1) Observations about Plots

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

1.1) Reading the Data

In [2]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' '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']
```

In [4]:

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

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

Out[4]:

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

In [5]:

```
# Let's print the first two rows of the project data
project_data.head(2)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

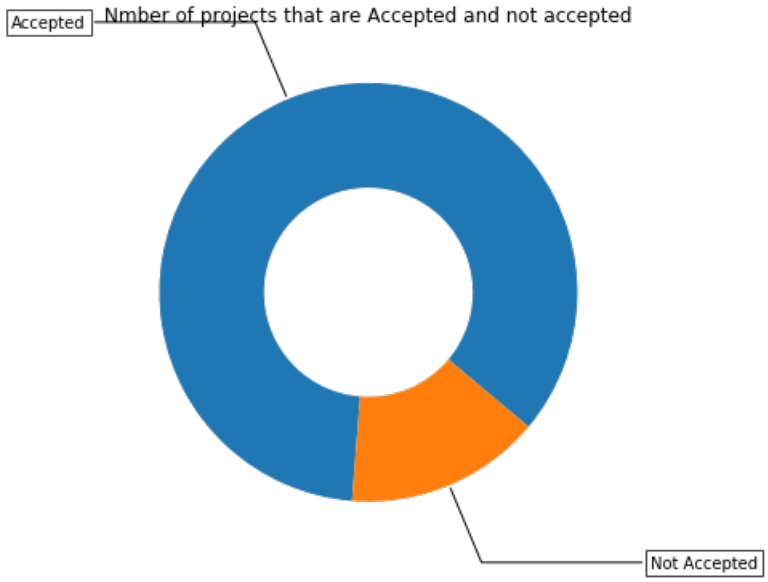
1.2) Data Analysis

In [6]:

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


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

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

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

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

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

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
```

```
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                horizontalalignment=horizontalalignment, **kw)

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

plt.show()
```

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



Observations: 1) FRom the above pie chart, and calculations, we have majority of projects that has been approved for funding and that are 84.85% 2) Projects that are not approved for funding are 15.14% of the total projects

In [7]:

```
# Let's check for any "Nan" values in our preject_data dataframe
project_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109248 entries, 0 to 109247
Data columns (total 17 columns):
Unnamed: 0                                      109248 non-null int64
id                                              109248 non-null object
teacher_id                                      109248 non-null object
teacher_prefix                                  109245 non-null object
school_state                                    109248 non-null object
project_submitted_datetime                      109248 non-null object
project_grade_category                          109248 non-null object
project_subject_categories                      109248 non-null object
project_subject_subcategories                   109248 non-null object
project_title                                   109248 non-null object
project_essay_1                                 109248 non-null object
project_essay_2                                 109248 non-null object
project_essay_3                                 3758 non-null object
project_essay_4                                 3758 non-null object
project_resource_summary                        109248 non-null object
teacher_number_of_previously_posted_projects    109248 non-null int64
project_is_approved                             109248 non-null int64
dtypes: int64(3), object(14)
memory usage: 14.2+ MB
```

In [8]:

```
# From the above data, "teacher_prefix" has "3" "NaN" or Missing values
# Let's replace the "missing values" with the most occuring values in teacher_prefix i.e., mode of
the teacher_prefix
project_data['teacher_prefix'].mode()
```

Out[8]:

```
0    Mrs.
dtype: object
```

In [9]:

```
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('Mrs.')
```

In [10]:

```
project_data['teacher_prefix'].isna().sum()
```

Out[10]:

```
0
```

1.2.1) Univariate analysis: School state

In [11]:

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

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

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

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)']]

data = [ dict(
        type='choropleth',
        colorscale = scl,
        autocolorscale = False,
        locations = temp['state_code'],
        z = temp['num_proposals'].astype(float),
        locationmode = 'USA-states',
        text = temp['state_code'],
        marker = dict(line = dict (color = 'rgb(255,255,255)',width = 2)),
        colorbar = dict(title = "% of pro")
    ) ]

layout = dict(
        title = 'Project Proposals % of Acceptance Rate by US States',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)',
        ),
    )

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='us-map-heat-map')
'''
```

Out[11]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n\nscl = [[0.0, \'rg
b(242,240,247)\'],[0.2, \'rgb(218,218,235)\'],[0.4, \'rgb(188,189,220)\'],            [0.6, \'rgb(1
58,154,200)\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)\']]\n\ndata = [ dict(\n        ty
pe=\'choropleth\',\n        colorscale = scl,\n        autocolorscale = False,\n        locations =
temp[\'state_code\'],\n        z = temp[\'num_proposals\'].astype(float),\n        locationmode = \
'USA-states\',\n        text = temp[\'state_code\'],\n        marker = dict(line = dict (color = \'
rgb(255,255,255)\',width = 2)),\n        colorbar = dict(title = "% of pro")\n    ) ]\n\nlayout = d
ict(\n        title = \'Project Proposals % of Acceptance Rate by US States\',\n        geo = dict(
\n            scope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n            show
akes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n        ),\n    )\n\nfig =
go.Figure(data=data, layout=layout)\noffline.iplot(fig, filename=\'us-map-heat-map\')\n'
```

In [12]:

```
# 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
```

In [13]:

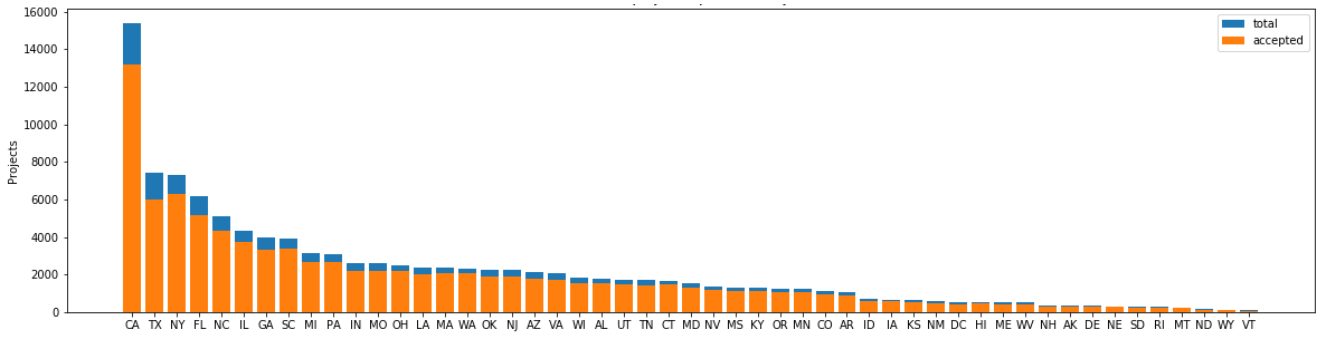
```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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 [14]:

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

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

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

    if top:
        temp = temp[0:top]

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

In [15]:

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



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

==========

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

Observation: 1) From the above graph, we can analyse that average number of projects approved with respect to the total project submitted by each state 2) We can see that California (CA) has submitted highest number of projects and also the state that has highest number of projects approved 3) Highest project submitted = 15388 by CA 4) Lowest project submiteed = 80 by VT

In [16]:

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

In [17]:

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

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

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

In [18]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['teacher_prefix'].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
[00:03<00:00, 29292.04it/s]
```

In [21]:

```
def remove_punct(prefix):
    if prefix.endswith('.'):
        return prefix[:-1]
    return prefix
```

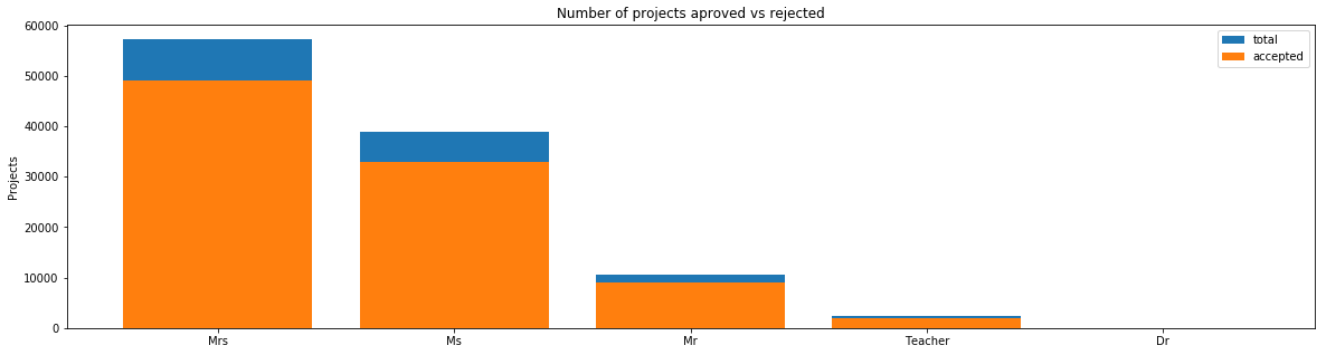
In [22]:

```
project_data['clean_prefix'] = project_data['teacher_prefix'].apply(remove_punct)
project_data.drop('teacher_prefix',axis = 1, inplace = True)
```

1.2.2) Univariate Analysis: teacher_prefix

In [24]:

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



	clean_prefix	project_is_approved	total	Avg
2	Mrs	49000	57272	0.855566
3	Ms	32860	38955	0.843537
1	Mr	8960	10648	0.841473
4	Teacher	1877	2360	0.795339
0	Dr	9	13	0.692308

==========

```
--------------------------------------------------
  clean_prefix  project_is_approved  total       Avg
2          Mrs                49000  57272  0.855566
3           Ms                32860  38955  0.843537
1           Mr                 8960  10648  0.841473
4      Teacher                 1877   2360  0.795339
0           Dr                    9     13  0.692308
```

Observations: 1) Highest number of projects submitted by the teachers having prefix as Mrs., Ms., followed by Mr. and Teacher, Dr., having 57269, 38955, 10648, 2360 and 13 respectively 2) Least number of projects are submitted by doctors teachers with prefix Dr. having only 13 projects only 3) Maximum projects approved here for Mrs(female teachers with married status) are 48997 and lowest number of projects approved are 9 only.

1.2.3) univariate Analysis: project_grade_category

In [25]:

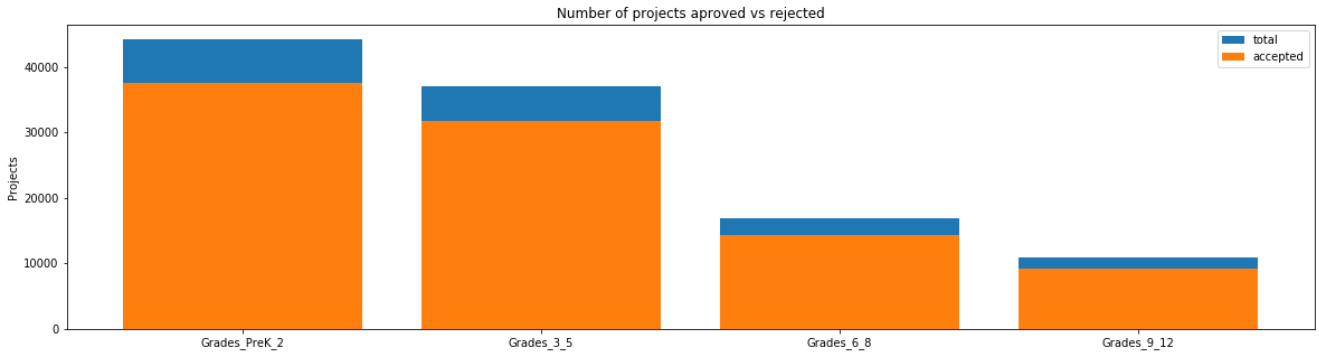
```
def clean_project_grade(list_text_feature,df,old_col_name,new_col_name):

    # remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039
    # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
    # https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
    # https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
    feature_list = []
    for i in list_text_feature:
        temp = i.split(' ')
        last_dig = temp[-1].split('-')
        fin = [temp[0]]
        fin.extend(last_dig)
        feature = '_'.join(fin)
        feature_list.append(feature.strip())

    df[new_col_name] = feature_list
    df.drop([old_col_name], axis=1, inplace=True)

    from collections import Counter
    my_counter = Counter()
    for word in df[new_col_name].values:
        my_counter.update(word.split())

    feature_dict = dict(my_counter)
    sorted_feature_dict = dict(sorted(feature_dict.items(), key=lambda kv: kv[1]))
    return sorted_feature_dict
```

In [26]:

```
grade_sorted_grade_dict = clean_project_grade(project_data['project_grade_category'],project_data,
'project_grade_category','clean_grade_category')
```

In [27]:

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




```
  clean_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
==================================================
  clean_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
```

Observations: 1) maximum projects submitted are came from Grades PreK - 2 and that is 44225 projects, out of which 37536 appeoved (84.87%) 2) Least projects came from Grades 9 - 12 and that is 10963, out of which 9183 are approved (83.76%

)

1.2.4) Univariate Analysis: project_subject_categories

In [28]:

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

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

In [29]:

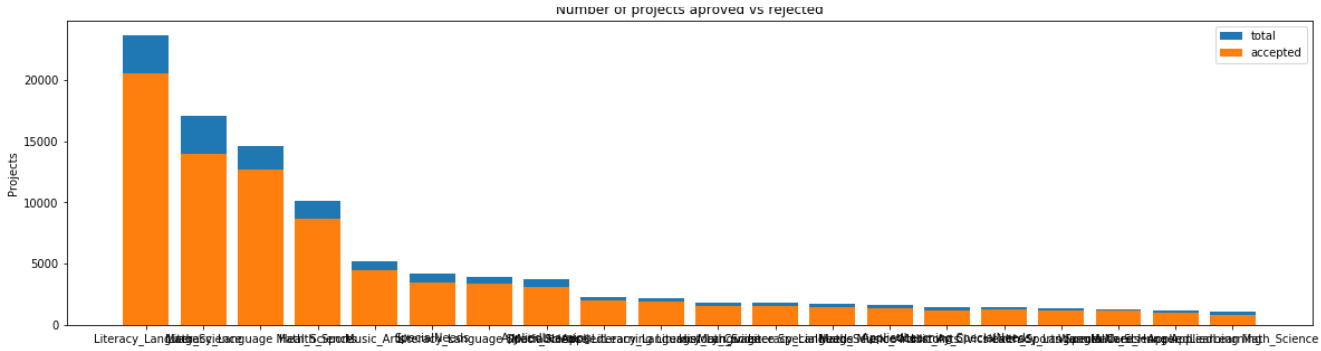
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[29]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_subject_su
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	ESL, Literacy
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Civics & Governmen Sports

In [30]:

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



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

==========

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

Observations: 1) Maximum projects submitted comes from the category of Literacy_Language, 23655 projects and total acceptance rate is 86.74 % accounting about 20520 projects accepted. 2) Least project submitted from AppliedLearning Math_Science categories with only 1052 submission, from that 81.27% projects accepted, around 855.

In [31]:

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

In [32]:

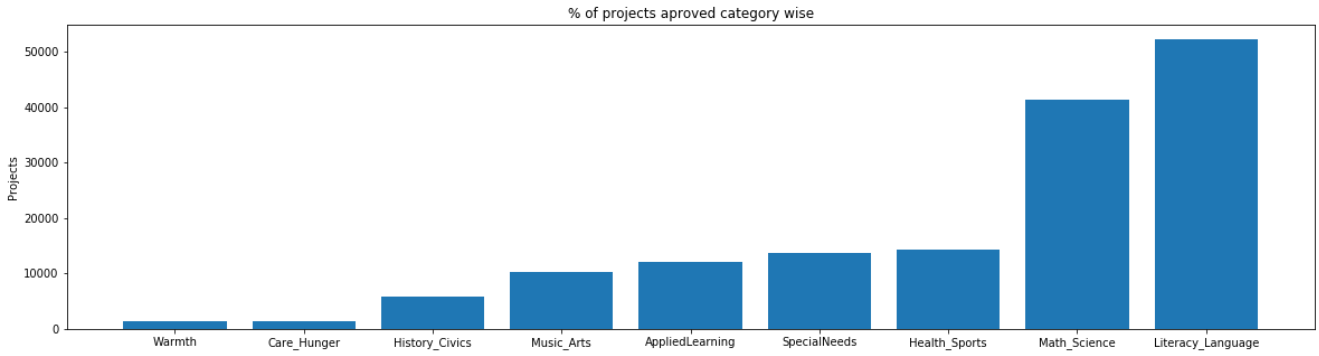
```
for word in project_data['clean_categories'].values[:50]:
    print(word.split())
```

```
['Literacy_Language']
['History_Civics', 'Health_Sports']
['Health_Sports']
['Literacy_Language', 'Math_Science']
['Math_Science']
['Literacy_Language', 'SpecialNeeds']
['Literacy_Language', 'SpecialNeeds']
['Math_Science']
['Health_Sports']
['Literacy_Language']
['Literacy_Language']
['Literacy_Language', 'AppliedLearning']
['Math_Science']
['SpecialNeeds']
['Literacy_Language']
['Health_Sports']
['Literacy_Language', 'SpecialNeeds']
['Math_Science', 'Literacy_Language']
['AppliedLearning']
['Health_Sports']
['Literacy_Language']
['Math_Science', 'SpecialNeeds']
['Literacy_Language']
['Music_Arts']
['Math_Science']
['Math_Science']
['Literacy_Language', 'Math_Science']
['Literacy_Language', 'Math_Science']
```

```
['Literacy_Language', 'SpecialNeeds']
['Literacy_Language', 'AppliedLearning']
['Literacy_Language']
['SpecialNeeds']
['Math_Science', 'Literacy_Language']
['History_Civics']
['Literacy_Language']
['Health_Sports']
['Literacy_Language', 'Math_Science']
['Health_Sports', 'Literacy_Language']
['Health_Sports']
['Literacy_Language']
['Literacy_Language']
['Literacy_Language']
['Literacy_Language']
['Literacy_Language', 'Music_Arts']
['Math_Science']
['Literacy_Language']
['Literacy_Language']
['Warmth', 'Care_Hunger']
['Literacy_Language', 'Math_Science']
['Health_Sports']
```

In [33]:

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


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

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



Observtions: 1) Above graph shows the % of projects approved from different categories 2) Maximum projects are accepted from Literacy_Language category followed by Math_Science and so on. 3) From Warmth category, Least number of projects ahs been approved, followed Care_Hunger and so on

In [34]:

```
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
```

1.2.5) Univariate Analysis: project_subject_subcategories

In [35]:

```
# Let's clean the sub_categories first
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

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

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

In [36]:

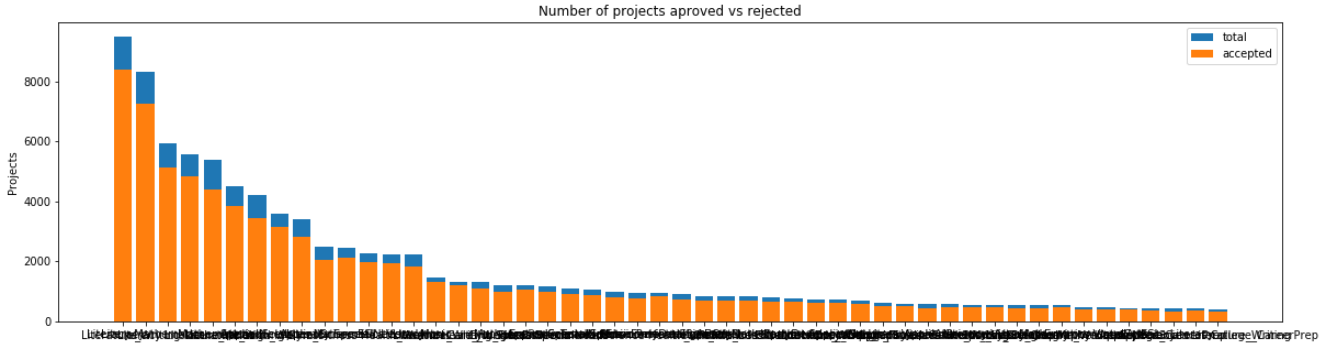
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[36]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stu Englisl that ar
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	Our st arrive school lea...

In [37]:

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



clean_subcategories project_is_approved total Avg

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

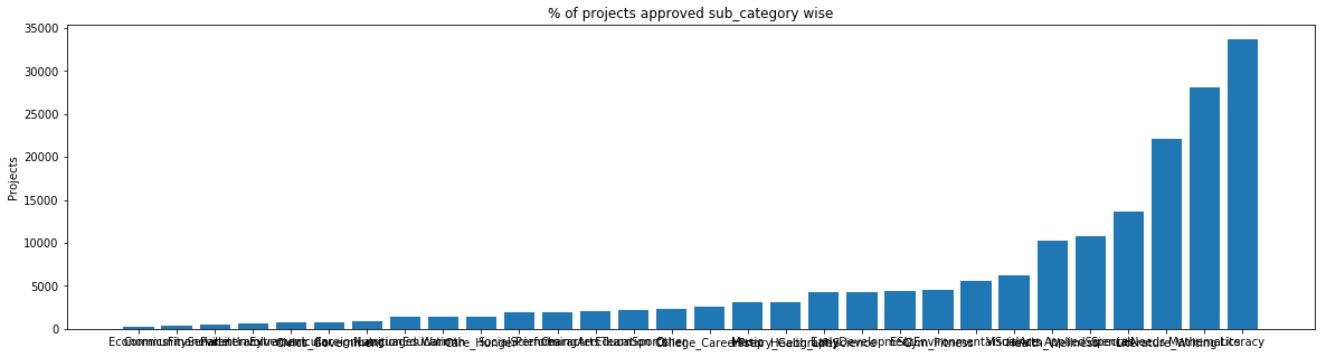
Obervations: 1) From sub_categories, maximum projects comes from Literacy with total of 9486 projects out of which 8371 projects are approved. 2) Least projects has came from AppliedScience College_CareerPre subcategory that has a total count of projects only 405 and 330 approved.

In [38]:

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

In [39]:

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


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

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



Observations: 1) From the graph it is clear that maximum projects approved comes from Literacy sub_catrgories and least comes from economics and Community_Service

In [40]:

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

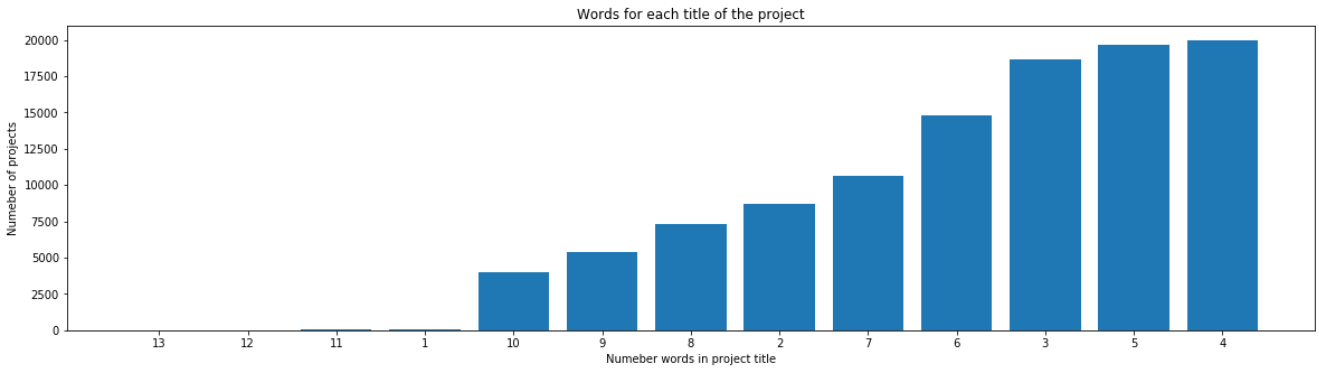
```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
```

```
NutritionEducation   :      1355
Warmth               :      1388
Care_Hunger          :      1388
SocialSciences       :      1920
PerformingArts       :      1961
CharacterEducation   :      2065
TeamSports           :      2192
Other                :      2372
College_CareerPrep   :      2568
Music                :      3145
History_Geography    :      3171
Health_LifeScience   :      4235
EarlyDevelopment     :      4254
ESL                  :      4367
Gym_Fitness          :      4509
EnvironmentalScience :      5591
VisualArts           :      6278
Health_Wellness      :     10234
AppliedSciences      :     10816
SpecialNeeds         :     13642
Literature_Writing   :     22179
Mathematics          :     28074
Literacy             :     33700
```

1.2.6) Univariate Analysis: Text Features(Title)

In [41]:

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


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

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



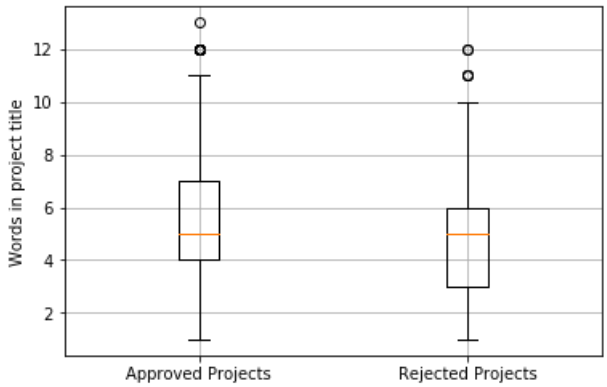
From the graph we can inferred that, Maximum number of projects are having only 4 words in their titles and least number of them are lengthy with 13 words.

In [42]:

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

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

In [43]:

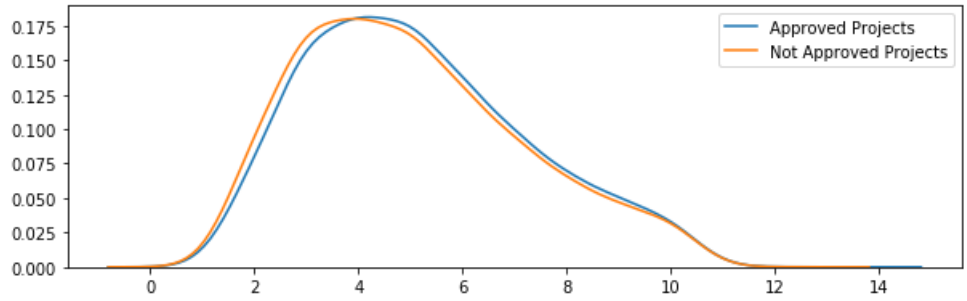
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



Observation:From the above graphs we can get comparison between the number of words in projects that are approved V/s projects that are not approved. As we can infered that, the approved projects comes with somewhat more number of words in their titles (about 4 to 7) and (3 to 6) in non approved projects. But the means of them looks like the same. We are not sure about the effect of the number of words in the acceptance rate of the project, But the slight effect might be, if more the descriptive project title of the project there might be higher chances of project acceptance.

In [44]:

```
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()
```



Observtion: From the kde plot above, we can assert that there is not much different between the approved and Not approved projects, as the distribution is mostly similar with slight advance for approved project for longer titles. Hence the number of words in ptoject title does not affect on the acceptance rate of the project.

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

In [45]:

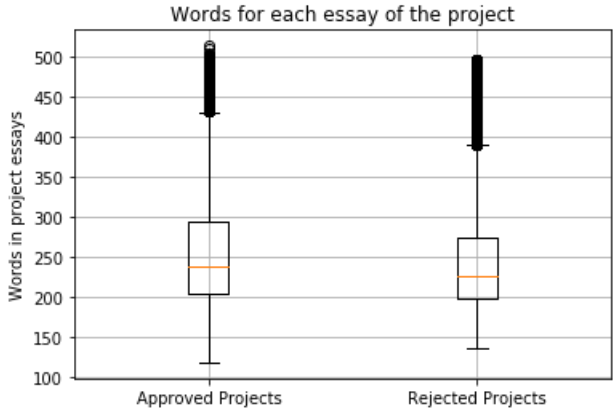
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [46]:

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

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

In [47]:

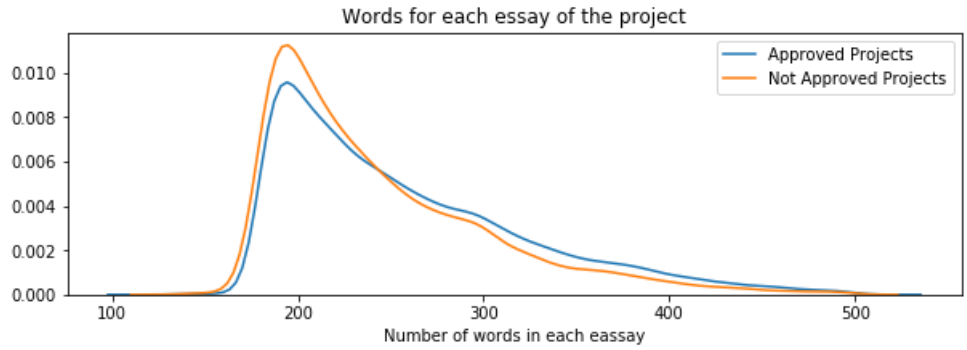
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()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



Observation: From the box plot above, we can infered that, the effect of number of words of project essay's on the project acceptance rate is minimum. As the approved projects and not approved projects seems to overlaped completely with mean words in approved projects is more than the not approved projects, around 240 and 225 respectively.

In [48]:

```
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()
```



It is clear from the kde plot above, that there is similar distribution of approved and not approved projects in terms of number of words in project essay's. So the number of words in the essay alon can not give us much sense about the acceptance rate of the project.

1.2.8) Univariate Analysis: Cost per project

In [49]:

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

Out[49]:

	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 [50]:

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

Out[50]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [51]:

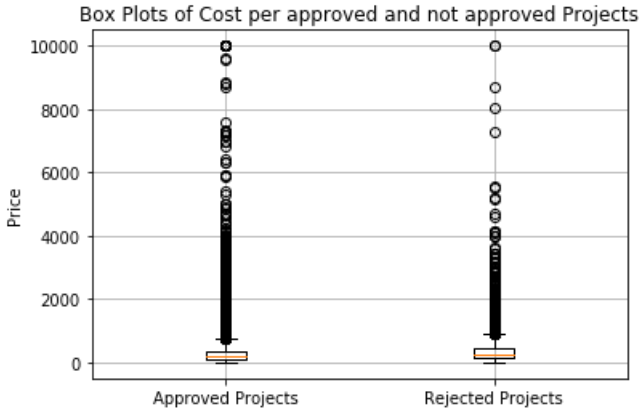
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [52]:

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

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

In [53]:

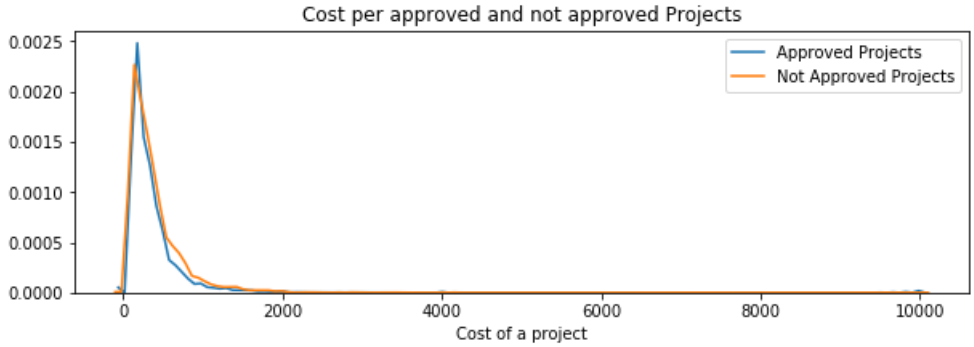
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Price')
plt.grid()
plt.show()
```



Observation: From the above box plot, the cost per project for both approved and not approved projects is nearly equal(little more for not approved projects) but it seems that the mean cost per project is also same. The number of outliers in approved projects are more than not approved projects.

In [54]:

```
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()
```



It is more clear from the above kde plot, that we have nearly similar distribution for cost per project for both categories and acceptance rate is nearly independent of cost per project.

In [55]:

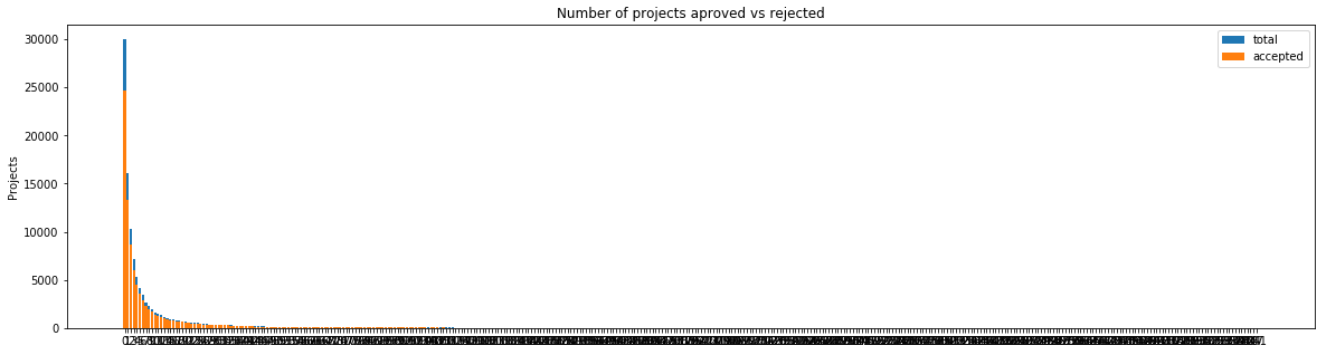
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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

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

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

1.2.9) Univariate Analysis: teacher_number_of_previously_posted_projects

In [56]:

```
temp = pd.DataFrame(project_data.groupby('teacher_number_of_previously_posted_projects')
["project_is_approved"].apply(np.mean)).reset_index()
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects',
'project_is_approved', False)
```

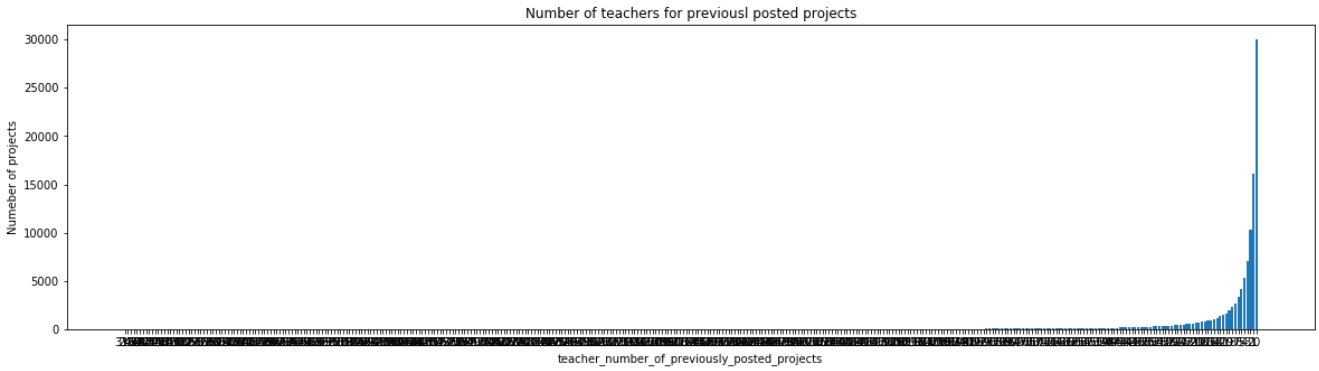



```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                24652  30014
1                                             1                13329  16058
2                                             2                 8705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

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

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

Observation: 1) It is clear from the figure that, the more emphasis is given for new teachers that have submitted the projects for the firs time, as we can see most of the teachers are new and they haven't submitted the projects previously. 2) Total 30014 teachers have submitted the projects for the first time out of which 24652 (nearly 82%) are accepted, followed by teacher which have submitted the project once in the past and so on. 3) If the teacher has appiled minimum number of times previously, its chances of project acceptance is more. 4) As the number of previously applied teachers becomes more, there chances of acceptance is drastically less.

In [57]:

```
teacher_count = project_data['teacher_number_of_previously_posted_projects'].value_counts()
teacher_dict = dict(teacher_count)
teacher_dict = dict(sorted(teacher_dict.items(), key=lambda kv: kv[1]))


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

plt.ylabel('Numeber of projects')
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.title('Number of teachers for previousl posted projects')
plt.xticks(ind, list(teacher_dict.keys()))
plt.show()
```



From the graph above, it is clear that the more the number teacher of previously posted projects increases there is a drastic decline in the acceptance rate of the project.

1.2.10) Univariate Analysis: project_resource_summary

In [58]:

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

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

In [59]:

```
project_data['clean_project_resource_summary'] = res_list
project_data.drop(['project_resource_summary'], axis=1, inplace=True)
project_data.head(2)
```

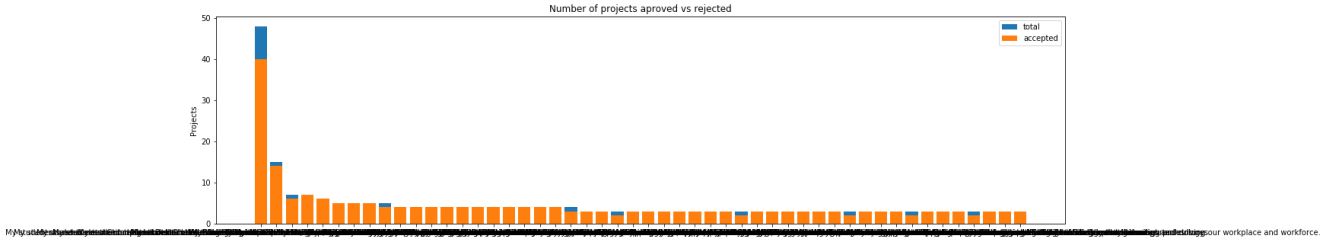
Out[59]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stu Englis that ar
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry	Our st arrive school

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	Learners project_title	lea... projec

In [60]:

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




```
                        clean_project_resource_summary  project_is_approved  \
56516  My students need electronic tablets to do all ...                   40
10289  My students need Chromebooks to do all the thi...                   14
51409  My students need chromebooks to do all the thi...                    6
18800  My students need a Dell Chromebook 3120 and a ...                    7
18791  My students need a Dell Chromebook 3120 11 6 C...                    6

       total       Avg
56516     48  0.833333
10289     15  0.933333
51409      7  0.857143
18800      7  1.000000
18791      6  1.000000
==================================================
                        clean_project_resource_summary  project_is_approved  \
43373  My students need another HP Chromebook and a G...                    3
10626  My students need Chromebooks to prepare and en...                    2
39381  My students need an Amazon Echo with remote to...                    3
7872   My students need 7 Google Chromebooks and 7 Go...                    3
66887  My students need iPad minis and iPad mini case...                    3

       total       Avg
43373      3  1.000000
10626      3  0.666667
39381      3  1.000000
7872       3  1.000000
66887      3  1.000000
```

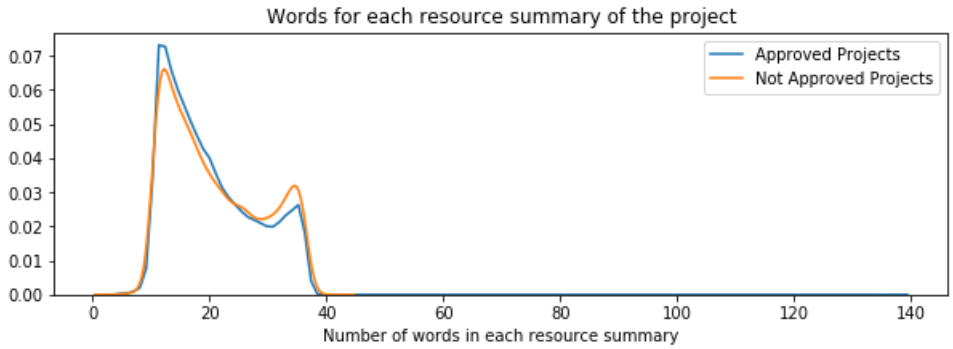
Observation: From the resource summary, it is clear that the more economical is the project, the more is the acceptance rate. As we can see the largest approved project resource summary(40 accepted out of 48) requires only a simple electronics tablets for all the there requirements as compared to other resources which requires more costly requirements as chromebook, amazon echo dot and so on.

In [61]:

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

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

In [62]:

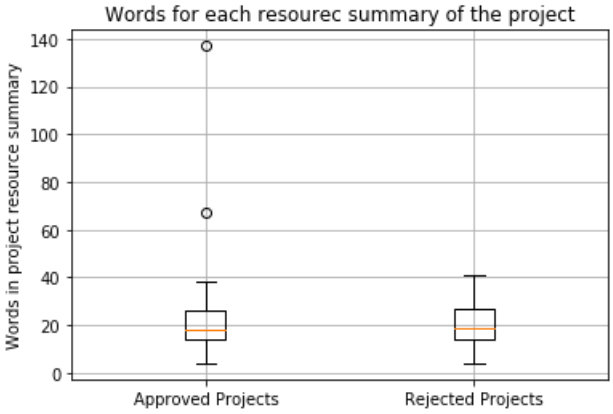
```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each resource summary of the project')
plt.xlabel('Number of words in each resource summary')
plt.legend()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



As evident from the graph above, the number of words in the resource summary does not affect on the acceptance rate of the project.

In [63]:

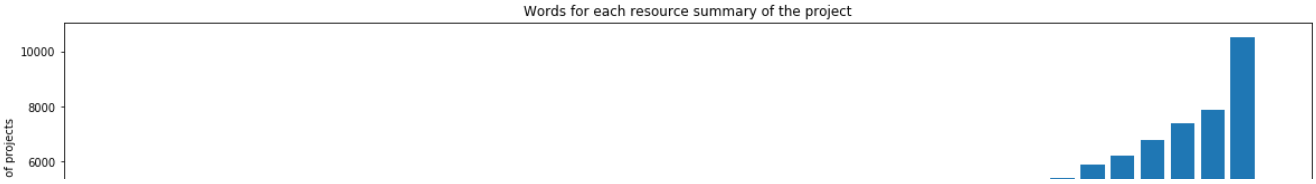
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each resourec summary of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project resource summary')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

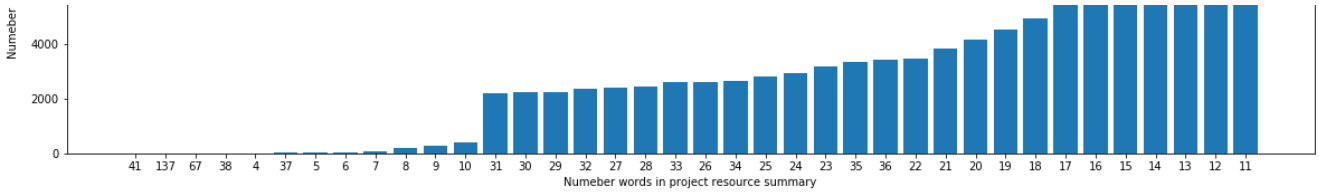


Again, from the above box plot, it is clear that the number of words in the resource summary does not affect on acceptance rate. Because all of them have nearly same number of words distribution in the resource summaries.

In [64]:

```
#How to calculate number of words in a string in DataFrame:
https://stackoverflow.com/a/37483537/4084039
word_count = project_data['clean_project_resource_summary'].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 resource summary')
plt.title('Words for each resource summary of the project')
plt.xticks(ind, list(word_dict.keys()))
plt.show()
```





In [65]:

```
# check for the effect of the numerical digit in the resourec_project_summary on the acceptance of
the project
with_digit_approved = []
without_digit_approved = []
with_digit_not_approved = []
without_digit_not_approved = []
for index in range(109248):
    flag = False
    for word in project_data['clean_project_resource_summary'].loc[index].split():
        if word.isdigit():
            if project_data['project_is_approved'].loc[index] == 1:
                with_digit_approved.append(project_data['clean_project_resource_summary'].loc[index
])
            else:
                with_digit_not_approved.append(project_data['clean_project_resource_summary'].loc[i
ndex])
            flag = True
            break
    if not flag:
        if project_data['project_is_approved'].loc[index] == 1:
                without_digit_approved.append(project_data['clean_project_resource_summary'].loc[in
dex])
        else:
                without_digit_not_approved.append(project_data['clean_project_resource_summary'].lo
c[index])
```

In [66]:

```
total_projects = len(with_digit_approved) + len(without_digit_approved) +
len(with_digit_not_approved) + len(without_digit_not_approved)
print("Total percentage of projects with resource having digits approved:",len(with_digit_approved
) / total_projects * 100,"%")
print("Total percentage of projects with resource having digits not
approved:",len(with_digit_not_approved) / total_projects * 100,"%")
print("Total percentage of projects with resource not having digits
approved:",len(without_digit_approved) / total_projects * 100,"%")
print("Total percentage of projects with resource not having digits not
approved:",len(without_digit_not_approved) / total_projects * 100,"%")
```

```
Total percentage of projects with resource having digits approved: 9.378661394258934 %
Total percentage of projects with resource having digits not approved: 1.0142062097246631 %
Total percentage of projects with resource not having digits approved: 75.47964264792033 %
Total percentage of projects with resource not having digits not approved: 14.127489748096075 %
```

In [67]:

```
print("Total projects summaries with digits:", (len(with_digit_approved) +
len(with_digit_not_approved)) / total_projects * 100,"%")
print("Total projects summaries without digits:", (len(without_digit_approved) +
len(without_digit_not_approved)) / total_projects * 100,"%")
```

```
Total projects summaries with digits: 10.392867603983596 %
Total projects summaries without digits: 89.6071323960164 %
```

Observation: From the above analysis, total number of project summaries that contains digits are about 10.4% approximately, out of which about 90% of the summary projects has been approved and only 10% has been rejected. So we can infered that, having digits inside the project summaries help to assess the impact of the project being submitted and plays a significant part in project acceptance rate.

1.3) Text Preprocessing

1.3.1) Essay Text

In [68]:

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

Out[68]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stu Englisl that ar
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	Our st arrive school lea...

In [69]:

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

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
```

chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade. This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of
my own pocket on resources to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir 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 pr
ice 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 gr
oove 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 dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids don't want to sit and do worksheets. They want to learn to count by ju
mping and playing. Physical engagement is the key to our success. The number toss and color and sh
ape 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 grea
t teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% Af
rican-American, making up the largest segment of the student body. A typical school in Dallas is m
ade up of 23.2% African-American students. Most of the students are on free or reduced lunch. We a
ren'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 smar
t, effective, efficient, and disciplined students with good character.In our classroom we can util
ize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the so
und enough to receive the message. Due to the volume of my speaker my students can't hear videos o
r books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my s
tudents will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will all
ow 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 [70]:

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

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

```
    phrase = re.sub(r"can\'t", "can not", phrase)

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

In [71]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir 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 pr
ice 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 gr
oove 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 dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids do not want to sit and do worksheets. They want to learn to count by j
umping and playing. Physical engagement is the key to our success. The number toss and color and s
hape 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 [72]:

```
# \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, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir 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 reduced price
lunch.  Despite their disabilities and limitations, my students love coming to school and come eag
er 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 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 develo
p their core, which enhances gross motor and in Turn fine motor skills.   They also want to learn t
hrough games, my kids do not want to sit and do worksheets. They want to learn to count by jumping
and playing. Physical engagement is the key to our success. The number toss and color and shape ma
ts can make that happen. My students will forget they are doing work and just have the fun a 6 yea
r old deserves.nannan
```

In [73]:

```
#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 cognitiv
e delays gross fine motor delays to autism They are 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 studen
ts 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 lea
rn and explore Have you ever felt like you had ants in your pants and you needed to groove and mov
e as you were in a meeting This is how my kids feel all the time The want to be able to move as th
ey learn or so they say Wobble chairs are the answer and I love then because they develop their co
re which enhances gross motor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Ph
ysical engagement is the key to our success The number toss and color and shape mats can make that
```

ysical 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 nan nan

In [74]:

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

```
100%|████████████████████████████████████████████████████████████████| 109248/109248
[01:47<00:00, 1014.25it/s]
```

In [75]:

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

Out[75]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gros
s fine motor delays autism they eager beavers always strive work hardest working past limitations
the materials ones i seek students i teach title i school students receive free reduced price lunc
h despite disabilities limitations students love coming school come eager learn explore have ever
felt like ants pants needed groove move meeting this kids feel time the want able move learn say w
obble chairs answer i love develop core enhances gross motor turn fine motor skills they also want
learn games kids not want sit worksheets they want learn count jumping playing physical engagement
key success the number toss color shape mats make happen my students forget work fun 6 year old de
serves nannan'
```

1.3.2) Project title Text

In [76]:

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

```
100%|████████████████████████████████████████████████████████████████| 109248/109248
[00:05<00:00, 18579.93it/s]
```

In [77]:

```
project_data['clean_project_title'] = preprocessed_titles
project_data.drop(['project_title'], axis=1, inplace=True)
project_data.head(2)
```

Out[77]:

	Unnamed:					

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_essay_1	p
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	My students are English learners that are work...	\" y a
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:22:10	Our students arrive to our school eager to lea...	T n s

1.4) Preparing Data for Models

In [78]:

```
project_data.columns
```

Out[78]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_prefix', 'clean_grade_category', 'clean_categories',
       'clean_subcategories', 'essay', 'price', 'quantity',
       'clean_project_resource_summary', 'clean_project_title'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data
- quantity : numerical
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

1.4.1) Vectorizing Categorical Data

In [79]:

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


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
```

```
Shape of matrix after one hot encodig  (109248, 9)
```

In [80]:

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


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

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

In [81]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category als
o
# using count vectorizer for one-hot encoding of state
vectorizer = CountVectorizer(vocabulary=set(project_data['school_state'].values), lowercase=False,
binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


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

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

In [82]:

```
# using count vectorizer for one-hot encoding of teacher_prefix
# vectorizer = CountVectorizer(vocabulary=set(project_data['teacher_prefix'].values),
lowercase=False, binary=True)
# vectorizer.fit(project_data['teacher_prefix'].values)
# print(vectorizer.get_feature_names())


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

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
vectorizer.fit(project_data['clean_prefix'].values)
feature_one_hot = vectorizer.transform(project_data['clean_prefix'].values)
feature_one_hot
```

Out[82]:

```
<109248x5 sparse matrix of type '<class 'numpy.int64'>'
 with 109248 stored elements in Compressed Sparse Row format>
```

In [83]:

```
# using count vectorizer for one-hot encoding of project_grade_category
vectorizer = CountVectorizer(vocabulary=set(project_data['clean_grade_category'].values),
lowercase=False, binary=True)
vectorizer.fit(project_data['clean_grade_category'].values)
```

```
vectorizer.fit(project_data['clean_grade_category'].values)
print(vectorizer.get_feature_names())


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

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of matrix after one hot encodig  (109248, 4)
```

In [84]:

```
# using count vectorizer for one-hot encoding of teacher_prefix as clean_prefix
vectorizer = CountVectorizer(vocabulary=set(project_data['clean_prefix'].values), lowercase=False,
binary=True)
vectorizer.fit(project_data['clean_prefix'].values)
print(vectorizer.get_feature_names())


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

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (109248, 4)
```

1.4.2) Vectorizing Text Data

1.4.2.1) Bag of Words

In [85]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

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

1.4.2.2) Bag of words on project title

In [86]:

```
# Lets vectorize project_titles
vectorizer = CountVectorizer(min_df=10)
title_text_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",title_text_bow.shape)
```

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

1.4.2.3) TFIDF vectorizer

In [87]:

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

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

1.4.2.4) TFIDF vectorizer on project title

In [88]:

In [88]:

```
# TFIDF vectorization for project_title
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
title_text_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",title_text_tfidf.shape)
```

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

1.4.2.5) Using Pretrained Models: Avg W2V

In [89]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[89]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef
loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\',
encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n
```

```
word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\'
\'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the
coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus",
len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words tha
t are present in both glove vectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\r
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [90]:

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

In [91]:

```
# 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
[01:23<00:00, 1313.97it/s]
```

```
109248
300
```

1.4.2.6) Using Pretrained Models: Avg W2V on project title

In [92]:

```
# Similarly you can vectorize for title also
# average Word2Vec
# compute average word2vec for each review/ project_titles
titles_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    titles_avg_w2v_vectors.append(vector)

print(len(titles_avg_w2v_vectors))
print(len(titles_avg_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████| 109248/109248
[00:03<00:00, 27852.28it/s]
```

```
109248
300
```

1.4.2.7) Using Pretrained Models: TFIDF weighted W2V

In [93]:

```
# 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 [94]:

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

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

```
100%|████████████████████████████████████████████████████████████| 109248/109248
[08:41<00:00, 209.49it/s]
```

```
109248
300
```

1.4.2.9) Using Pretrained Models: TFIDF weighted W2V on project title

In [95]:

```
# Similarly you can vectorize for title also
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
# we are converting a dictionary with word as a key, and the idf as a value
titles_dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
titles_tfidf_words = set(tfidf_model.get_feature_names())
```

In [96]:

```
# average Word2Vec
# compute average word2vec for each review.
titles_tfidf_w2v_vectors = []; # 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 titles_tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = titles_dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    titles_tfidf_w2v_vectors.append(vector)

print(len(titles_tfidf_w2v_vectors))
print(len(titles_tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:09<00:00, 12115.59it/s]
```

```
109248
300
```

1.4.3) Vectorizing Numerical Data

In [97]:

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

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

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

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

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

In [98]:

```
price_standardized
```

Out[98]:

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

In [99]:

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

```
# Now standardize the data with above maen and variance.
teacher_number_standardized =
teacher_number_scalar.transform(project_data['teacher_number_of_previously_posted_projects'].value
s.reshape(-1, 1))
```

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

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

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

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

In [100]:

```
teacher_number_standardized
```

Out[100]:

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

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

3) Building Data Matrix using all Features

Let's describe the number of features we have

In [101]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(school_state_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(project_category_one_hot.shape)
print(price_standardized.shape)
print(teacher_number_standardized.shape)
print(text_bow.shape)
print(title_text_bow.shape)
print(text_tfidf.shape)
print(title_text_tfidf.shape)
print(np.asarray(avg_w2v_vectors).shape)
print(np.asarray(titles_avg_w2v_vectors).shape)
print(np.asarray(tfidf_w2v_vectors).shape)
print(np.asarray(titles_tfidf_w2v_vectors).shape)
```

```
(109248, 9)
(109248, 30)
(109248, 51)
(109248, 5)
(109248, 4)
(109248, 1)
(109248, 1)
(109248, 16623)
(109248, 3329)
(109248, 16623)
(109248, 3329)
(109248, 300)
(109248, 300)
(109248, 300)
(109248, 300)
```

In [103]:

```
from scipy.sparse import hstack
X_whole = hstack((categories_one_hot,
sub_categories_one_hot,school_state_one_hot,teacher_prefix_one_hot,project_category_one_hot, price
_standardized,teacher_number_standardized,text_bow,\
                  title_text_bow,text_tfidf, title_text_tfidf,
avg_w2v_vectors,titles_avg_w2v_vectors,tfidf_w2v_vectors,titles_tfidf_w2v_vectors))
X_whole.shape
```

Out[103]:

```
(109248, 41205)
```

4) T-SNE Plots

4.1) T-SNE plot for

- Categorical, Numerical feutures + Project_title(BOW)

In [104]:

```
# Let's select top 5000 data points for t-sne plot
sample_categories = categories_one_hot[:5000]
sample_subcategories = sub_categories_one_hot[:5000]
sample_school_state = school_state_one_hot[:5000]
sample_teacher_prefix = teacher_prefix_one_hot[:5000]
sample_project_cat = project_category_one_hot[:5000]
sample_price = price_standardized[:5000]
sample_teacher_number = teacher_number_standardized[:5000]
sample_text_bow = text_bow[:5000]
sample_title_text_bow = title_text_bow[:5000]
sample_text_tfidf = text_tfidf[:5000]
sample_title_text_tfidf = title_text_tfidf[:5000]
sample_avg_w2v_vectors = np.asarray(avg_w2v_vectors)[:5000]
sample_titles_avg_w2v_vectors = np.asarray(titles_avg_w2v_vectors)[:5000]
sample_tfidf_w2v_vectors = np.asarray(tfidf_w2v_vectors)[:5000]
sample_titles_tfidf_w2v_vectors = np.asarray(titles_tfidf_w2v_vectors)[:5000]
```

TSNE with BOW encoding of project_title feature

In [105]:

```
# Let's prepare data matrix for Categorical, Numerical + project_title(BOW) data
X_Bow = hstack((sample_categories, sample_subcategories,sample_school_state,sample_teacher_prefix,
sample_project_cat, sample_price,sample_teacher_number,\
                  sample_title_text_bow))
```

In [106]:

```
Y = project_data['project_is_approved'][:5000]
np.asarray(Y).reshape(-1,1)
```

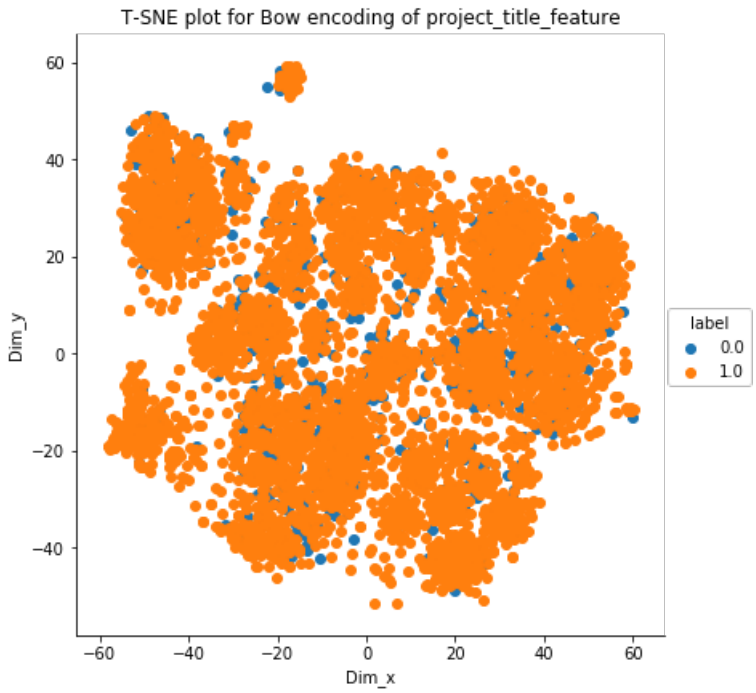
Out[106]:

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

In [107]:

```
# import t-sne
```

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, perplexity=50, random_state = 0, n_iter = 5000)

X_embedding = tsne.fit_transform(X_Bow.toarray())
Y = project_data['project_is_approved'][:5000]
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .
toarray() will convert the sparse matrix into dense matrix

for_tsne = np.hstack((X_embedding, np.asarray(Y).reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_x','Dim_y','label'])
sns.FacetGrid(for_tsne_df,hue='label',size=6).map(plt.scatter,'Dim_x','Dim_y').add_legend()
plt.title("T-SNE plot for Bow encoding of project_title_feature")
plt.show()
```



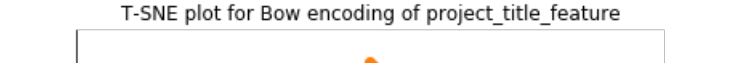
TSNE with TFIDF encoding of project_title feature

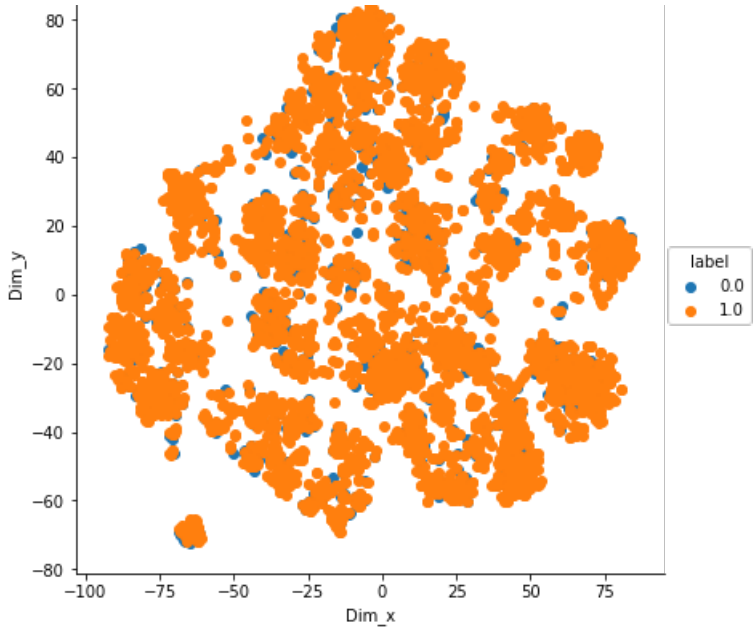
In [108]:

```
# Let's prepare data matrix for Categorical, Numerical + project_title(BOW) data
X_TFIDF = hstack((sample_categories,
sample_subcategories,sample_school_state,sample_teacher_prefix,sample_project_cat, sample_price,sa
mple_teacher_number,\
                  sample_title_text_tfidf))
```

In [109]:

```
# import t-sne
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, perplexity=50, random_state = 0, n_iter = 5000)

X_embedding = tsne.fit_transform(X_TFIDF.toarray())
Y = project_data['project_is_approved'][:5000]
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .
toarray() will convert the sparse matrix into dense matrix

for_tsne = np.hstack((X_embedding, np.asarray(Y).reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_x','Dim_y','label'])
sns.FacetGrid(for_tsne_df,hue='label',size=6).map(plt.scatter,'Dim_x','Dim_y').add_legend()
plt.title("T-SNE plot for Bow encoding of project_title_feature")
plt.show()
```





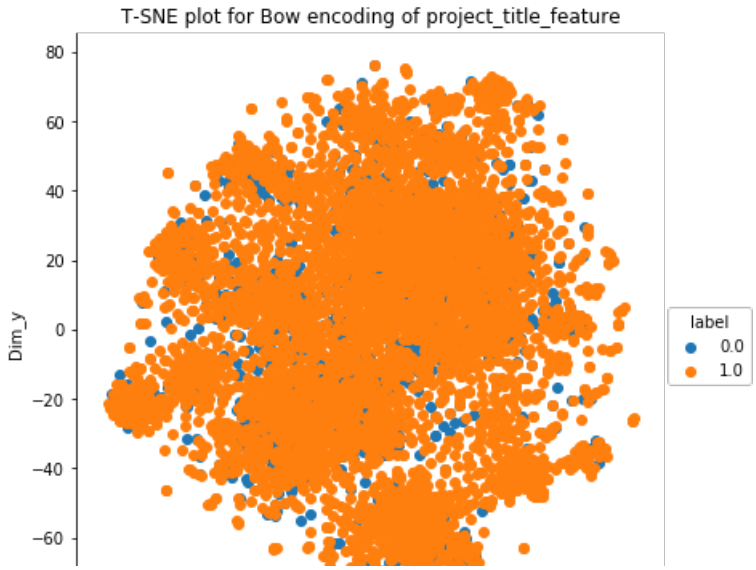
TSNE with AVG W2V encoding of project_title feature

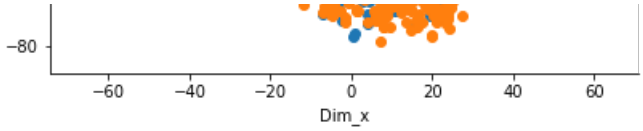
In [110]:

```
# Let's prepare data matrix for Categorical, Numerical + project_title(BOW) data
X_AVG_W2V = hstack((sample_categories,
sample_subcategories,sample_school_state,sample_teacher_prefix,sample_project_cat, sample_price,sa
mple_teacher_number,\
                 sample_titles_avg_w2v_vectors))
```

In [111]:

```
# import t-sne
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, perplexity=50, random_state = 0, n_iter = 5000)

X_embedding = tsne.fit_transform(X_AVG_W2V.toarray())
Y = project_data['project_is_approved'][:5000]
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .
toarray() will convert the sparse matrix into dense matrix

for_tsne = np.hstack((X_embedding, np.asarray(Y).reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_x','Dim_y','label'])
sns.FacetGrid(for_tsne_df,hue='label',size=6).map(plt.scatter,'Dim_x','Dim_y').add_legend()
plt.title("T-SNE plot for Bow encoding of project_title_feature")
plt.show()
```





TSNE with TFIDF Weighted W2V encoding of project_title feature

In [112]:

```
# Let's prepare data matrix for Categorical, Numerical + project_title(BOW) data
X_TFIDF_W2V = hstack((sample_categories,
sample_subcategories,sample_school_state,sample_teacher_prefix,sample_project_cat, sample_price,sa
mple_teacher_number,\
                  sample_titles_tfidf_w2v_vectors))
```

In [113]:

```
# import t-sne
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, perplexity=50, random_state = 0, n_iter = 5000)

X_embedding = tsne.fit_transform(X_TFIDF_W2V.toarray())
Y = project_data['project_is_approved'][:5000]
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .
toarray() will convert the sparse matrix into dense matrix

for_tsne = np.hstack((X_embedding, np.asarray(Y).reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_x','Dim_y','label'])
sns.FacetGrid(for_tsne_df,hue='label',size=6).map(plt.scatter,'Dim_x','Dim_y').add_legend()
plt.title("T-SNE plot for Bow encoding of project_title_feature")
plt.show()
```