In [32]:

```
%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


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
```

Reading the data

In [38]:

```
dft = pd.read_csv('train_data.csv')
dfr = pd.read_csv('resources.csv')
```

In [39]:

```
print("Number of data points in train data", dft.shape)
print('-'*50)
print("The attributes of data :",dft.columns.values)
dft.head(5)
```

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

Out[39]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proj
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

In [40]:

```
print("Number of data points in train data",dfr.shape)
print(dfr.columns.values)
dfr.head(5)
```

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

Out[40]:

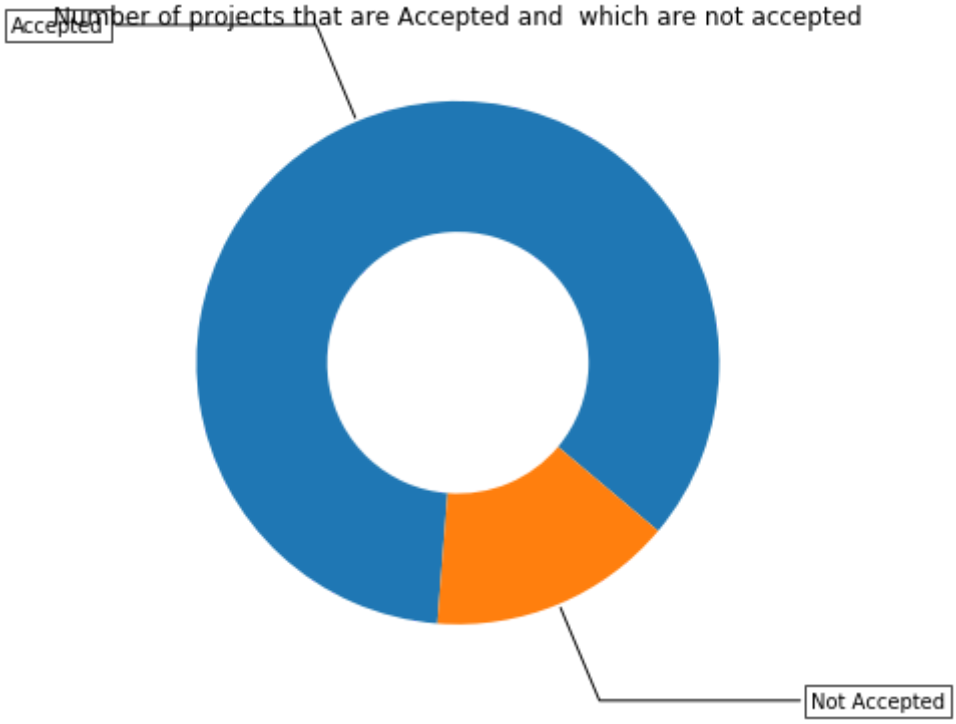
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

Data Analysis

In [41]:

```
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-gl
r-gallery-pie-and-polar-charts-pie-and-donut-labels-py


y_value_counts = dft['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("Number of projects that are Accepted and  which are not accepted")

plt.show()
```

```
Number of projects thar are approved for funding  92706 , ( 84.85830404217
927 %)
Number of projects thar are not approved for funding  16542 , ( 15.1416959
57820739 %)
```



Univariate Analysis: School State

In [42]:

```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039

temp = pd.DataFrame(dft.groupby("school_state")["project_is_approved"].apply(np.mean)).
reset_index()
# if you have data which contain only 0 and 1, then the mean = percentage (think about
 it)
temp.columns = ['state_code', 'num_proposals']

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

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,14
3)']]

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[42]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9
620\n\nscl = [[0.0, \'rgb(242,240,247)\'],[0.2, \'rgb(218,218,235)\'],[0.
4, \'rgb(188,189,220)\'],            [0.6, \'rgb(158,154,200)\'],[0.8, \'r
gb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\ndata = [ dict(\n        t
ype=\'choropleth\',\n        colorscale = scl,\n        autocolorscale = F
alse,\n        locations = temp[\'state_code\'],\n        z = temp[\'num_p
roposals\'].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 p
ro")\n    ) ]\n\nlayout = dict(\n        title = \'Project Proposals % of
Acceptance Rate by US States\',\n        geo = dict(\n            scope=
\'usa\',\n            projection=dict( type=\'albers usa\' ),\n
showlakes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n
),\n    )\n\nfig = go.Figure(data=data, layout=layout)\noffline.iplot(fig,
filename=\'us-map-heat-map\')\n'
```

In [43]:

```
# 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 :

Delaware (DE) state from the United States has the highest percent of projects accepted within the whole country having almost 90% acceptance rate, followed by North Dakota (ND) and Washington (WA) nearly 89% and 88% respectively each. Vermont (VT) has the lowest Approval rate with exactly 80% followed by District of Columbia (DC) and Texas (TX) with nearly 80% and 81% respectively.

In [44]:

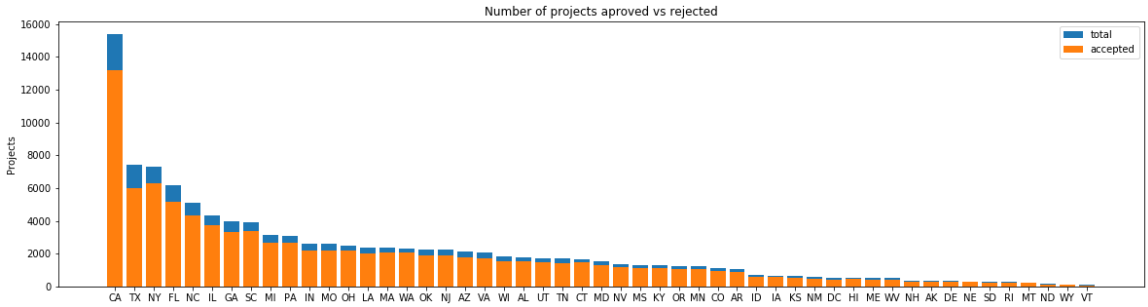
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

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

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

In [45]:

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

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

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

    if top:
        temp = temp[0:top]

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

In [46]:

```
univariate_barplots(dft, '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

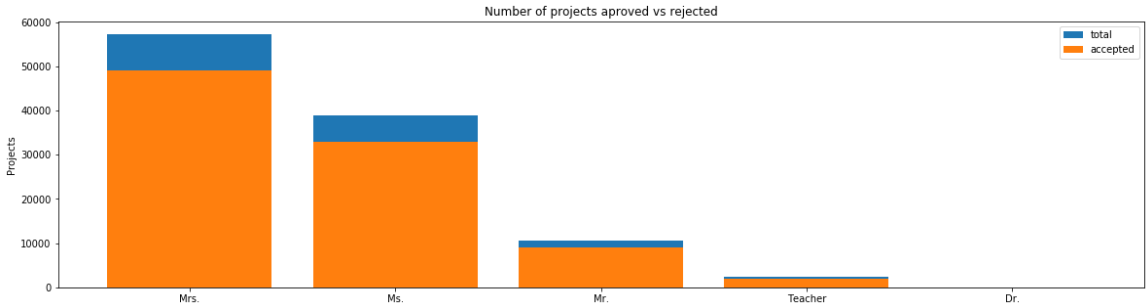
SUMMARY :

1. Every state has greater than 80% success rate in approval.
2. There is a lot of variability in the number of projects that have been submitted across the States.
3. California(CA) has the highest number of project proposals when compared to the other states, Surprisingly, 85% of the projects gets approved on an average which is nearly 13205 out of 15388 project proposals.
4. Vermont(VT) has the lowest number of project proposals initiated (80) and almost 80% of the project proposal gets acceptance (64 out of 80). Well, in terms of rejection only 16 were rejected.

Univariate Analysis: teacher_prefix

In [47]:

```
univariate_barplots(dft, '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
```

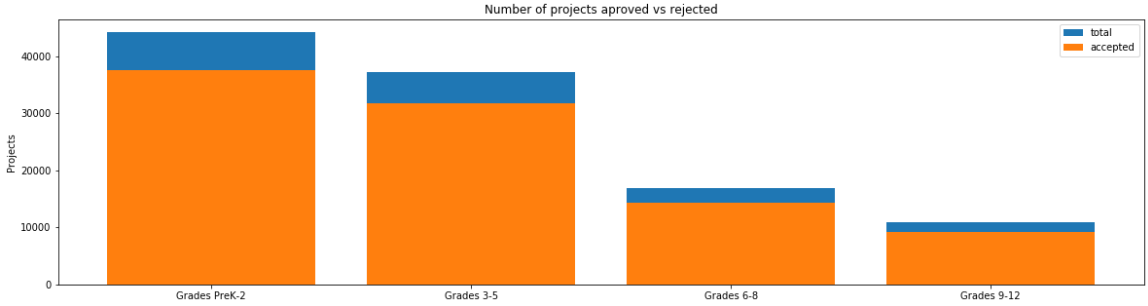
SUMMARY :

1. Female Teachers have the maximum number of projects proposed and accepted compared to the male teachers.
2. Teachers with prefixes Mrs. , which means Married Women as teachers have a higher number of projects Proposed as well as accepted when compared to the younger Unmarried Women Teachers.
3. Teachers with Dr. title have proposed hardly 13 projects and out of which 9 of them have been approved.

Univariate Analysis: project_grade_category

In [48]:

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




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

SUMMARY :

1. There are alot of projects proposed for the students between Pre Kindergarden and 2nd Grade while for the rest it keeps decreasing.
2. The average Acceptance rate of the project is 84% irrespective of the Grade.
3. We also notice that Students between the 9th Grade and 12th Grade have the lowest number of projects proposed as well as accepted.

Univariate Analysis: project_subject_categories

In [49]:

```
catogories = list(dft['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [50]:

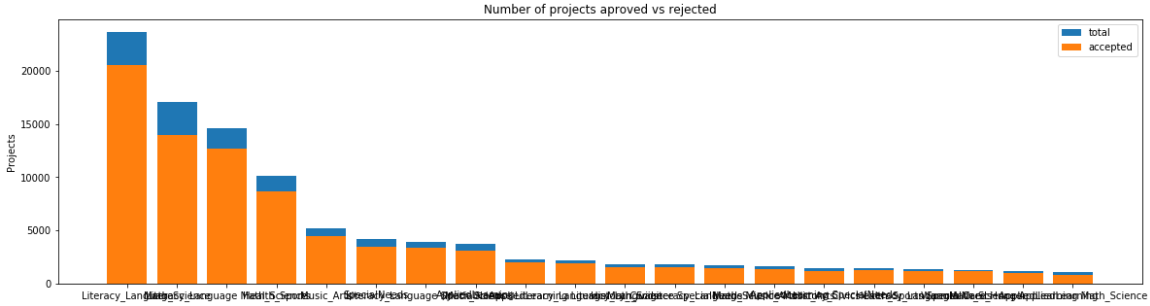
```
dft['clean_categories'] = cat_list
dft.drop(['project_subject_categories'], axis=1, inplace=True)
dft.head(2)
```

Out[50]:

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

In [51]:

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



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

==========

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

SUMMARY :

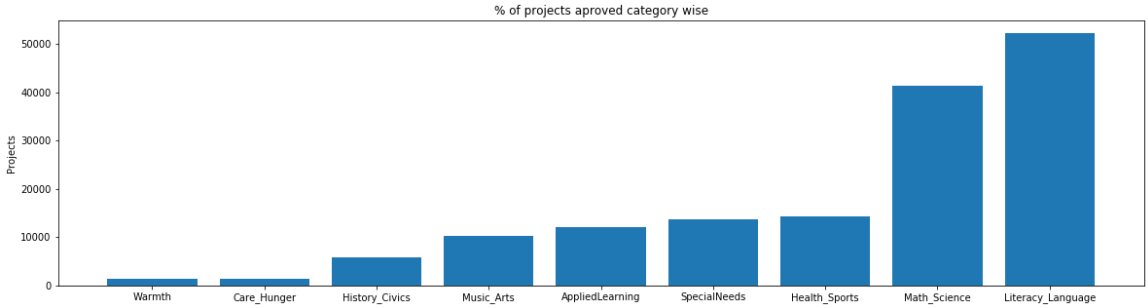
1. Projects belonging to the Literacy and Language categories have the highest number of projects proposed under. The maximum number of accepted projects also belong to this category, having an acceptance rate of nearly 87%.
2. Projects belonging to both Maths and Science have acceptance rate of nearly 82% while introducing the concept of Literacy and Language to this can increase its accpetance rate to nearly 87%
3. There is a lot of variablity in the total number of projects proposed per Category of the project.
4. Projects belonging to both Maths and Science when combined with Applied Learning has the least number of projects proposed as well approved.
5. There is also Variability in Acceptance rate, projects under the category Warmth, Care and Hunger have an acceptance rate of 93.5%

In [52]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in dft['clean_categories'].values:
    my_counter.update(word.split())
```

In [53]:

```
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 [54]:

```
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 (While Considering individual Categories for each project) :

1. The highest number of projects are registered under Literacy and Language with 52,239 projects, followed by Maths and Science having 41,421 projects.
2. There are only 1388 projects under the category of Warmth , Care or Hunger.

Univariate Analysis: project_subject_subcategories

In [56]:

```
sub_catogories = list(dft['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [57]:

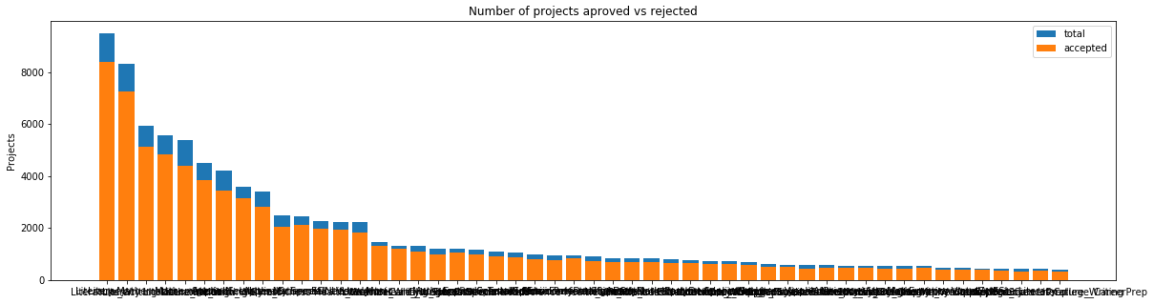
```
dft['clean_subcategories'] = sub_cat_list
dft.drop(['project_subject_subcategories'], axis=1, inplace=True)
dft.head(2)
```

Out[57]:

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

In [58]:

```
univariate_barplots(dft, '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  0.876
126
127                                  ESL                  349    421  0.828
979
79                   College_CareerPrep                  343    421  0.814
727
17   AppliedSciences Literature_Writing                  361    420  0.859
524
3    AppliedSciences College_CareerPrep                  330    405  0.814
815
```

SUMMARY :

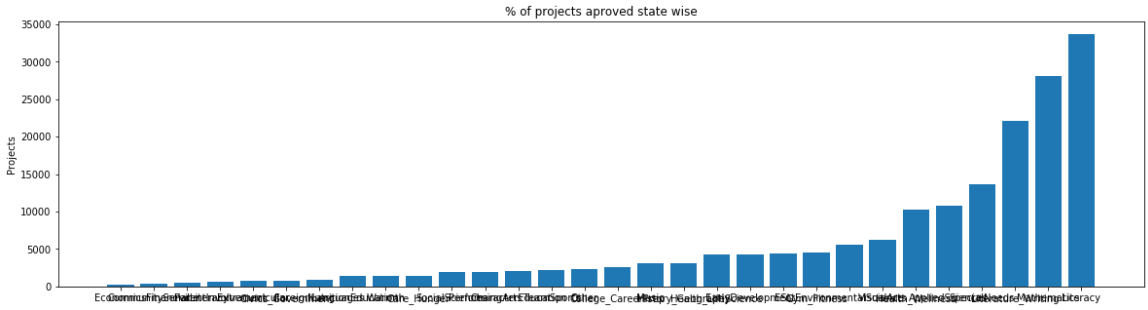
1. The sub-Category Literacy has the highest number of projects approved with 8371 projects. Also the acceptance rate is 88%.
2. The sub-Category Health and Wellness have the lowest number of projects proposed with 3,583 projects only.

In [59]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in dft['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [60]:

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


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

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



In [62]:

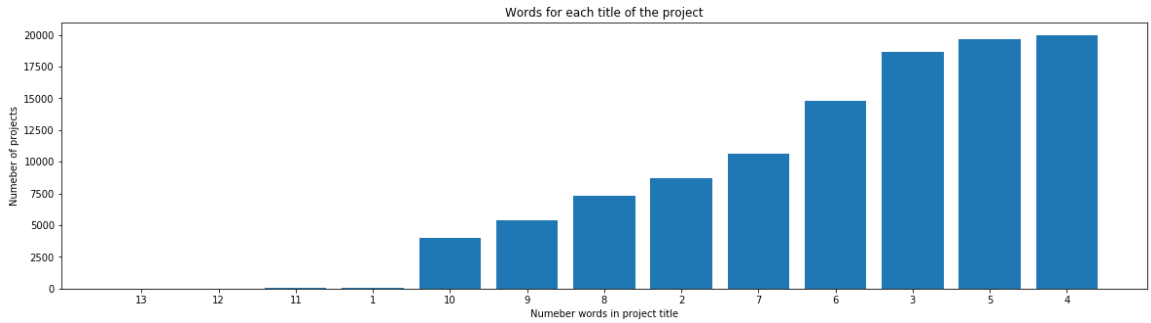
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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
```

Univariate Analysis: Text features (Title)

In [63]:

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
word_count = dft['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 :

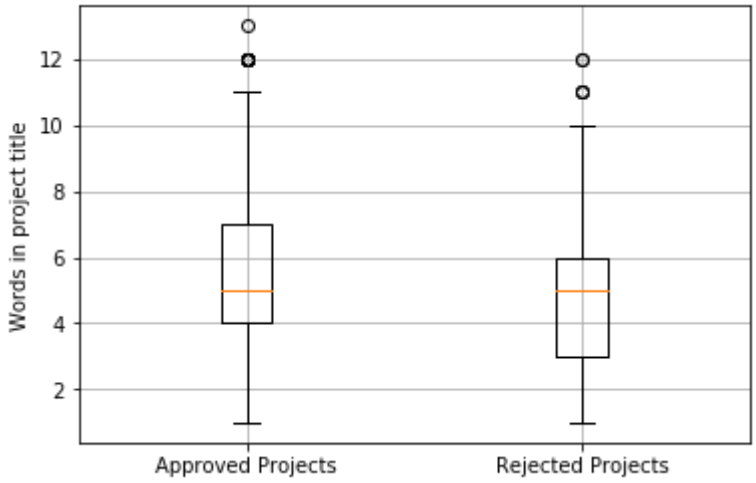
1. Most of the projects have 4 words in the title. 2.There are hardly any project titles containing more than 10 words. 3.Roughly most of the projects have 3, 4 or 5 words in the title.

In [64]:

```
approved_title_word_count = dft[dft['project_is_approved']==1]['project_title'].str.split().apply(len)
approved_title_word_count = approved_title_word_count.values

rejected_title_word_count = dft[dft['project_is_approved']==0]['project_title'].str.split().apply(len)
rejected_title_word_count = rejected_title_word_count.values
```

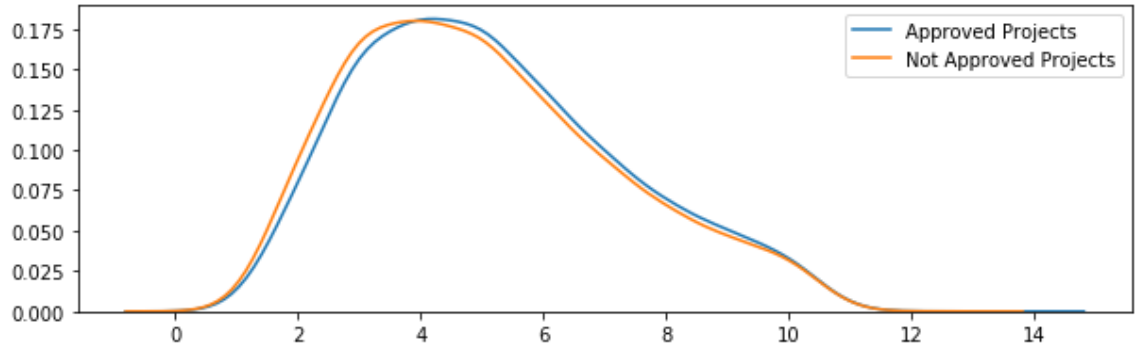
In [65]:

```
# 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 [66]:

```
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 : The number of Projects approved have a slightly more number of words in the Title when compared to the Rejected Projects. The Boxplots use the Percentiles while the above graph used Probability densities.

Univariate Analysis: Text features (Project Essay's)

In [68]:

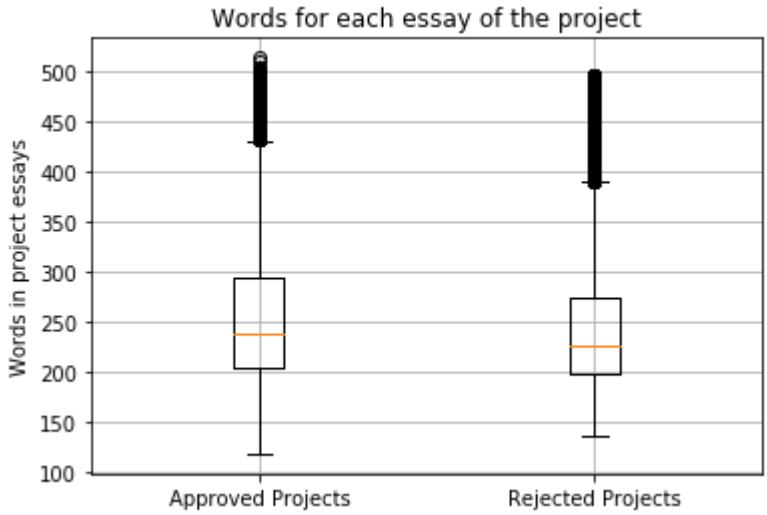
```
# merge two column text dataframe:
dft["essay"] = dft["project_essay_1"].map(str) +\
                        dft["project_essay_2"].map(str) + \
                        dft["project_essay_3"].map(str) + \
                        dft["project_essay_4"].map(str)
```

In [69]:

```
approved_word_count = dft[dft['project_is_approved']==1]['essay'].str.split().apply(len
)
approved_word_count = approved_word_count.values

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

In [70]:

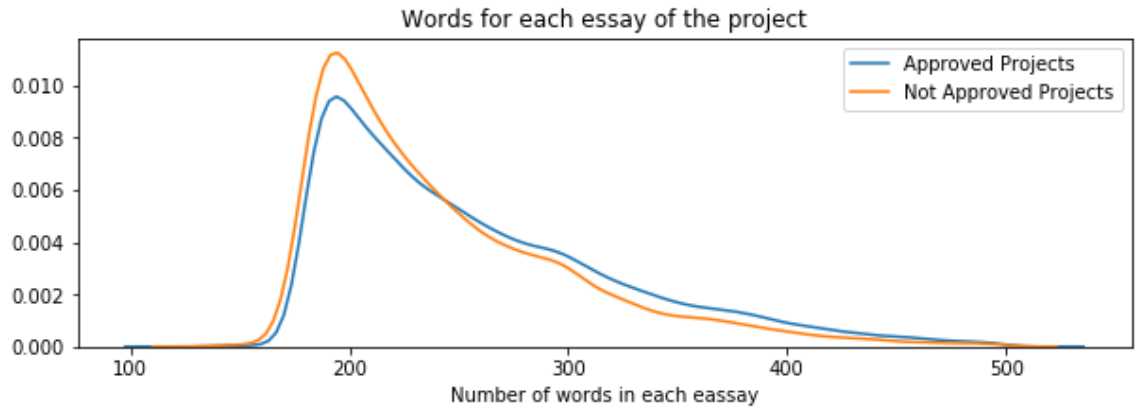
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()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



SUMMARY : Approved projects have a slightly more number of words in the project essays when compared to the projects that have not been approved. This difference can be noticed in the percentile difference after the 50.0

In [71]:

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



SUMMARY : The number of words in the Project Essays of Approved Projects are slightly more than the number of words in the Project Essays of the Rejected Projects. This can be noticed by looking at the Blue Line (PDF Curve of Approved Projects) which is denser for words more than 240 to almost 480 or 500.

Univariate Analysis: Cost per project

In [72]:

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

Out[72]:

	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 [73]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = dfr.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[73]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [74]:

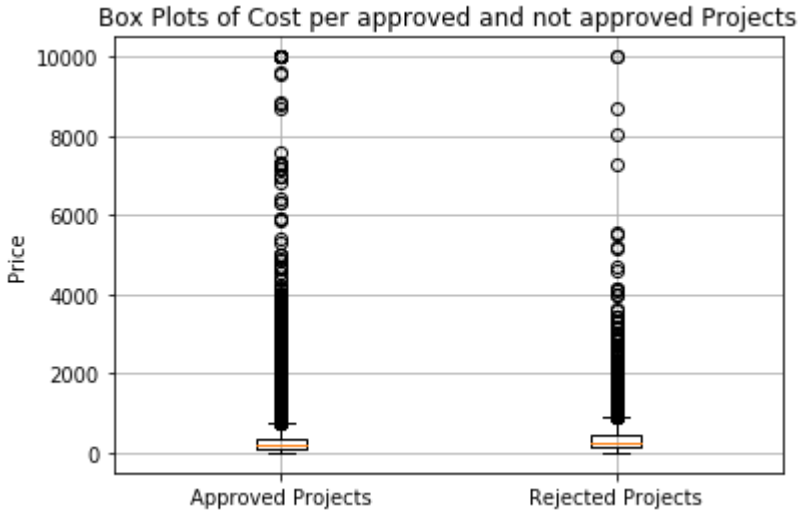
```
# join two dataframes in python:
dft = pd.merge(dft, price_data, on='id', how='left')
```

In [75]:

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

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

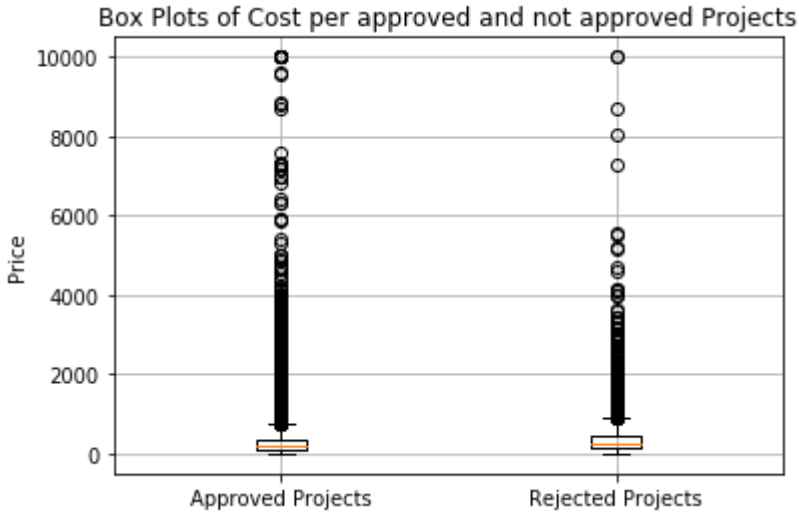
In [76]:

```
# 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()
```



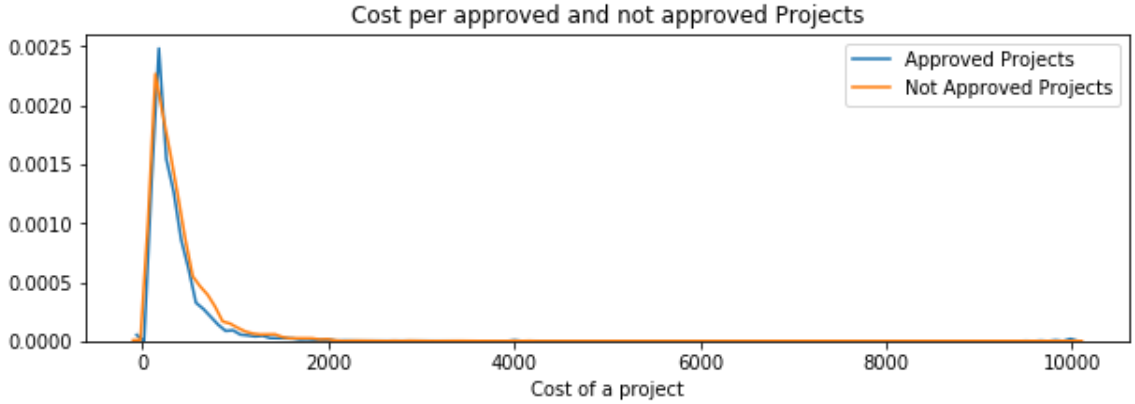
In [77]:

```
# 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 [78]:

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



SUMMARY : Not much can be understood from the box plot depicting the Cost involved per project. We can generalise from the PDF curves that mostly Projects that are very costly are usually not approved.

In [80]:

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

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

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

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

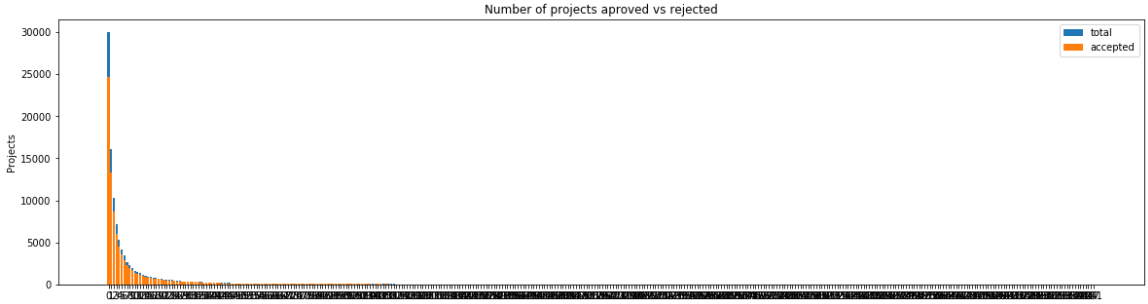
```
+------------+------------------+----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+------------------+----------------------+
|     0      |        0.66       |          1.97        |
|     5      |       13.59       |          41.9        |
|     10     |       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 : 1.The Maximum price for any project should be less than 10,000 dollars. 2.The Maximum price for any project should be less than 10,000 dollars. 3.The approved projects tend to have lower cost when compared to the projects that have not been approved. This can be noticed by looking at the percentile values. The 50th percentile Cost value for an approved project is 199 dollars while for the cost for the not approved projects is 263 dollars

Univariate Analysis: teacher_number_of_previously_posted_projects

In [81]:

```
univariate_barplots(dft, 'teacher_number_of_previously_posted_projects', 'project_is_approved' , top=False)
```




```
   teacher_number_of_previously_posted_projects  project_is_approved  tota
l  \
0                                             0                24652  3001
4
1                                             1                13329  1605
8
2                                             2                 8705  1035
0
3                                             3                 5997   711
0
4                                             4                 4452   526
6

        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  to
tal  \
242                                           242                   1
1
268                                           270                   1
1
234                                           234                   1
1
335                                           347                   1
1
373                                           451                   1
1

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

SUMMARY : 1.We observe that it is not mandatory for a teacher to have proposed any project prior. Maximum number of teachers, nearly 82% of the approved projects have been submitted by teachers with no prior project proposals. New talent and efforts are well appreciated. 2.Very few teachers who have proposed more than 20 projects have got approval. But the rate of approval is Higher given the teacher has proposed atleast 19 different projects. 3.There is alot of variability in the number of projects previously proposed by the teacher varying from 0 to more than 20.

project_resource_summary

In [82]:

```
## Let us separate the data and carry out our work only on the required Project Resourc
e Summaries.

summaries = []

for a in dft["project_resource_summary"] :
    summaries.append(a)

summaries[0:10]
```

Out[82]:

```
['My students need opportunities to practice beginning reading skills in E
nglish at home.',
 'My students need a projector to help with viewing educational programs',
 'My students need shine guards, athletic socks, Soccer Balls, goalie glov
es, and training materials for the upcoming Soccer season.',
 'My students need to engage in Reading and Math in a way that will inspir
e them with these Mini iPads!',
 'My students need hands on practice in mathematics. Having fun and person
alized journals and charts will help them be more involved in our daily Ma
th routines.',
 'My students need movement to be successful. Being that I have a variety
of students that have all different types of needs, flexible seating would
assist not only these students with special needs, but all students.',
 'My students need some dependable laptops for daily classroom use for rea
ding and math.',
 'My students need ipads to help them access a world of online resources t
hat will spark their interest in learning.',
 "My students need three devices and three management licenses for small g
roup's easy access to newly-implemented online programs--Go Noodle Plus, f
or increased in-class physical activity and Light Sail, an interactive rea
ding program.",
 'My students need great books to use during Independent Reading, Read Alo
uds, Partner Reading and Author Studies.']
```

In [83]:

```
## The length of the obtained list of Project summaries should match the total number o
f project summaries in
## the project data. Just to ensure
len(summaries)
```

Out[83]:

```
109248
```

In [84]:

```
## Identifying the numbers from the project summaries and storing the values as a key v
alue pair in a dictionary to
## avoid missing the position of the value within the huge ocean of summary data.

numeric_summary_values = {}

for x in tqdm(range(len(summaries))):
    for s in summaries[x].split():
        if s.isdigit() :
            numeric_summary_values[x] = int(s)
```

```
100%|████████████████████████████████████████████████████████████████
██| 109248/109248 [00:01<00:00, 89769.35it/s]
```

In [85]:

```
numeric_summary_values[14]
```

Out[85]:

```
5
```

In [86]:

```
## We only have the key value pairs for Summaries containing Numeric values, so in this
step

numeric_digits = {}

for c in range(len(summaries)) :
    if c in numeric_summary_values.keys() :
        numeric_digits[c] = numeric_summary_values[c]
    else :
        numeric_digits[c] = 0
```

In [87]:

```
for i in range (20) :
    print(numeric_digits[i])
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
5
0
2
0
0
7
```

In [88]:

```
len(numeric_digits)
```

Out[88]:

```
109248
```

In [89]:

```
## Converting the key value pairs to 1 or 0 based on presence of Numeric Values.

digit_in_summary = []

for a in numeric_digits.values() :
    if a > 0 :
        digit_in_summary.append(1)
    else :
        digit_in_summary.append(0)
```

In [90]:

```
digit_in_summary[0:20]
```

Out[90]:

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1]
```

In [91]:

```
dft['digit_in_summary'] = digit_in_summary

dft.head(20)
```

Out[91]:

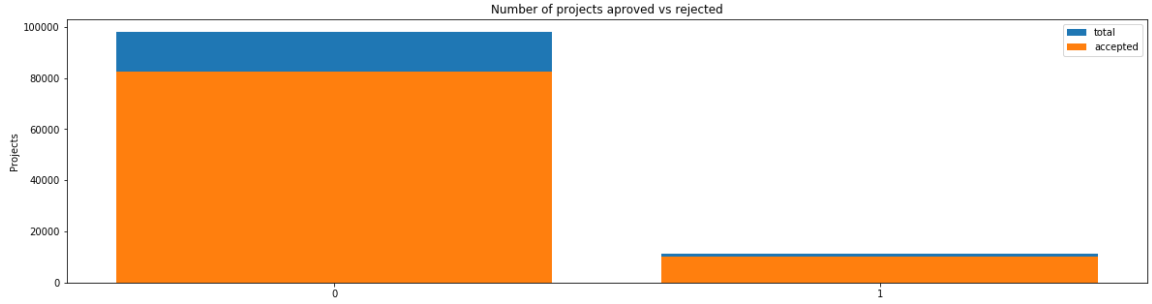
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	
5	141660	p154343	a50a390e8327a95b77b9e495b58b9a6e	Mrs.	FL	
6	21147	p099819	9b40170bfa65e399981717ee8731efc3	Mrs.	CT	
7	94142	p092424	5bfd3d12fae3d2fe88684bbac570c9d2	Ms.	GA	
8	112489	p045029	487448f5226005d08d36bdd75f095b31	Mrs.	SC	
9	158561	p001713	140eeac1885c820ad5592a409a3a8994	Ms.	NC	

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
10	43184	p040307	363788b51d40d978fe276bcb1f8a2b35	Mrs.	CA	
11	127083	p251806	4ba7c721133ef651ca54a03551746708	Ms.	CA	
12	19090	p051126	5e52c92b7e3c472aad247a239d345543	Mrs.	NY	
13	15126	p003874	178f6ae765cd4e0fb143a77c47fd65e2	Mrs.	OK	
14	62232	p233127	424819801de22a60bba7d0f4354d0258	Ms.	MA	
15	67303	p132832	bb6d6d054824fa01576ab38dfa2be160	Ms.	TX	
16	127215	p174627	4ad7e280fddff889e1355cc9f29c3b89	Mrs.	FL	
17	157771	p152491	e39abda057354c979c5b075cffbe5f88	Ms.	NV	
18	122186	p196421	fcd9b003fc1891383f340a89da02a1a6	Mrs.	GA	
19	146331	p058343	8e07a98deb1bc74c75b97521e05b1691	Ms.	OH	

20 rows × 21 columns

In [92]:

```
univariate_barplots(dft, 'digit_in_summary', 'project_is_approved', top=2)
```




```
   digit_in_summary  project_is_approved  total       Avg
0                 0                82563  98012  0.842376
1                 1                10143  11236  0.902723
==================================================
   digit_in_summary  project_is_approved  total       Avg
0                 0                82563  98012  0.842376
1                 1                10143  11236  0.902723
```

SUMMARY : 1.The project summaries containing numeric values have a very high acceptance rate of 90%. Well, proper numbered requirements suggest clarity in the proposals and hence Alot of people tend to donate for a better cause, that is to help children 2.It is obvious from the graph that majority of the projects do not have numeric values stating the requirement of certain products

Text preprocessing

Essay Text

In [93]:

```
dft.head(4)
```

Out[93]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proj
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	

4 rows × 21 columns

In [94]:

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

My students are English learners that are working on English as their seco nd or third languages. We are a melting pot of refugees, immigrants, and n ative-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 repres ented with the families within our school. Each student brings a wealth o f knowledge and experiences to us that open our eyes to new cultures, beli efs, and respect.\"The limits of your language are the limits of your worl d.\"-Ludwig Wittgenstein Our English learner's have a strong support syst em at home that begs for more resources. Many times our parents are learn ing to read and speak English along side of their children. Sometimes thi s creates barriers for parents to be able to help their child learn phonet ics, letter recognition, and other reading skills.\r\n\r\nBy providing the se dvd's and players, students are able to continue their mastery of the E nglish language even if no one at home is able to assist. All families wi th students within the Level 1 proficiency status, will be a offered to be a part of this program. These educational videos will be specially chosen by the English Learner Teacher and will be sent home regularly to watch. The videos are to help the child develop early reading skills.\r\n\r\nPare nts that do not have access to a dvd player will have the opportunity to c heck out a dvd player to use for the year. The plan is to use these video s and educational dvd's for the years to come for other EL students.\r\nna nnan

==========

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% a re minority students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parad e to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and gam es. At the end of the year the school hosts a carnival to celebrate the ha rd work put in during the school year, with a dunk tank being the most pop ular activity.My students will use these five brightly colored Hokki stool s in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an i ndividual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by th e students who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is mis sing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group wi th me on the Hokki Stools, they are always moving, but at the same time do ing their work. Anytime the students get to pick where they can sit, the H okki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my student s to do desk work and move at the same time. These stools will help studen ts to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, th ese chairs will take away the barrier that exists in schools for a child w ho can't sit still.nannan

==========

How do you remember your days of school? Was it in a sterile environment w ith plain walls, rows of desks, and a teacher in front of the room? A typi cal day in our room is nothing like that. I work hard to create a warm inv iting themed room for my students look forward to coming to each day.\r\n \r\nMy class is made up of 28 wonderfully unique boys and girls of mixed r

aces 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 classrooms. 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 more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take 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 thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

==========

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe 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 coming to school and come eager to learn and explore. Have you ever felt like you had ants in 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. \r\nThey also want to learn through games, my kids don't 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

==========

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest 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 lunch. We aren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character.In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn'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 the lessons as meaningful. But with the blue tooth speaker my students will be able 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 letter, words and pictures for students to learn about different letters and it is more accessible.nannan

==========

In [95]:

```
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 [96]:

```
sent = decontracted(dft['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. The
y are eager beavers and always strive to work their hardest working past t
heir limitations. \r\n\r\nThe materials we have are the ones I seek out fo
r my students. I teach in a Title I school where most of the students rece
ive free or reduced price lunch.  Despite their disabilities and limitatio
ns, my students love coming to school and come eager to learn and explore.
Have you ever felt like you had ants in your pants and you needed to groov
e and move as you were in a meeting? This is how my kids feel all the tim
e. 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 enha
nces gross motor and in Turn fine motor skills. \r\nThey also want to lear
n through games, my kids do not want to sit and do worksheets. They want t
o 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 happe
n. My students will forget they are doing work and just have the fun a 6 y
ear old deserves.nannan
==================================================
```

In [97]:

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

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. The
y are eager beavers and always strive to work their hardest working past t
heir limitations.     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 f
ree or reduced price lunch.  Despite their disabilities and limitations, m
y students love coming to school and come eager to learn and explore.Have
you ever felt like you had ants in 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 gro
ss 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 succes
s. The number toss and color and shape mats can make that happen. My stude
nts will forget they are doing work and just have the fun a 6 year old des
erves.nannan
```

In [98]:

```
#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
language delays cognitive delays gross fine motor delays to autism They ar
e eager beavers and always strive to work their hardest working past their
limitations 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 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
like you had ants in your pants and you needed to groove and move as you w
ere 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 lo
ve 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 no
t 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 [99]:

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

In [100]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(dft['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 [03:00<00:00, 604.54it/s]
```

In [101]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[101]:

```
'my kindergarten students varied disabilities ranging speech language dela
ys cognitive delays gross fine motor delays autism they eager beavers alwa
ys strive work hardest working past limitations the materials ones i seek
students i teach title i school students receive free reduced price lunch
despite disabilities limitations students love coming school come eager le
arn explore have ever felt like ants pants needed groove move meeting this
kids feel time the want able move learn say wobble chairs answer i love de
velop core enhances gross motor turn fine motor skills they also want lear
n games kids not want sit worksheets they want learn count jumping playing
physical engagement key success the number toss color shape mats make happ
en my students forget work fun 6 year old deserves nannan'
```

In [102]:

```
# printing some random essays.
print(dft['project_title'].values[0])
print("="*50)
print(dft['project_title'].values[150])
print("="*50)
print(dft['project_title'].values[1000])
print("="*50)
print(dft['project_title'].values[20000])
print("="*50)
print(dft['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 [104]:

```
preprocessed_titles = []

for titles in tqdm(dft["project_title"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_titles.append(title.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
██| 109248/109248 [00:08<00:00, 13019.75it/s]
```

Preparing data for models

In [105]:

```
dft.columns
```

Out[105]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_ti
tle',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quant
ity',
       'digit_in_summary'],
      dtype='object')
```

Vectorizing Categorical data

In [106]:

```
# 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(dft['clean_categories'].values)
print(vectorizer.get_feature_names())


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

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

In [107]:

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


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutrition
Education', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'A
ppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Lit
eracy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [108]:

```
#One Hot Encode - School States
my_counter = Counter()
for state in dft['school_state'].values:
    my_counter.update(state.split())
```

In [109]:

```
school_state_cat_dict = dict(my_counter)
sorted_school_state_cat_dict = dict(sorted(school_state_cat_dict.items(), key=lambda kv
: kv[1]))
```

In [110]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()), lowe
rcase=False, binary=True)
vectorizer.fit(dft['school_state'].values)
print(vectorizer.get_feature_names())

school_state_categories_one_hot = vectorizer.transform(dft['school_state'].values)
print("Shape of matrix after one hot encoding ",school_state_categories_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', 'N
V', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'M
A', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'N
Y', 'TX', 'CA']
Shape of matrix after one hot encoding  (109248, 51)
```

In [111]:

```
#One Hot Encode - Project Grade Category
my_counter = Counter()
for project_grade in dft['project_grade_category'].values:
    my_counter.update(project_grade.split())
```

In [112]:

```
project_grade_cat_dict = dict(my_counter)
sorted_project_grade_cat_dict = dict(sorted(project_grade_cat_dict.items(), key=lambda
kv: kv[1]))
```

In [113]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), low
ercase=False, binary=True)
vectorizer.fit(dft['project_grade_category'].values)
print(vectorizer.get_feature_names())

project_grade_categories_one_hot = vectorizer.transform(dft['project_grade_category'].v
alues)
print("Shape of matrix after one hot encoding ",project_grade_categories_one_hot.shape)
```

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

In [114]:

```
#One Hot Encode - Teacher Prefix
my_counter = Counter()
for teacher_prefix in dft['teacher_prefix'].values:
    teacher_prefix = str(teacher_prefix)
    my_counter.update(teacher_prefix.split())
```

In [115]:

```
teacher_prefix_cat_dict = dict(my_counter)
sorted_teacher_prefix_cat_dict = dict(sorted(teacher_prefix_cat_dict.items(), key=lambd
a kv: kv[1]))
```

In [116]:

```
## we use count vectorizer to convert the values into one hot encoded features
## Unlike the previous Categories this category returns a
## ValueError: np.nan is an invalid document, expected byte or unicode string.
## The link below explains h0w to tackle such discrepancies.
## https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueer
ror-np-nan-is-an-invalid-document/39308809#39308809

vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys()), lo
wercase=False, binary=True)
vectorizer.fit(dft['teacher_prefix'].values.astype("U"))
print(vectorizer.get_feature_names())

teacher_prefix_categories_one_hot = vectorizer.transform(dft['teacher_prefix'].values.a
stype("U"))
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot.shape
)
```

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

Vectorizing Text data

Bag of words

In [117]:

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

In [118]:

```
print("There are {} unique words among the {} number of Project essays, considering atl
east 10 different projects has the same word".format(text_bow.shape[1], text_bow.shape[
0]))
```

```
There are 16623 unique words among the 109248 number of Project essays, co
nsidering atleast 10 different projects has the same word
```

Bag of Words on project_title

In [119]:

```
# We are considering only the words which appeared in at least 5 documents(rows or proj
ects).
vectorizer = CountVectorizer(min_df=5)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encoding ",title_bow.shape)
```

```
Shape of matrix after one hot encoding  (109248, 5107)
```

In [120]:

```
print ("There are {} unique words among the {} number of project titles, considering at
least 5 different projects has the same word ".format(title_bow.shape[1], title_bow.sha
pe[0]))
```

```
There are 5107 unique words among the 109248 number of project titles, con
sidering atleast 5 different projects has the same word
```

TFIDF vectorizer

In [121]:

```
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)
```

TFIDF Vectorizer on project_title

In [123]:

```
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects).
vectorizer = TfidfVectorizer(min_df=5)
title_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encoding ",title_tfidf.shape)
```

```
Shape of matrix after one hot encoding  (109248, 5107)
```

Using Pretrained Models: Avg W2V

In [124]:

```
# 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
```

In [125]:

```
model = loadGloveModel('glove.42B.300d.txt')
```

```
Loading Glove Model

1917495it [14:34, 2192.01it/s]

Done. 1917495  words loaded!
```

In [126]:

```
words = []

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

for i in preprocessed_titles:
    words.extend(i.split(' '))
```

In [127]:

```
print("all the words in the corpus", len(words))
```

```
all the words in the corpus 17014413
```

In [128]:

```
words = set(words)
print("the unique words in the corpus", len(words))
```

```
the unique words in the corpus 58968
```

In [129]:

```
inter_words = set(model.keys()).intersection(words)

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

```
The number of words that are present in both glove vectors and our coupus
51503 ( 87.341 %)
```

In [130]:

```
words_corpus = {}

words_glove = set(model.keys())

for i in words:
    if i in words_glove:
        words_corpus[i] = model[i]

print("word 2 vec length", len(words_corpus))
```

```
word 2 vec length 51503
```

In [131]:

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

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

In [132]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-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 [133]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

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

```
100%|████████████████████████████████████████████████████████████████
████| 109248/109248 [01:57<00:00, 925.87it/s]

109248
300
```

In [134]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(preprocessed_titles): # for each title
    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_titles.append(vector)

print(len(avg_w2v_vectors_titles))
print(len(avg_w2v_vectors_titles[0]))
```

```
100%|████████████████████████████████████████████████████████████████
███| 109248/109248 [00:15<00:00, 7134.33it/s]

109248
300
```

Using Pretrained Models: TFIDF weighted W2V

In [135]:

```
# 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 [136]:

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

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

```
100%|████████████████████████████████████████████████████████████████
████| 109248/109248 [10:44<00:00, 169.56it/s]

109248
300
```

In [137]:

```
# Similarly you can vectorize for title also

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

In [138]:

```
# average Word2Vec
# compute average word2vec for each Project Title
tfidf_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this
 list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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 [00:12<00:00, 8612.89it/s]

109248
300
```

Vectorizing Numerical features

Vectorizing - Price (Numerical Data)

In [140]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ...
 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(dft['price'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
print("Mean : {}".format(price_scalar.mean_[0]))
print("Standard deviation : {}".format(np.sqrt(price_scalar.var_[0])))
# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(dft['price'].values.reshape(-1, 1))
```

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

In [141]:

```
price_standardized
```

Out[141]:

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

Summary : We observe that on an average Each project costs nearly 298 Dollars. With a Standard Deviation of 368 dollars. So , mostly majority of the projects are less than 1000 Dollars

Vectorizing - Quantity (Numerical Data)

In [144]:

```
import warnings
warnings.filterwarnings("ignore")

quantity_scalar = StandardScaler()

## Finding the mean and standard deviation of this data
quantity_scalar.fit(dft['quantity'].values.reshape(-1,1))

print("Mean : {}".format(quantity_scalar.mean_[0]))

print("Standard deviation : {}".format(np.sqrt(quantity_scalar.var_[0])))

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

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

In [145]:

```
quantity_standardized
```

Out[145]:

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

Summary : The projects on an average require atleast 17 Different of similar items. We observe that the Price paid is generally for the purchase of these Items. Donors can choose on projects to donate based on the Items provided to aid the Students of any Grade

Vectorizing - Number of Projects Proposed Previously by the Teacher (Numerical Data)

In [147]:

```
prev_projects_scalar = StandardScaler()

## Finding the mean and standard deviation of this data
prev_projects_scalar.fit(dft['teacher_number_of_previously_posted_projects'].values.res
hape(-1,1))

print("Mean : {}".format(prev_projects_scalar.mean_[0]))

print("Standard deviation : {}".format(np.sqrt(prev_projects_scalar.var_[0])))

# Now standardize the data with above maen and variance.
prev_projects_standardized = prev_projects_scalar.transform(dft['teacher_number_of_prev
iously_posted_projects'].values.reshape(-1, 1))
```

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

In [148]:

```
prev_projects_standardized
```

Out[148]:

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

Summary : We observe that Teachers generally on an average propose atleast 11 different projects. Well, The teachers are indeed actively seeking help to aid for the betterment of the students in their locality.

Merging all the above features

In [149]:

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

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

In [151]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[151]:

```
(109248, 16663)
```

TSNE with `BOW` encoding of `project_title` feature (5000 Data Entries)

In [152]:

```
print("The Shape of Data matrices for Categorical Data are :")
print("\n")
print("The Shape of Data Matrix for different Categories of projects is : {}".format(categories_one_hot.shape))
print("The Shape of Data Matrix for different Sub-categories of projects is : {}".format(sub_categories_one_hot.shape))
print("The Shape of Data Matrix with respect to Projects from a particular State in the United States is : {}".format(school_state_categories_one_hot.shape))
print("The Shape of the Data Matrix of the different projects with respect to the Grades of the students is : {} ".format(project_grade_categories_one_hot.shape))
print("The Shape of the Data Matrix with respect to title of the Teacher proposing the Teacher is : {}".format(teacher_prefix_categories_one_hot.shape))
print("\n")
print("="*100)
print("\n")
print("The Shape of Data matrices for Numerical Data are :")
print("\n")
print("The Shape of the Data Matrix for price of the projects is : {}".format(price_standardized.shape))
print("The Shape of the Data Matrix for Quantity of the items for the projects is : {}".format(quantity_standardized.shape))
print("The Shape of the Data Matrix for the Number of Projects Proposed Previously by the Teacher is : {}".format(prev_projects_standardized.shape))
print("\n")
print("="*100)
print("\n")
print("TITLE BOW : {}".format(title_bow.shape))
print("\n")
print("TITLE TFIDF : {}".format(title_tfidf.shape))
print("\n")
print("TITLE AVG W2V : ({}, {})".format(len(avg_w2v_vectors_titles), len(avg_w2v_vectors_titles[0])))
print("\n")
print("TITLE TFIDF W2V : ({}, {})".format(len(tfidf_w2v_vectors_title), len(tfidf_w2v_vectors_title[0])))
```

```
The Shape of Data matrices for Categorical Data are :


The Shape of Data Matrix for different Categories of projects is : (10924
8, 9)
The Shape of Data Matrix for different Sub-categories of projects is : (10
9248, 30)
The Shape of Data Matrix with respect to Projects from a particular State
in the United States is : (109248, 51)
The Shape of the Data Matrix of the different projects with respect to the
Grades of the students is : (109248, 5)
The Shape of the Data Matrix with respect to title of the Teacher proposin
g the Teacher is : (109248, 6)


==========================================================================
==========================


The Shape of Data matrices for Numerical Data are :


The Shape of the Data Matrix for price of the projects is : (109248, 1)
The Shape of the Data Matrix for Quantity of the items for the projects is
: (109248, 1)
The Shape of the Data Matrix for the Number of Projects Proposed Previousl
y by the Teacher is : (109248, 1)


==========================================================================
==========================


TITLE BOW : (109248, 5107)


TITLE TFIDF : (109248, 5107)


TITLE AVG W2V : (109248, 300)


TITLE TFIDF W2V : (109248, 300)
```

In [153]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, school_state_categories_one_hot
, project_grade_categories_one_hot, teacher_prefix_categories_one_hot, price_standardiz
ed, quantity_standardized, prev_projects_standardized, title_bow))
X.shape
```

Out[153]:

```
(109248, 5211)
```

In [154]:

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

In [157]:

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

Out[157]:

```
5000
```

In [158]:

```
tsne_data_b = np.vstack((tsne_data_b.T, labels_new)).T
tsne_df_b = pd.DataFrame(tsne_data_b, columns = ("1st_Dim","2nd_Dim","Labels"))
```

In [159]:

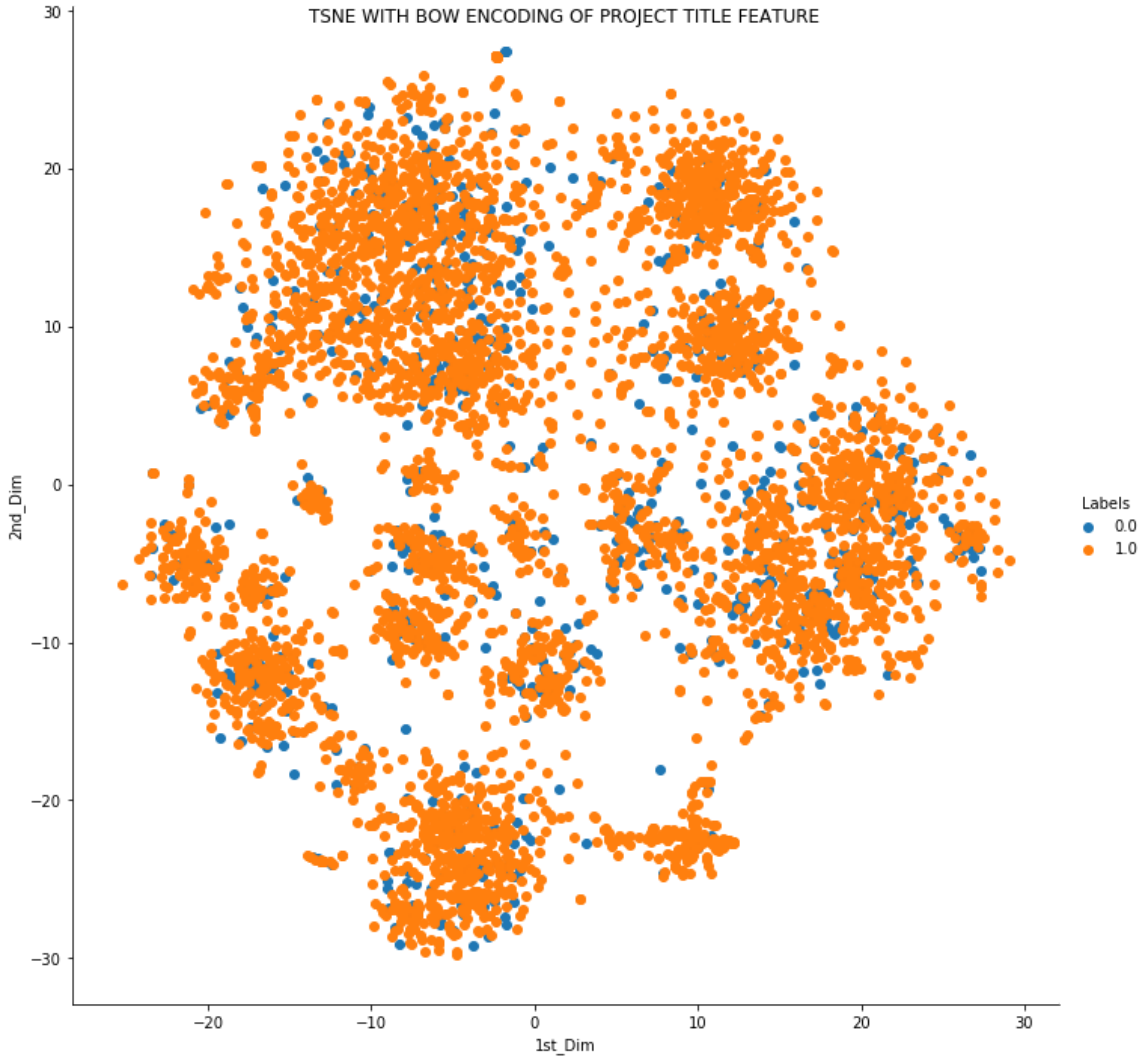
```
tsne_df_b.shape
```

Out[159]:

```
(5000, 3)
```

In [160]:

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

sns.FacetGrid(tsne_df_b, hue = "Labels", size = 10).map(plt.scatter, "1st_Dim", "2nd_Di
m").add_legend().fig.suptitle("TSNE WITH BOW ENCODING OF PROJECT TITLE FEATURE ")
plt.show()
```


TSNE WITH BOW ENCODING OF PROJECT TITLE FEATURE
2nd_Dim
1st_Dim
Labels
0.0
1.0
30
20
10
0
−10
−20
−30
−20
−10
0
10
20
30

Summary :

1. We observe alot of overlapping in the datapoints and the points are well scattered, unable to draw any proper conclusion

TSNE with `TFIDF` encoding of `project_title` feature (5000 Data Entries)

In [161]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, school_state_categories_one_hot
, project_grade_categories_one_hot, teacher_prefix_categories_one_hot, price_standardiz
ed, quantity_standardized, prev_projects_standardized, title_tfidf))
X.shape
```

Out[161]:

```
(109248, 5211)
```

In [162]:

```
X = X.tocsr()
X_new = X[0:5000,:]
```

In [163]:

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

In [164]:

```
tsne_data_tfidf = np.vstack((tsne_data_tfidf.T, labels_new)).T
tsne_df_tfidf = pd.DataFrame(tsne_data_tfidf, columns = ("1st_Dim","2nd_Dim","Labels"))
```

In [165]:

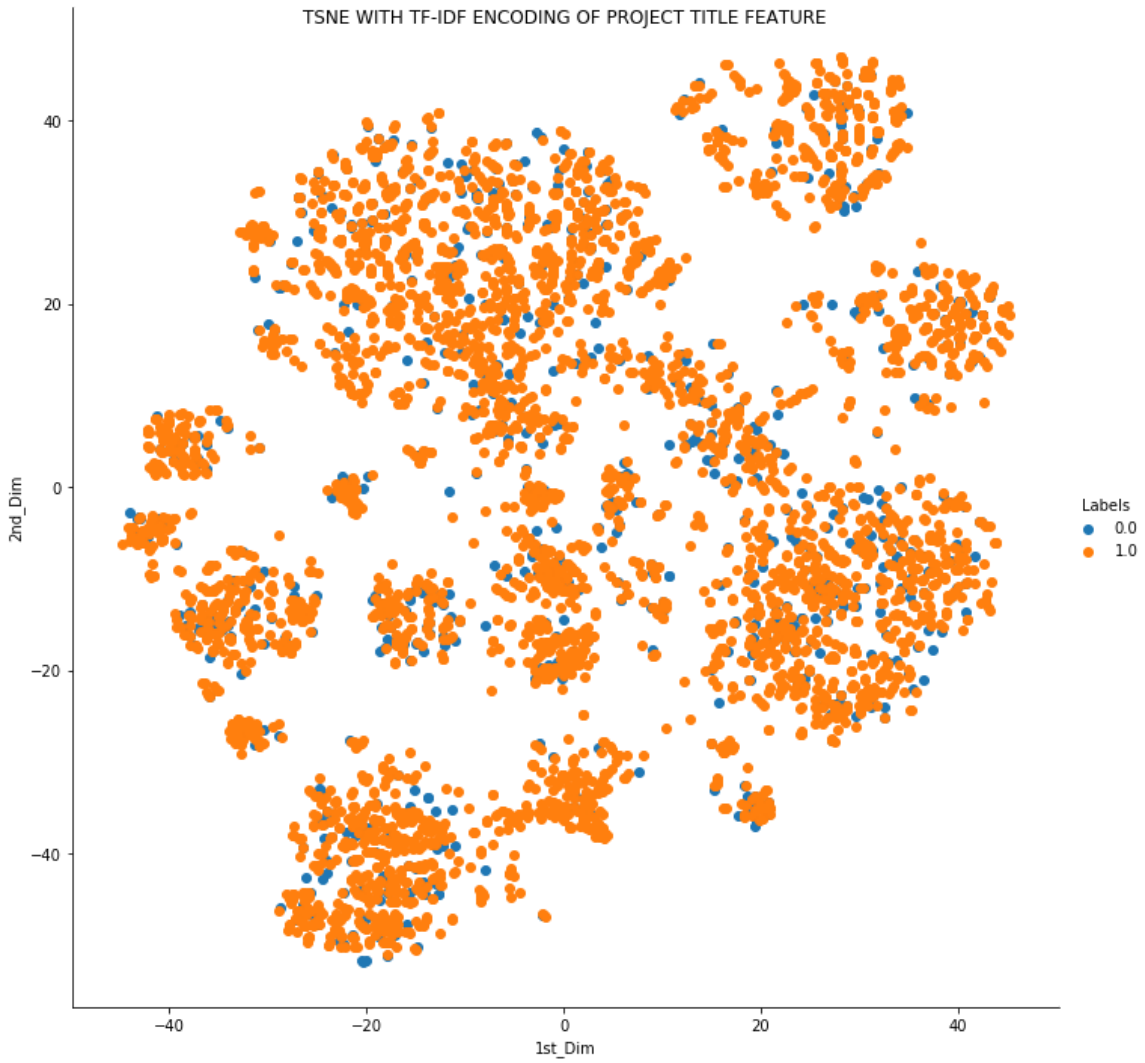
```
tsne_df_tfidf.shape
```

Out[165]:

```
(5000, 3)
```

In [166]:

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

sns.FacetGrid(tsne_df_tfidf, hue = "Labels", size = 10).map(plt.scatter, "1st_Dim", "2n
d_Dim").add_legend().fig.suptitle("TSNE WITH TF-IDF ENCODING OF PROJECT TITLE FEATURE "
)
plt.show()
```



Summary : The Blue and the Orange points do not form any clusters or accumulation of any type, Hence drawing conclusions seems to quite impossible with the current state of the T-SNE data using TF - IDF Encoding

TSNE with `AVG W2V` encoding of `project_title` feature (5000 Data Entries)

In [168]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, school_state_categories_one_hot
, project_grade_categories_one_hot, teacher_prefix_categories_one_hot, price_standardiz
ed, quantity_standardized, prev_projects_standardized, avg_w2v_vectors_titles))
X.shape
```

Out[168]:

```
(109248, 404)
```

In [169]:

```
X = X.tocsr()
X_new = X[0:5000,:]
```

In [170]:

```
X_new = X_new.toarray()
model = TSNE(n_components = 2, perplexity = 100.0, random_state = 0)
tsne_data_avg_w2v = model.fit_transform(X_new)
```

In [171]:

```
tsne_data_avg_w2v = np.vstack((tsne_data_avg_w2v.T, labels_new)).T
tsne_df_avg_w2v = pd.DataFrame(tsne_data_avg_w2v, columns = ("1st_Dim","2nd_Dim","Label
s"))
```

In [172]:

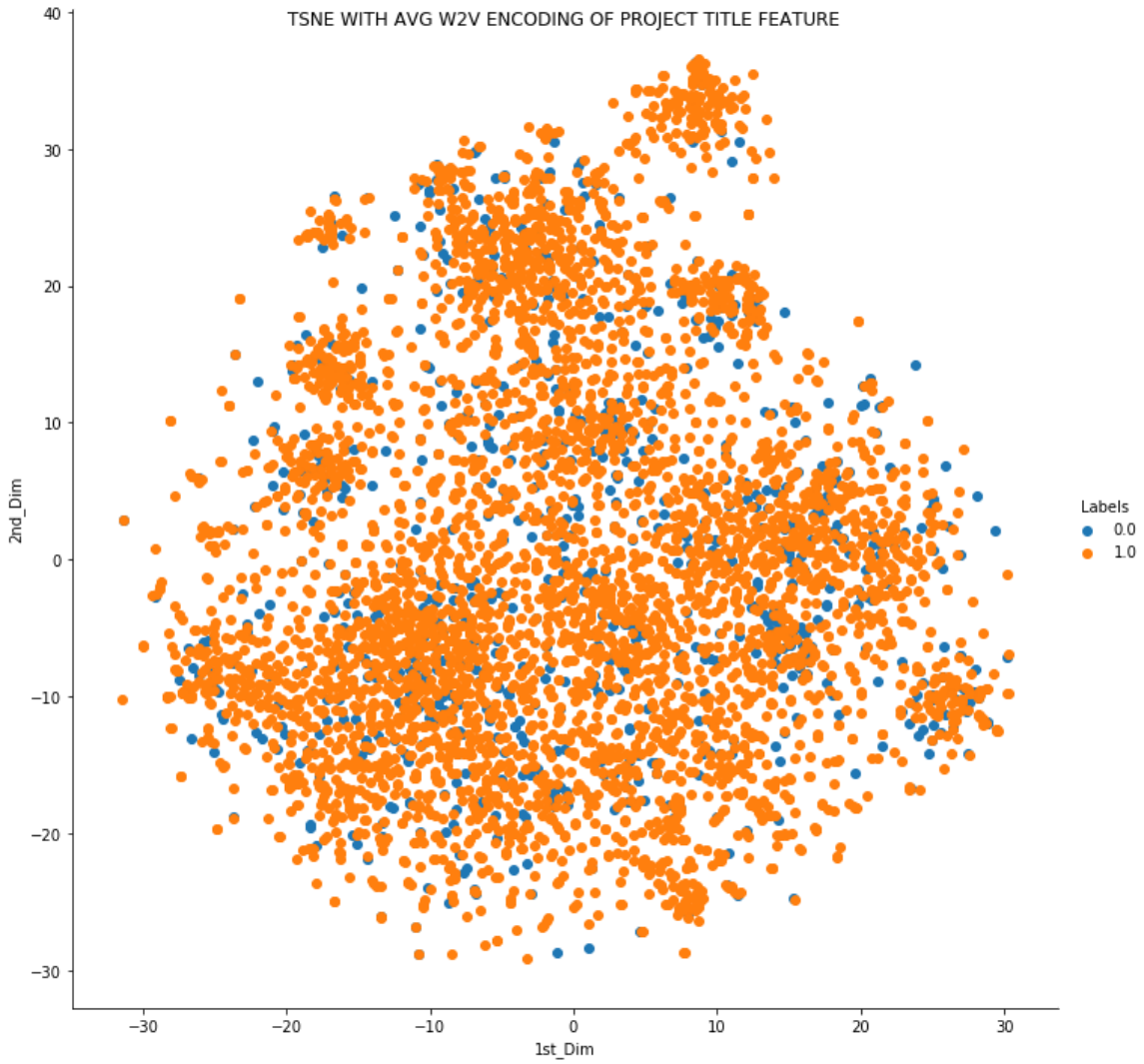
```
tsne_df_avg_w2v.shape
```

Out[172]:

```
(5000, 3)
```

In [173]:

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

sns.FacetGrid(tsne_df_avg_w2v, hue = "Labels", size = 10).map(plt.scatter, "1st_Dim",
"2nd_Dim").add_legend().fig.suptitle("TSNE WITH AVG W2V ENCODING OF PROJECT TITLE FEATU
RE ")
plt.show()
```



Summary : We do not observe any clusters for whether the Project is accepted or not accepted. Hence we are not able to achieve the desired result using Avg- Word2vec

TSNE with TFIDF Weighted W2V encoding of project_title feature (5000 Data Entries)

In [174]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, school_state_categories_one_hot
, project_grade_categories_one_hot, teacher_prefix_categories_one_hot, price_standardiz
ed, quantity_standardized, prev_projects_standardized, tfidf_w2v_vectors_title))
X.shape
```

Out[174]:

```
(109248, 404)
```

In [175]:

```
X = X.tocsr()
X_new = X[0:5000,:]
```

In [176]:

```
X_new = X_new.toarray()
model = TSNE(n_components = 2, perplexity = 100.0, random_state = 0)
tsne_data_tfidf_w2v = model.fit_transform(X_new)
```

In [177]:

```
tsne_data_tfidf_w2v = np.vstack((tsne_data_tfidf_w2v.T, labels_new)).T
tsne_df_tfidf_w2v = pd.DataFrame(tsne_data_tfidf_w2v, columns = ("1st_Dim","2nd_Dim","L
abels"))
```

In [178]:

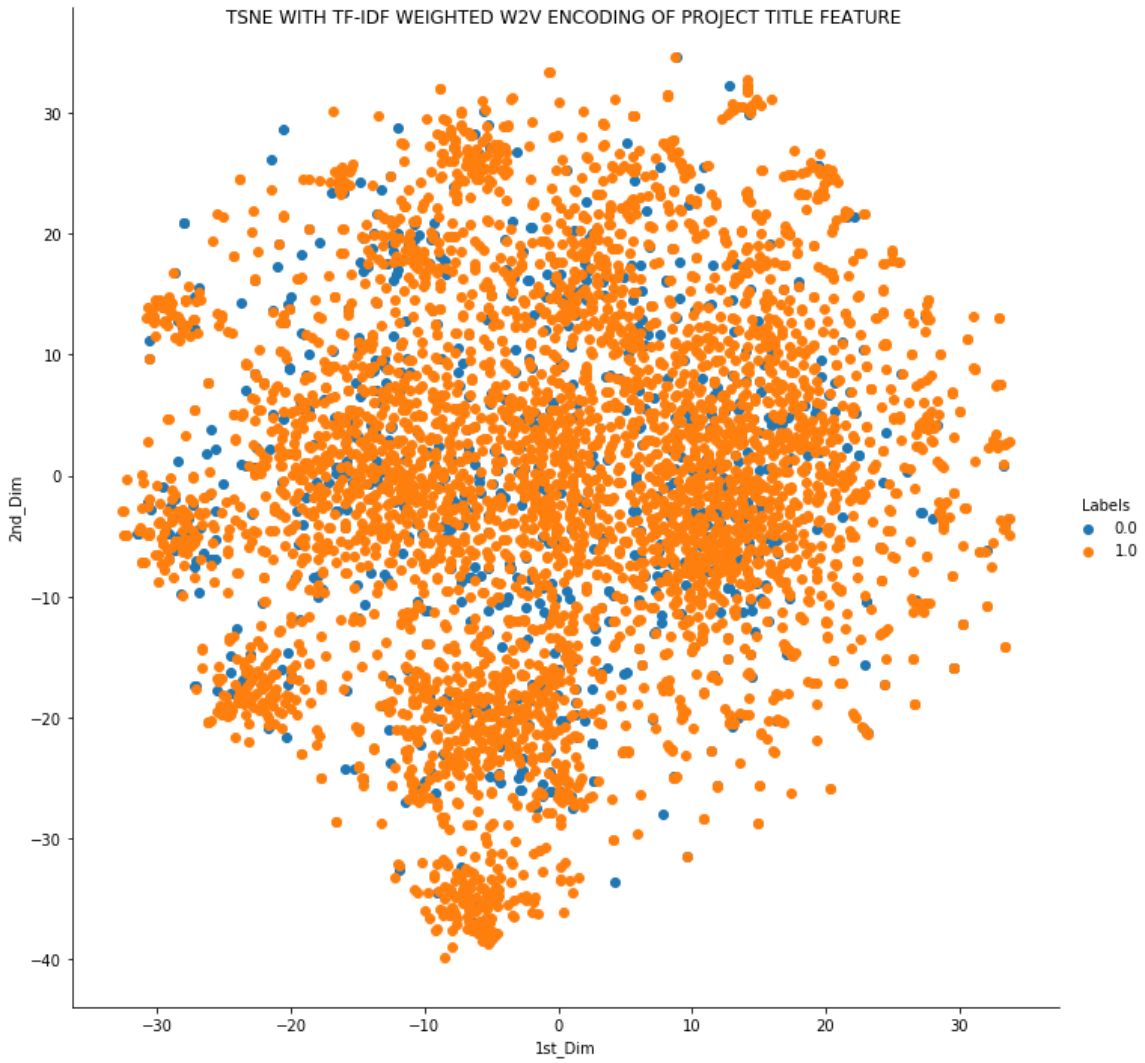
```
tsne_df_tfidf_w2v.shape
```

Out[178]:

```
(5000, 3)
```

In [179]:

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

sns.FacetGrid(tsne_df_tfidf_w2v, hue = "Labels", size = 10).map(plt.scatter, "1st_Dim",
"2nd_Dim").add_legend().fig.suptitle("TSNE WITH TF-IDF WEIGHTED W2V ENCODING OF PROJECT
TITLE FEATURE ")
plt.show()
```



Summary :

This visualisation of TSNE with TF-IDF Weighted Word2Vec does not seem to yield the expected result of clustering similar data points. Hence we would have to try any other method

TSNE with BOW, TFIDF, AVG W2V, TFIDF Weighted W2V encoding of project_title feature (5000 Data Entries)

In [180]:

```
X = hstack((categories_one_hot, sub_categories_one_hot, school_state_categories_one_hot
, project_grade_categories_one_hot, teacher_prefix_categories_one_hot, price_standardiz
ed, quantity_standardized, prev_projects_standardized, title_bow, title_tfidf, avg_w2v_
vectors_titles, tfidf_w2v_vectors_title))
X.shape
```

Out[180]:

```
(109248, 10918)
```

In [181]:

```
X = X.tocsr()
X_new = X[0:5000,:]
```

In [182]:

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

In [183]:

```
tsne_data_complete = np.vstack((tsne_data_complete.T, labels_new)).T
tsne_df_complete = pd.DataFrame(tsne_data_complete, columns = ("1st_Dim","2nd_Dim","Lab
els"))
```

In [184]:

```
tsne_df_complete.shape
```

Out[184]:

```
(5000, 3)
```

In [185]:

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

sns.FacetGrid(tsne_df_complete, hue = "Labels", size = 10).map(plt.scatter, "1st_Dim",
"2nd_Dim").add_legend().fig.suptitle("TSNE WITH BOW, TF-IDF, AVG W2V, TF-IDF WEIGHTED W
2V ENCODING OF PROJECT TITLE FEATURE ")
plt.show()
```

Summary : This visualisation of TSNE with Bag of Words, TF-IDF, Avg Word2Vec, TF-IDF Weighted Word2Vec does not seem to yield the expected result of clustering similar data points. Hence we would have to try any other method.

CONCLUSION

1. Delaware (DE) state from the United States has the highest percent of projects accepted within the whole country having almost 90% acceptance rate, followed by North Dakota (ND) and Washington (WA) nearly 89% and 88% respectively each.
2. Vermont (VT) has the lowest Approval rate with exactly 80% followed by District of Columbia (DC) and Texas (TX) with nearly 80% and 81% respectively.
3. Female Teachers have the maximum number of projects proposed and accepted compared to the male teachers.
4. There are alot of projects proposed for the students between Pre Kindergarden and 2nd Grade while for the rest it keeps decreasing as the Grades increase.
5. We also notice that Students between the 9th Grade and 12th Grade have the lowest number of projects proposed as well as accepted.
6. Projects belonging to the Literacy and Language categories have the highest number of projects proposed under. The maximum number of accepted projects also belong to this category, having an acceptance rate of nearly 87%.
7. Projects belonging to both Maths and Science have acceptance rate of nearly 82% while introducing the concept of Literacy and Language to this can increase its accpetance rate to nearly 87%
8. Projects belonging to both Maths and Science when combined with Applied Learning has the least number of projects proposed as well approved.¶
9. There is also Variability in Acceptance rate, projects under the category Warmth, Care and Hunger have an accpetance rate of 93.5%
10. The highest number of projects are registered under Literacy and Langauage with 52,239 projects, followed by Maths and Science having 41,421 projects.
11. The sub-Category Literacy has the highest number of projects approved with 8371 projects. Also the accpetance rate is 88%.
12. The sub-Category Health and Wellness have the lowest number of projects proposed with 3,583 projects only.
13. Roughly most of the projects have 3, 4 or 5 words in the title.There are hardly any project titles containing more than 10 words.
14. The number of words in the Project Essays of Approved Projects are slightly more than the number of words in the Project Essays of the Rejected Projects.
15. The Maximum price for any project should be less than 10,000 dollars.The approved projects tend to have lower cost when compared to the projects that have not been approved.
16. We observe that it is not mandatory for a teacher to have proposed any project prior. Maximum number of teachers, nearly 82% of the approved projects have been submitted by teachers with no prior project proposals. New talent and efforts are well appreciated.
17. Very few teachers who have proposed more than 20 projects have got approval. But the rate of approval is Higher given the teacher has proposed atleast 19 different projects.
18. The project summaries containing numeric values have a very high acceptance rate of 90%. Well, proper numbered requirements suggest clarity in the proposals and hence Alot of people tend to donate for a better cause, that is to help children.
19. We observe that on an average Each project costs nearly 298 Dollars. The Price paid is generally for the purchase of the Items. The projects on an average require atleast 17 Different of similar items.
20. Visualisation of TSNE with Bag of Words, TF-IDF, Avg Word2Vec, TF-IDF Weighted Word2Vec does not seem to yield the expected result of clustering similar data points. Hence we would have to try any other method.