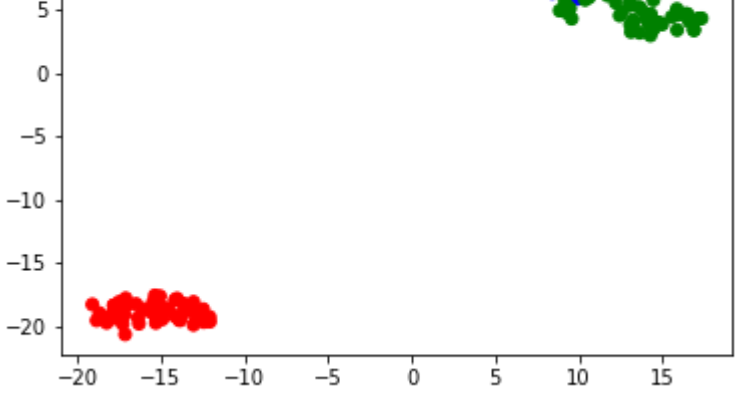
Assignment 2: Apply TSNE

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.

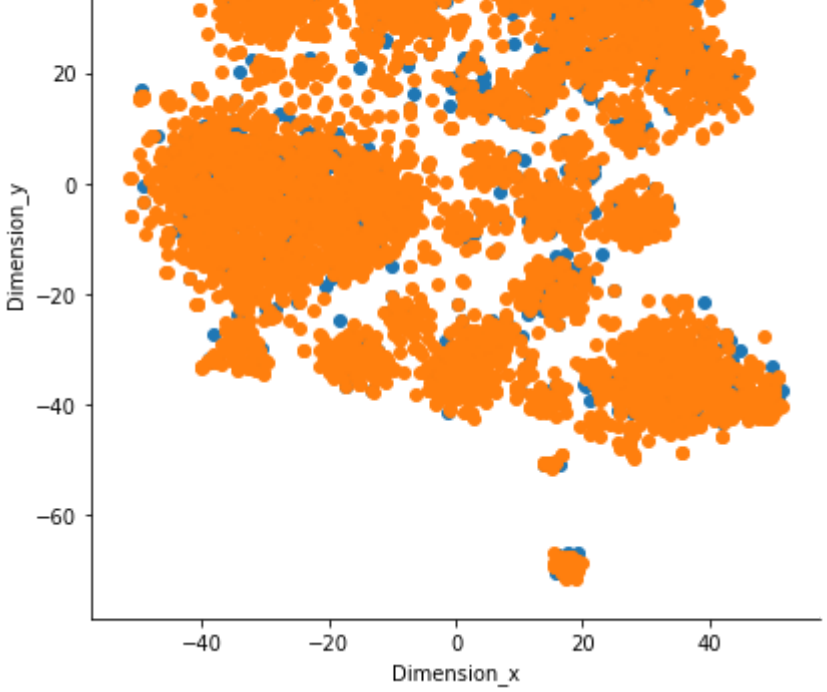
2. EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects
3. Build the data matrix using these features
   - school_state : categorical data (one hot encoding)
   - clean_categories : categorical data (one hot encoding)
   - clean_subcategories : categorical data (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 [90]:
```
# 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 [89]:
```
# please write all of the code with proper documentation and proper titles for
 each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the read
er
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
# https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
# tsne python code:-> https://scipy-lectures.org/packages/scikit-learn/auto_exa
mples/plot_tsne.html
# code copy for above
X1 = hstack(( categories_one_hot, sub_categories_one_hot, school_state_one_hot,
price_standardized, quantity_standardized, title_bow))
X1.shape
# csr metrix :-> https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/sc
ipy.sparse.csr_matrix.html
"""x = X1.data[:5000]
y = X1.target[:5000]"""
csr_mat = X1.tocsr()
"""dense_mat = csr_mat.todense()"""
#i m taking only first 5000 data as my machine is not powesful enought
csr_mat_1000 = csr_mat[0:5000,:].toarray()
# for labels
leb_1000= project_data['project_is_approved'].values[0:5000]

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

X_embedding = tsne.fit_transform(csr_mat_1000)
# 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.vstack((X_embedding.T, leb_1000)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'label'])
"""colors = {0:'red', 1:'blue', 2:'green'}"""
# https://github.com/RogerD044/DonorsChoose_Analysis/blob/master/2_DonorsChoose
_EDA_TSNE.ipynb
sns.FacetGrid(for_tsne_df, hue="label", height=6).map(plt.scatter, 'Dimension_
x', 'Dimension_y')
"""plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_ts
ne_df['label'].apply(lambda x: colors[x]))"""
plt.legend()
plt.show()
```

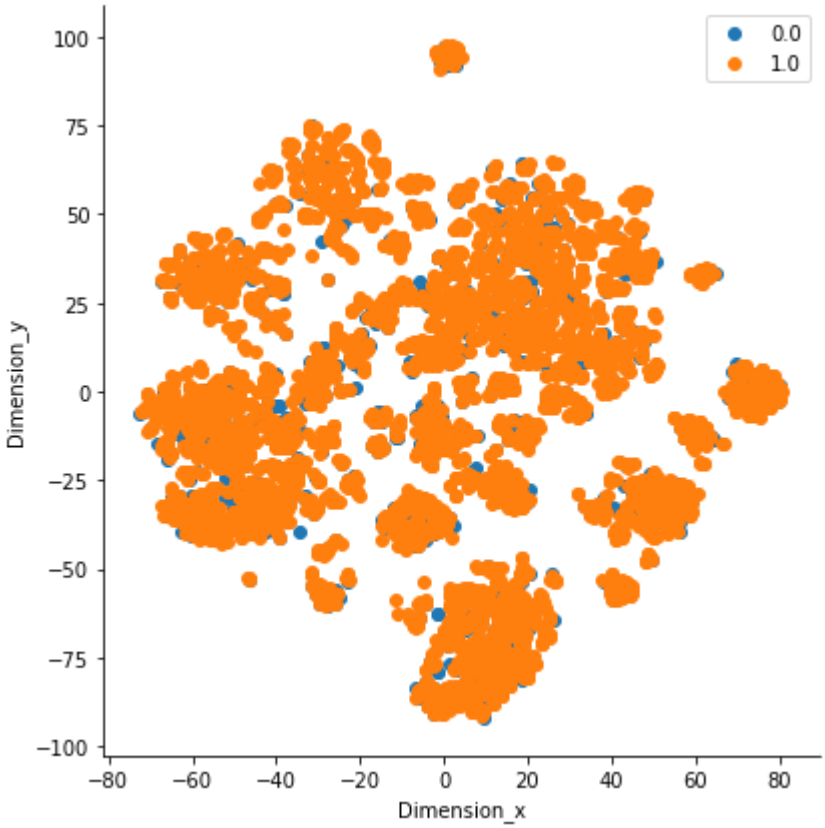


2.2 TSNE with `TFIDF` encoding of `project_title` feature

In [90]:

```
# please write all the code with proper documentation, and proper titles for ea
ch 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
# https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
# tsne python code:-> https://scipy-lectures.org/packages/scikit-learn/auto_exa
mples/plot_tsne.html
# code copy for above
X1 = hstack(( categories_one_hot, sub_categories_one_hot, school_state_one_hot,
price_standardized, quantity_standardized, title_tfidf))
X1.shape
# csr metrix :-> https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/sc
ipy.sparse.csr_matrix.html
"""x = X1.data[:5000]
y = X1.target[:5000]"""
csr_mat = X1.tocsr()
"""dense_mat = csr_mat.todense()"""
#i m taking only first 5000 data as my machine is not powesful enought
csr_mat_1000 = csr_mat[0:5000,:].toarray()
# for labels
leb_1000= project_data['project_is_approved'].values[0:5000]

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

X_embedding = tsne.fit_transform(csr_mat_1000)
# 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.vstack((X_embedding.T, leb_1000)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'label'])
"""colors = {0:'red', 1:'blue', 2:'green'}"""
# https://github.com/RogerD044/DonorsChoose_Analysis/blob/master/2_DonorsChoose
_EDA_TSNE.ipynb
sns.FacetGrid(for_tsne_df, hue="label", height=6).map(plt.scatter, 'Dimension_
x', 'Dimension_y')
```

```
"""plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_ts
ne_df['label'].apply(lambda x: colors[x]))"""
plt.legend()
plt.show()
```

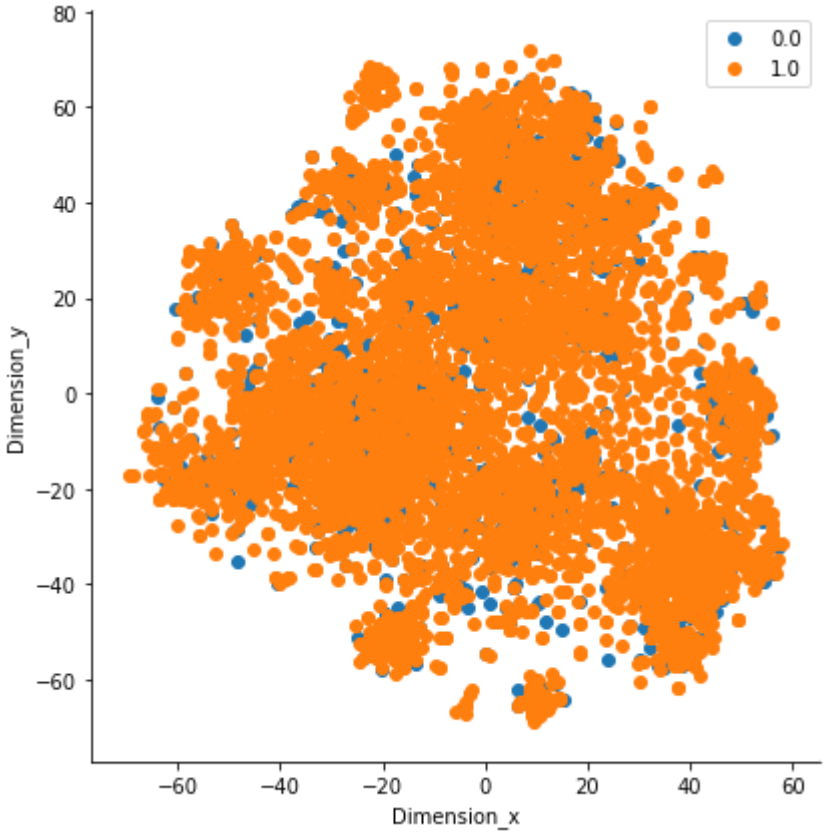


2.3 TSNE with `AVG W2V` encoding of `project_title` feature

In [92]:

```
# please write all the code with proper documentation, and proper titles for ea
ch 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
 # https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
# tsne python code:-> https://scipy-lectures.org/packages/scikit-learn/auto_exa
mples/plot_tsne.html
# code copy for above
X1 = hstack(( categories_one_hot, sub_categories_one_hot, school_state_one_hot,
price_standardized, quantity_standardized, avg_w2v_vectors_title))
X1.shape
# csr metrix :-> https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/sc
ipy.sparse.csr_matrix.html
"""x = X1.data[:5000]
y = X1.target[:5000]"""
csr_mat = X1.tocsr()
"""dense_mat = csr_mat.todense()"""
#i m taking only first 5000 data as my machine is not powesful enought
csr_mat_1000 = csr_mat[0:5000,:].toarray()
# for labels
leb_1000= project_data['project_is_approved'].values[0:5000]

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

X_embedding = tsne.fit_transform(csr_mat_1000)
# 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
```

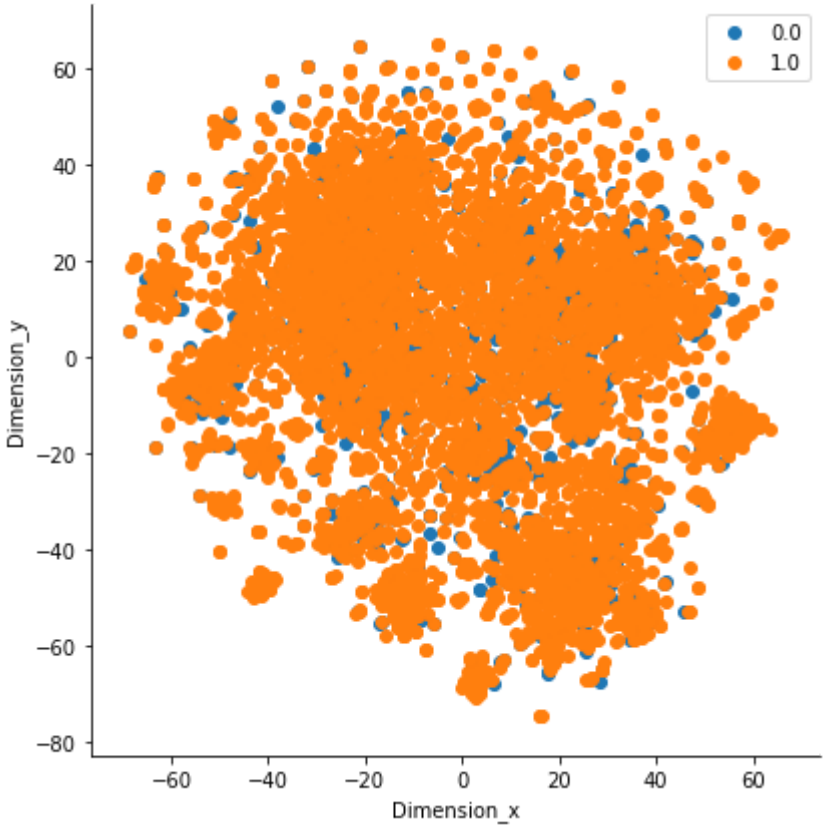
```
for_tsne = np.vstack((X_embedding.T, leb_1000)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'label'])
"""colors = {0:'red', 1:'blue', 2:'green'}"""
# https://github.com/RogerD044/DonorsChoose_Analysis/blob/master/2_DonorsChoose
_EDA_TSNE.ipynb
sns.FacetGrid(for_tsne_df, hue="label", height=6).map(plt.scatter, 'Dimension_
x', 'Dimension_y')
"""plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_ts
ne_df['label'].apply(lambda x: colors[x]))"""
plt.legend()
plt.show()
```



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

In [92]:
```
# please write all the code with proper documentation, and proper titles for ea
ch 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 tfidf_w2v_vectors
 # https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
# tsne python code:-> https://scipy-lectures.org/packages/scikit-learn/auto_exa
mples/plot_tsne.html
# code copy for above
X1 = hstack(( categories_one_hot, sub_categories_one_hot, school_state_one_hot,
price_standardized, quantity_standardized,tfidf_w2v_vectors_title))
X1.shape
# csr metrix :-> https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/sc
ipy.sparse.csr_matrix.html
"""x = X1.data[:5000]
y = X1.target[:5000]"""
csr_mat = X1.tocsr()
"""dense_mat = csr_mat.tdense()"""
```

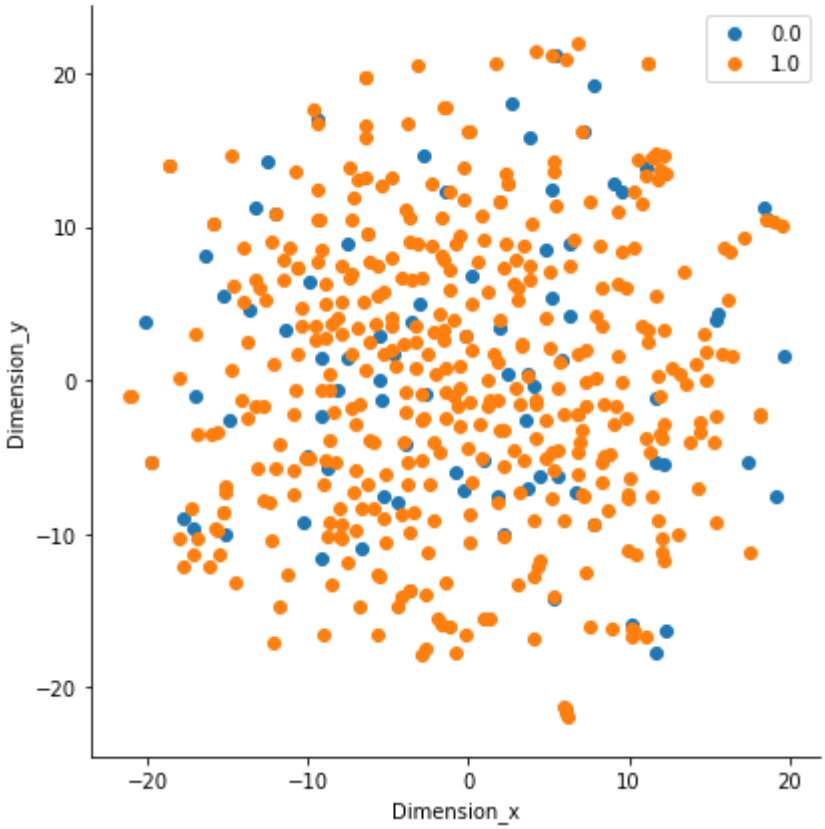
```
#i m taking only first 5000 data as my machine is not powesful enought
csr_mat_1000 = csr_mat[0:5000,:].toarray()
# for labels
leb_1000= project_data['project_is_approved'].values[0:5000]

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

X_embedding = tsne.fit_transform(csr_mat_1000)
# 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.vstack((X_embedding.T, leb_1000)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'label'])
"""colors = {0:'red', 1:'blue', 2:'green'}"""
# https://github.com/RogerD044/DonorsChoose_Analysis/blob/master/2_DonorsChoose
_EDA_TSNE.ipynb
sns.FacetGrid(for_tsne_df, hue="label", height=6).map(plt.scatter, 'Dimension_
x', 'Dimension_y')
"""plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_ts
ne_df['label'].apply(lambda x: colors[x]))"""
plt.legend()
plt.show()
```



In [91]:

```
# combining all the parameters


X1 = hstack(( categories_one_hot, sub_categories_one_hot, school_state_one_hot,
teacher_prefix_one_hot, \
              project_grade_category_one_hot, price_standardized, quantity_stand
ardized,\
              teacher_number_of_previously_posted_projects_standardized, avg_w2v
_vectors_title, \
              tfidf_w2v_vectors_essay, tfidf_w2v_vectors_title, avg_w2v_vectors_
essay))
X1.shape
# csr metrix :-> https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/sc
ipy.sparse.csr_matrix.html
"""x = X1.data[:5000]
```

```
y = X1.target[:5000]"""
csr_mat = X1.tocsr()
"""dense_mat = csr_mat.todense()"""
#i m taking only first 500 data as my machine is not powesful enought,
csr_mat_1000 = csr_mat[0:500,:].toarray()
# for labels
leb_1000= project_data['project_is_approved'].values[0:500]

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

X_embedding = tsne.fit_transform(csr_mat_1000)
# 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.vstack((X_embedding.T, leb_1000)).T
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y',
'label'])
"""colors = {0:'red', 1:'blue', 2:'green'}"""
# https://github.com/RogerD044/DonorsChoose_Analysis/blob/master/2_DonorsChoose
_EDA_TSNE.ipynb
sns.FacetGrid(for_tsne_df, hue="label", height=6).map(plt.scatter, 'Dimension_
x', 'Dimension_y')
"""plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_ts
ne_df['label'].apply(lambda x: colors[x]))"""
plt.legend()
plt.show()
```



2.5 Summary

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

1. From the above graph we can say that all the features are not well separated in hyper plan so T-SNE is not able to separate it in to 2D as well.
2. If we combine all the features then also T-SNE fails to separate.
3. It may possible that if we take more data (i have take only 1000 points) then result may improve.