DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!</code`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

```
#Importing Essential library & packages

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

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

```
#Reading frm the train csv & resources csv files
#making copies of the dataframe

project_data_60 = project_data_50 = project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:

```
#Printing shape of the data & columns present in the dataset

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]:

```
#Printing data points is train data & Column values of resource data
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

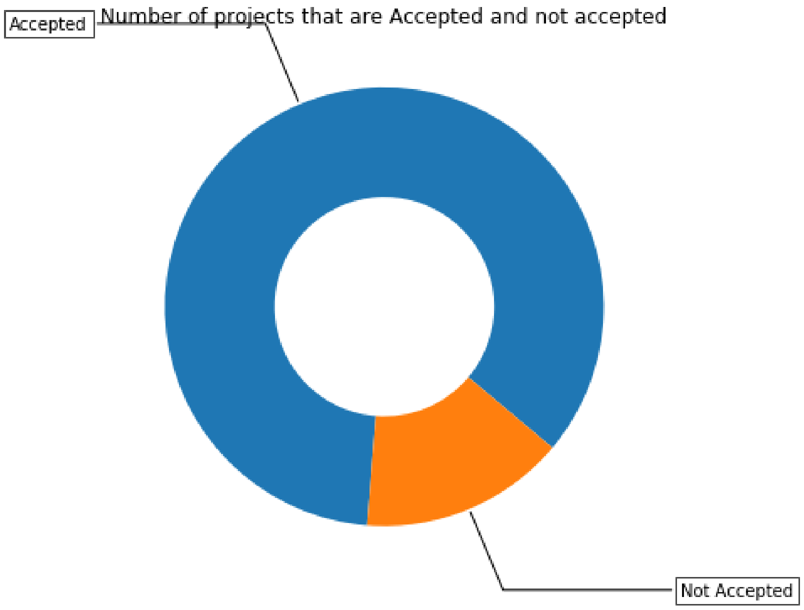
Out[4]:

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

1.2 Data Analysis

In [5]:
```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-glr-gallery-pie-and-p
#Calculating & plotting (Donut Chart) for Number of approved & Non-approved projects

y_value_counts = project_data['project_is_approved'].value_counts()
print("Number of projects than are approved for funding ", y_value_counts[1], ", (", (y_value_counts[1]/(y
print("Number of projects than are not approved for funding ", y_value_counts[0], ", (", (y_value_counts[0

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

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

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

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

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

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

plt.show()
```

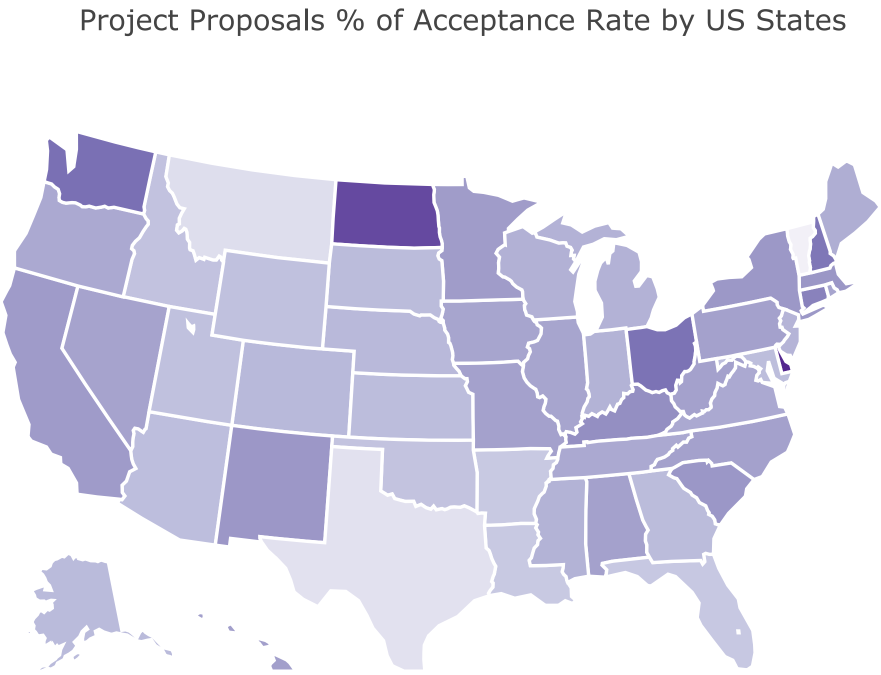
```
Number of projects thar are approved for funding  92706 , ( 84.85830404217927 %)
Number of projects thar are not approved for funding  16542 , ( 15.141695957820739 %)
```



Observations - The above plot shows that approximately 85% of the projects are approved for funding while 15% of them are rejected.

1.2.1 Univariate Analysis: School State

In [6]:
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039
#Plotting US states heat map for different percentage of proposals

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].apply(np.mean)).reset_inde
# 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')
```



Observation - The above heat map shows

1. Highest percentage of Approval rates in the states of North-Dakota & Delaware.
2. The above states are followed by the states of Ohio,New Hampshire & Washington.

3. Rest of the states have lower percentage of project approval than the states mentioned above.

In [7]:

```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.pdf
# Percentage of approval rates for various states in US

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
```

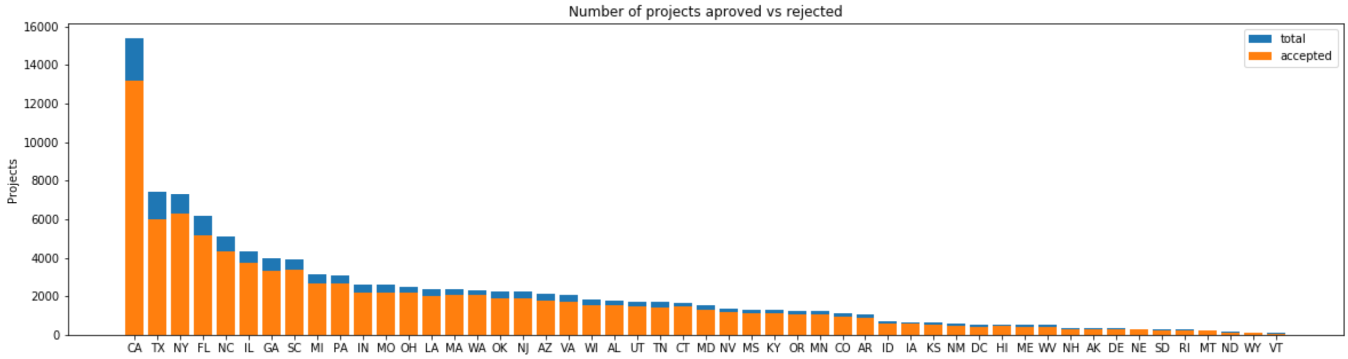
In [247]:

```
#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 [9]:

```
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()['
    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 [10]:
```
univariate_barplots(project_data, 'school_state', 'project_is_approved', False)
```

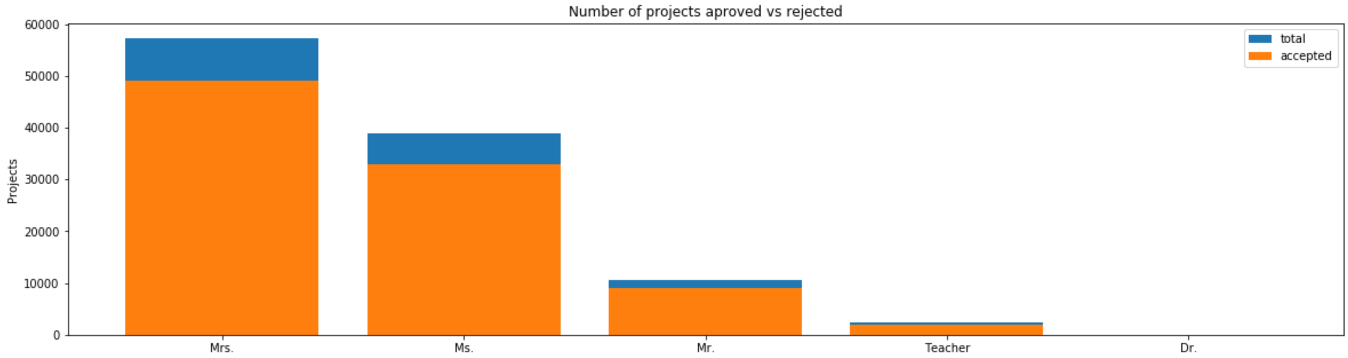


```
   school_state  project_is_approved  total       Avg
4            CA                13205  15388  0.858136
43           TX                 6014   7396  0.813142
34           NY                 6291   7318  0.859661
9            FL                 5144   6185  0.831690
27           NC           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

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

1.2.2 Univariate Analysis: teacher_prefix

In [11]:
```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)
```

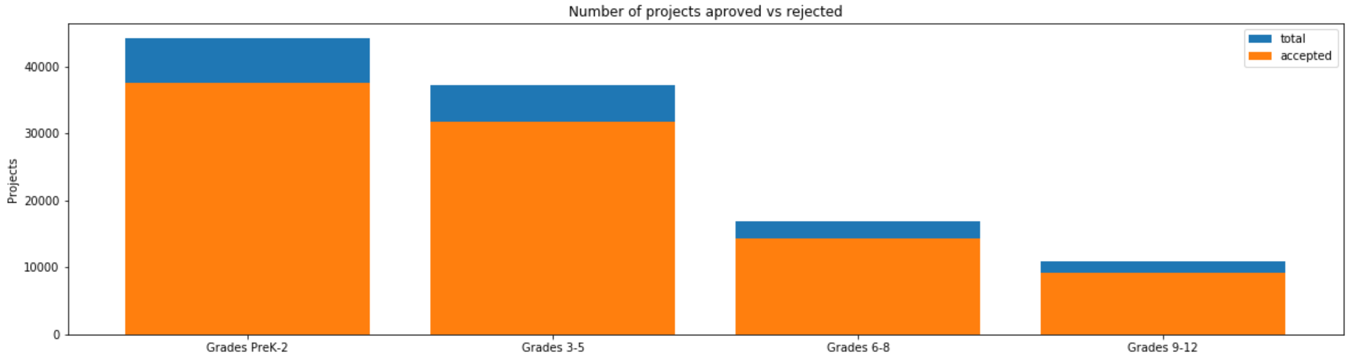


```
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.     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

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

1.2.3 Univariate Analysis: project_grade_category

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



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

1.2.4 Univariate Analysis: project_subject_categories

In [13]:
```python
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

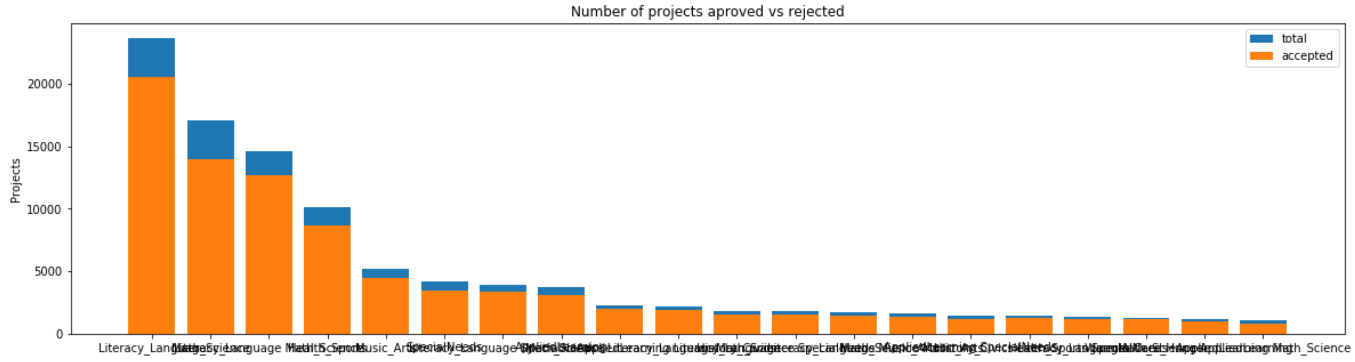
In [14]:
```python
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[14]:

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

In [15]:

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



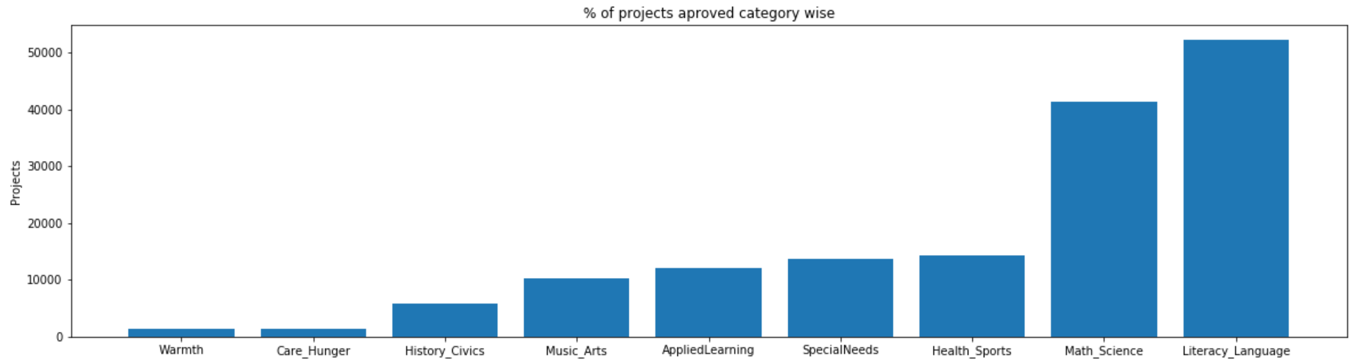
```
                   clean_categories  project_is_approved  total       Avg
24                Literacy_Language                20520  23655  0.867470
32                     Math_Science                13991  17072  0.819529
28   Literacy_Language Math_Science                12725  14636  0.869432
8                     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
```

In [16]:

```
# 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 [17]:

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


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

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



In [18]:
```
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

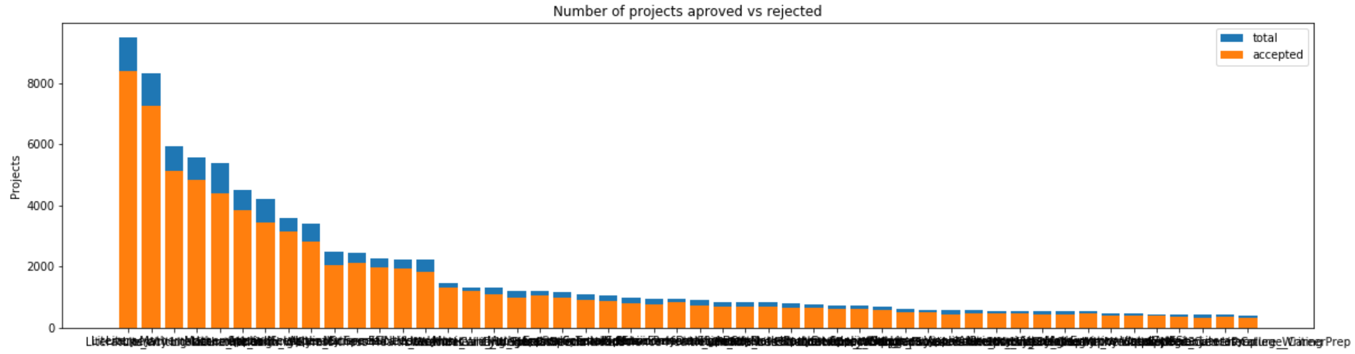
In [19]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

In [20]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[20]:

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

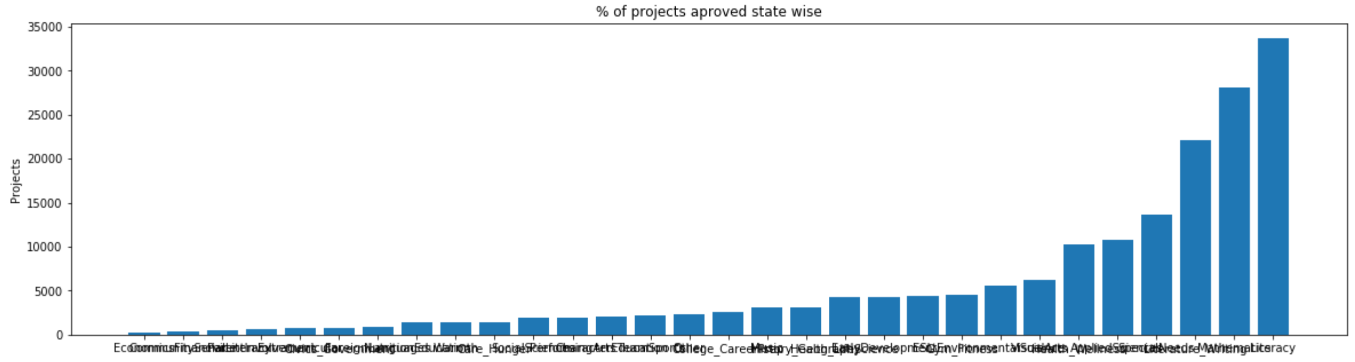
```
In [21]: univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top=50)
```



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

```
In [22]: # 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 [23]: # dict sort by value python: https://stackoverflow.com/a/613218/4084039
         sub_cat_dict = dict(my_counter)
         sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))


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

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

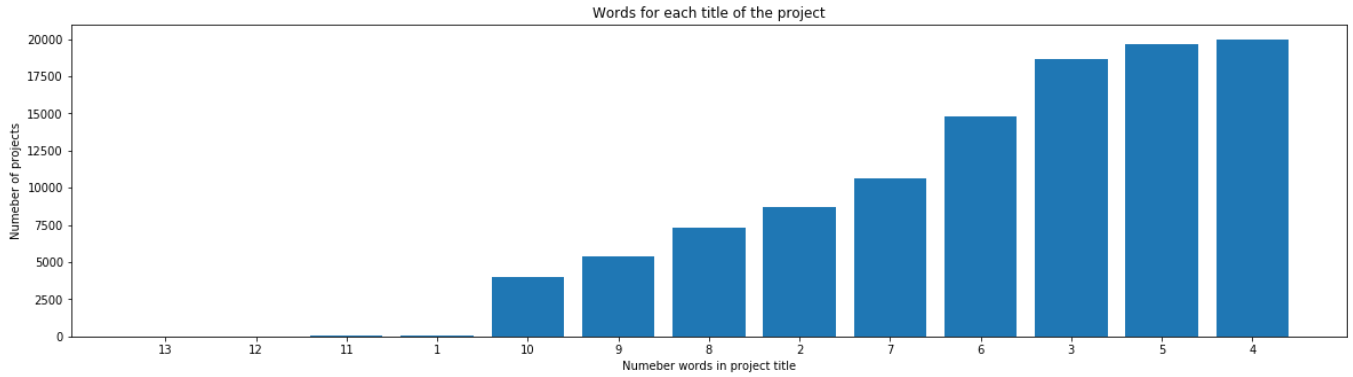


In [24]:
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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)

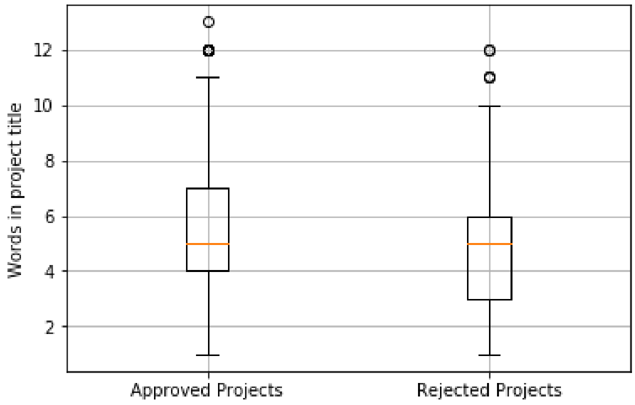
In [25]:
```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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

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



In [26]:
```
approved_title_word_count = project_data[project_data['project_is_approved']==1]['project_title'].str.spl
approved_title_word_count = approved_title_word_count.values

rejected_title_word_count = project_data[project_data['project_is_approved']==0]['project_title'].str.spl
rejected_title_word_count = rejected_title_word_count.values
```

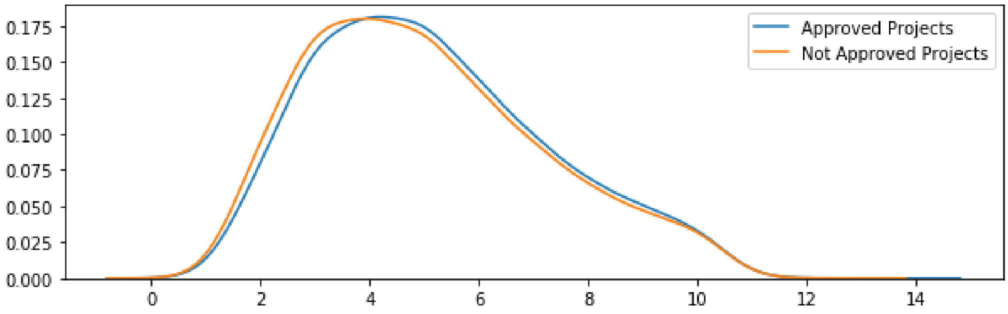
In [27]:

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



In [28]:

```
plt.figure(figsize=(10,3))
sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)
sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6)
plt.legend()
plt.show()
```



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

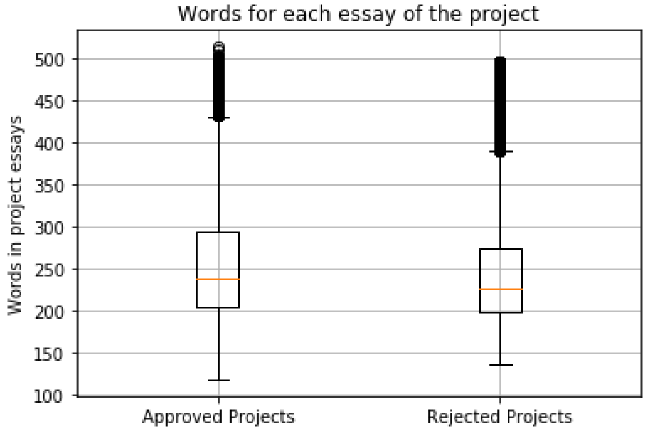
In [29]:

```
# 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)
```

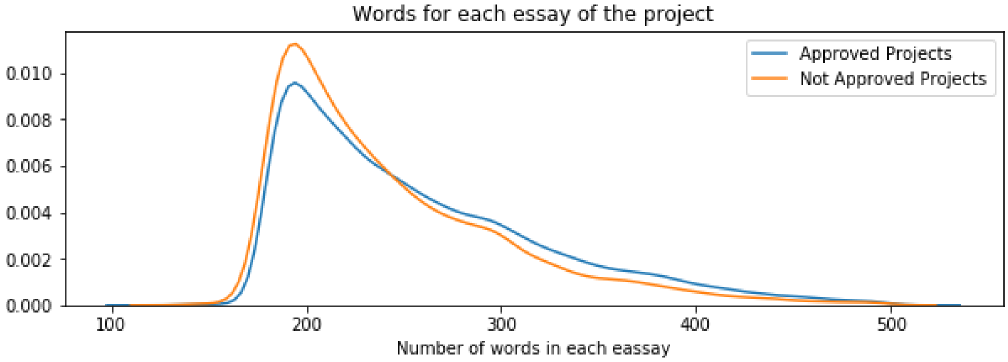
In [30]:

```
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 [31]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()
```



In [32]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each essay of the project')
plt.xlabel('Number of words in each eassay')
plt.legend()
plt.show()
```



1.2.8 Univariate Analysis: Cost per project

In [33]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[33]:

	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 [34]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-ste
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[34]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

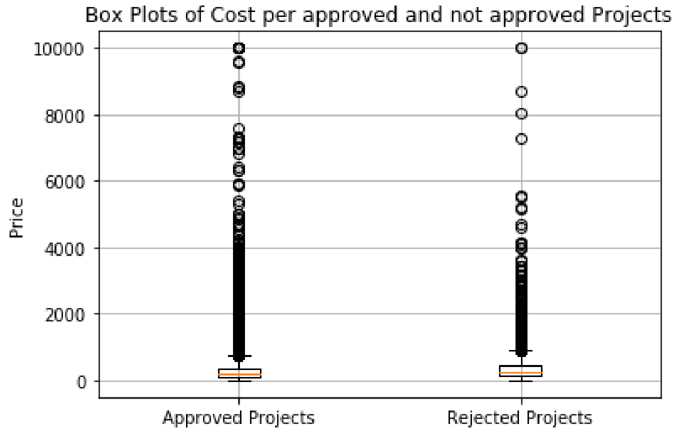
In [37]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
project_data.head(2)
```

Out[37]:

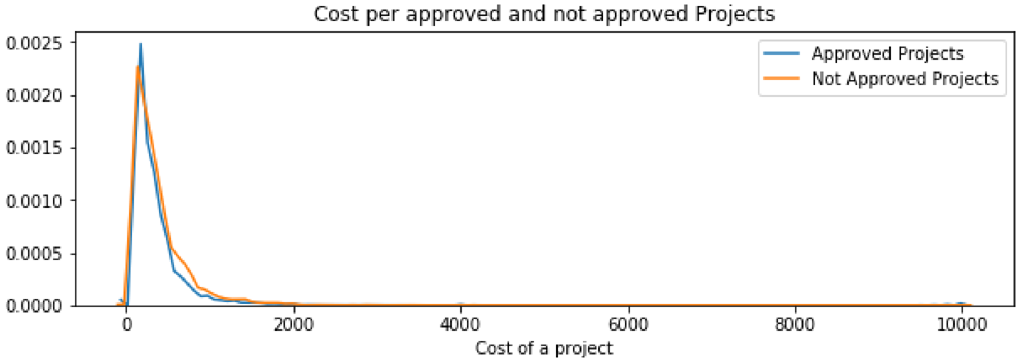
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grad
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	G

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

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

In [39]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Price')
plt.grid()
plt.show()
```



In [40]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_price, hist=False, label="Approved Projects")
sns.distplot(rejected_price, hist=False, label="Not Approved Projects")
plt.title('Cost per approved and not approved Projects')
plt.xlabel('Cost of a project')
plt.legend()
plt.show()
```



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

#If you get a ModuleNotFoundError error , install prettytable using: pip3 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_price,i), 
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 [42]:
```
#Please do this on your own based on the data analysis that was done in the above cells
#printing rows from project data dataframe
project_data.head(2)
```

Out[42]:

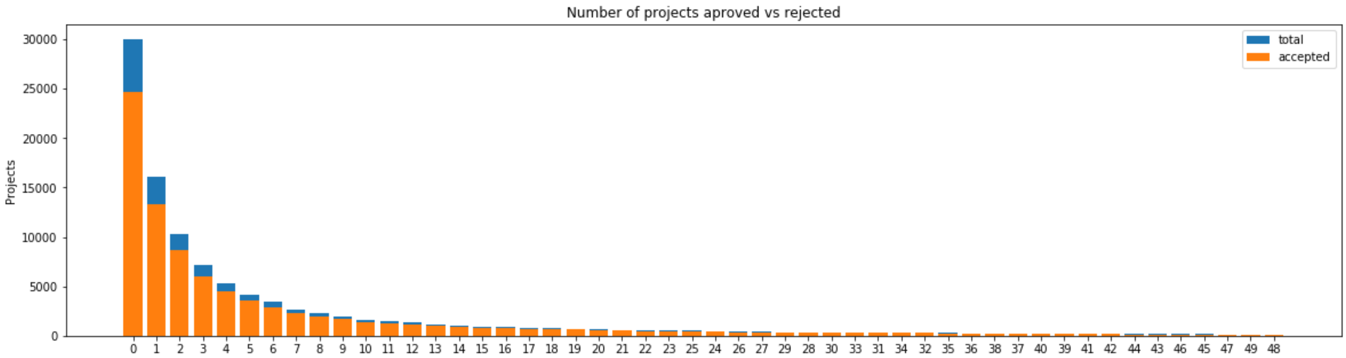
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grad
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	G

```
In [43]: #Making a copy of project_data Dataframe
         teacher_data = project_data
         teacher_data.head(2)
```

Out[43]:

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

```
In [248]: #Plotting for number of projects previously posted by teachers
          univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects', 'project_is_approved', '
```



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

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

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

```
Summary :-

1. Number of Approved projects are highest for the teacher who have not submitted any projects
previously which is                 approximately 30000 projects.

2. With any teacher that has submitted one project previously this count drops to approximately 16000.
```

```
3. From the above plot we can observe that as the number of previously submitted projects goes on
increasing the number of        approved projects keep on decreasing.

4. There is a wide spread in number of previously submitted projects.
```

1.2.10 Univariate Analysis: project_resource_summary

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

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

In [48]:

```
resource_description = resource_data.filter(['description'], axis=1)
resource_description.head(2)
```

Out[48]:

	description
0	LC652 - Lakeshore Double-Space Mobile Drying Rack
1	Bouncy Bands for Desks (Blue support pipes)

In [296]:

```
#Check if a string has numbers python - https://stackoverflow.com/questions/19859282/check-if-a-string-con
def hasNumbers(inputString):
    return int(bool(re.search(r'\d+\.?\d', inputString)))
#r'\d+\.?\d*
print(hasNumbers("he has wolves and horses.he left the town with 57 horses"))
```

```
1
```

In [306]:

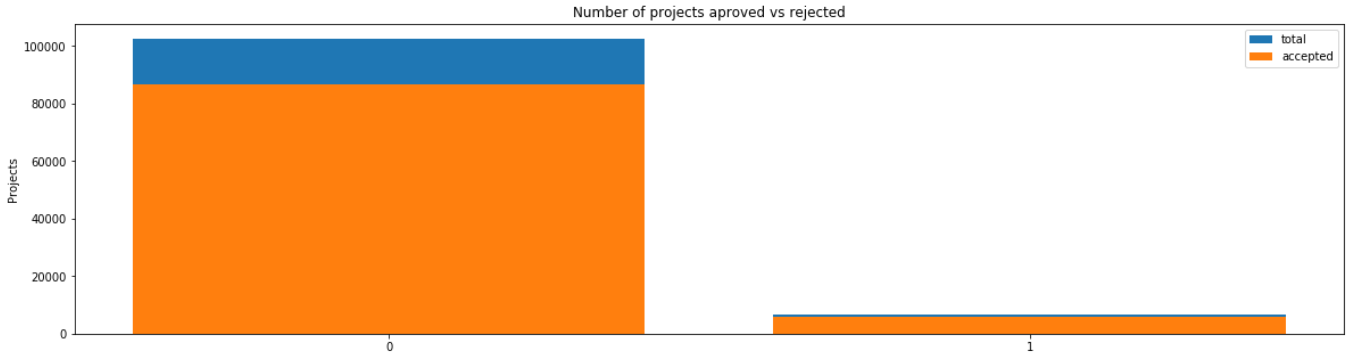
```
#Checking how many summaries in the dataset have digits in them
res=[]
for i in project_data['project_resource_summary']:
    res.append(hasNumbers(i))
project_data['presence_of_the_numerical_digits']=res
np.count_nonzero(project_data[project_data['project_is_approved']==1]['presence_of_the_numerical_digits']

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

Out[306]:

```
0    102572
1      6676
Name: presence_of_the_numerical_digits, dtype: int64
```

In [307]:

```
#Performing univariate analysis on the approved projects & summaries having numerical digits in them
univariate_barplots(project_data, 'presence_of_the_numerical_digits', 'project_is_approved')
```



```
   presence_of_the_numerical_digits  project_is_approved   total       Avg
0                                 0                86753  102572  0.845777
1                                 1                 5953    6676  0.891702
==================================================
   presence_of_the_numerical_digits  project_is_approved   total       Avg
0                                 0                86753  102572  0.845777
1                                 1                 5953    6676  0.891702
```

```
Summary :-
```

```
1. There are 102572 summaries which do not have any numerical digit in them out of which 86753 projects
have still been            approved.

2. There are 6676 summaries which have numerical digits in them out of which 5953 have got approved.

3. The approval percentage for case 1 is approximately 84%.

4. The approval percentage for case 2 is approximately 89%.

5. So we can conclude that the presence of numerical digits in the summary does not affect the approval
of project.
```

1.3 Text preprocessing

1.3.1 Essay Text

In [51]:
```
project_data.head(2)
```

Out[51]:

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

2 rows × 21 columns

In [52]:

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

```
My students are English learners that are working on English as their second or third languages. We are
a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our sc
hool. \r\n\r\n We have over 24 languages represented in our English Learner program with students at eve
ry level of mastery.  We also have over 40 countries represented with the families within our school.  E
ach student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, belie
fs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Our E
nglish learner's have a strong support system at home that begs for more resources.  Many times our pare
nts are learning to read and speak English along side of their children.  Sometimes this creates barrier
s for parents to be able to help their child learn phonetics, letter recognition, and other reading skil
ls.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the Engl
ish language even if no one at home is able to assist.  All families with students within the Level 1 pr
oficiency status, will be a offered to be a part of this program.  These educational videos will be spec
ially chosen by the English Learner Teacher and will be sent home regularly to watch.  The videos are to
help the child develop early reading skills.\r\n\r\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
educational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least m
ost of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 st
udents, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together a
nd celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that st
udents wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and gam
es. At the end of the year the school hosts a carnival to celebrate the hard work put in during the scho
ol year, with a dunk tank being the most popular activity.My students will use these five brightly color
ed 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 have 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 o
ccasion. I will utilize them in place of chairs at my small group tables during math and reading times.
The rest of the day they will be used by the students who need the highest amount of movement in their l
ife 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 studen
ts are sitting in 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 disapp
ointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The H
okki stools will be a compromise that allow my students to do desk work and move at the same time. These
stools will help students to meet their 60 minutes a day of movement by allowing them to activate their
core muscles for balance while they sit. For many of my students, these chairs will take away the barrie
r 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 desk
s, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to cr
eate 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 sch
ool, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our schoo
l is an \"open classroom\" concept, which is very unique as there are no walls separating the classroom
s. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the in
formation and experiences and keep on wanting more.With these resources such as the comfy red throw pill
ows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the m
ood in our classroom setting to be one of a themed nautical environment. Creating a classroom environmen
t 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 evenin
g. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready
for their first day of 4th grade.  This kind gesture will set the tone before even the first day of scho
ol! The nautical thank you cards will be used throughout the year by the students as they create thank y
ou cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a
fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on
resources to get our classroom ready. Please consider helping with this project to make our new school y
ear a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive del
ays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest
```

```
working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced price lunch.  Despite their
disabilities and limitations, my students love coming to school and come eager to learn and explore.Have
you ever felt like you had ants in your pants and you needed to groove and move as you were in a meetin
g? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobbl
e chairs are the answer and I love then because they develop their core, which enhances gross motor and
in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do w
orksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our suc
cess. The number toss and color and shape mats can make that happen. My students will forget they are do
ing work and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teac
her inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-Americ
an, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% Af
rican-American students. Most of the students are on free or reduced lunch. We aren't receiving doctors,
lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring min
ds of young children and we focus not only on academics but one smart, effective, efficient, and discipl
ined students with good character.In our classroom we can utilize the Bluetooth for swift transitions du
ring class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the vo
lume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as mea
ningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and repla
y it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for
the day and has an extra part to it I can use.  The table top chart has all of the letter, words and pic
tures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [53]:

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

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

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

In [54]:

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

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

In [55]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive del
ays, 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 students. I tea
ch in a Title I school where most of the students receive free or reduced price lunch.  Despite their di
sabilities and limitations, my students love coming to school and come eager to learn and explore.Have y
ou ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting?
This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble c
hairs are the answer and I love then because they develop their core, which enhances gross motor and in
Turn fine motor skills.   They also want to learn through games, my kids do not want to sit and do works
heets. They want to learn to count by jumping and playing. Physical engagement is the key to our succes
s. The number toss and color and shape mats can make that happen. My students will forget they are doing
work and just have the fun a 6 year old deserves.nannan
```

In [56]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitive dela
ys gross fine motor delays to autism They are eager beavers and always strive to work their hardest work
ing past their limitations The materials we have are the ones I seek out for my students I teach in a Ti
tle I school where most of the students receive free or reduced price lunch Despite their disabilities a
nd limitations my students love coming to school and come eager to learn and explore Have you ever felt
like you had ants in your pants and you needed to groove and move as you were in a meeting This is how m
y kids feel all the time The want to be able to move as they learn or so they say Wobble chairs are the
answer and I love then because they develop their core which enhances gross motor and in Turn fine motor
skills They also want to learn through games my kids do not want to sit and do worksheets They want to l
earn to count by jumping and playing Physical engagement is the key to our success The number toss and c
olor and shape mats can make that happen My students will forget they are doing work and just have the f
un a 6 year old deserves nannan
```

In [57]:
```
# 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', 'the
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', '
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do'
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while'
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'aga
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each',
            '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'
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'we
            'won', "won't", 'wouldn', "wouldn't"]
```

In [58]:
```
# 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 [01:12<00:0
0, 1503.90it/s]
```

In [59]:

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

Out[59]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross 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 lunch despite disabiliti
es limitations students love coming school come eager learn explore have ever felt like ants pants neede
d groove move meeting this kids feel time the want able move learn say wobble chairs answer i love devel
op core enhances gross motor turn fine motor skills they also want learn games kids not want sit workshe
ets they want learn count jumping playing physical engagement key success the number toss color shape ma
ts make happen my students forget work fun 6 year old deserves nannan'
```

1.3.2 Project title Text

In [60]:

```
# similarly you can preprocess the titles also
#Printing the columns in the dataframe
project_data.head(2)
```

Out[60]:

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

2 rows × 21 columns

In [61]:

```
# printing some random project titles.

print(project_data['project_title'].values[25])
print("="*50)
print(project_data['project_title'].values[72])
print("="*50)
print(project_data['project_title'].values[964])
print("="*50)
print(project_data['project_title'].values[10240])
print("="*50)
print(project_data['project_title'].values[89656])
print("="*50)
```

```
Math Masters!
==================================================
Gotta Catch a ChromeBook!
==================================================
Virtual Field Trips for KG Kids
==================================================
Warming Up With Fitness and Gaming!
==================================================
World Weather Investigations
==================================================
```

In [62]:
```
#Removing phrases from the title features
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"Gotta",  "Got to",  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 [63]:
```
#Checkingt titles after removing phrases
sent = decontracted(project_data['project_title'].values[72])
print(sent)
print("="*50)
```

```
Got to Catch a ChromeBook!
==================================================
```

In [64]:
```
# Remove \\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)
```

```
Got to Catch a ChromeBook!
```

In [65]:
```
#Removing numbers & symbols form the titles
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Got to Catch a ChromeBook
```

In [66]:
```
#Removing stop words from the preprocessed titles
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', 'the
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 't
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while',
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'aga
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 
            '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',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'r
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'we
            'won', "won't", 'wouldn', "wouldn't"]
```

In [67]:

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

```
100%|██████████████████████████████████████████████████████████████████████| 109248/109248 [00:03<00:0
0, 29294.63it/s]
```

In [68]:

```
#checking cleaned text after preprocesing
preprocessed_titles[72]
```

Out[68]:

```
'got catch chromebook'
```

1. 4 Preparing data for models

In [69]:

```
#Printing columns for project_data Dataframe
project_data.columns
```

Out[69]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'project_numcheck_rs'],
      dtype='object')
```

we are going to consider

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

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

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

1.4.1 Vectorizing Categorical data

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

In [70]:

```
# we use count vectorizer to convert the values from categories 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_Spo
rts', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [71]:

```
# we use count vectorizer to convert the values from subcategories 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_G
overnment', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'Perfor
mingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geograph
y', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArt
s', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literac
y']
Shape of matrix after one hot encodig  (109248, 30)
```

In [72]:

```
#Checking values in school_state
states = project_data[['school_state']]
states.head(2)
```

Out[72]:

	school_state
0	IN
1	FL

In [73]:

```
#Chcecking count of different states in the dataframe
project_data['school_state'].value_counts()
```

Out[73]:

```
CA    15388
TX     7396
NY     7318
FL     6185
NC     5091
IL     4350
GA     3963
SC     3936
MI     3161
PA     3109
IN     2620
MO     2576
OH     2467
LA     2394
MA     2389
WA     2334
OK     2276
NJ     2237
AZ     2147
VA     2045
WI     1827
AL     1762
UT     1731
TN     1688
CT     1663
MD     1514
NV     1367
MS     1323
KY     1304
OR     1242
MN     1208
CO     1111
AR     1049
ID      693
IA      666
KS      634
NM      557
DC      516
HI      507
ME      505
WV      503
NH      348
AK      345
DE      343
NE      309
SD      300
RI      285
MT      245
ND      143
WY       98
VT       80
Name: school_state, dtype: int64
```

In [73]:

```
#Converting states text into smaller case
project_data['school_state'] = project_data['school_state'].str.lower()
project_data['school_state'].value_counts()
```

Out[73]:

```
ca    15388
tx     7396
ny     7318
fl     6185
nc     5091
il     4350
ga     3963
sc     3936
mi     3161
pa     3109
in     2620
mo     2576
oh     2467
la     2394
ma     2389
wa     2334
ok     2276
nj     2237
az     2147
va     2045
wi     1827
al     1762
ut     1731
tn     1688
ct     1663
md     1514
nv     1367
ms     1323
ky     1304
or     1242
mn     1208
co     1111
ar     1049
id      693
ia      666
ks      634
nm      557
dc      516
hi      507
me      505
wv      503
nh      348
ak      345
de      343
ne      309
sd      300
ri      285
mt      245
nd      143
wy       98
vt       80
Name: school_state, dtype: int64
```

In [74]:

```
# Applying count vectorizer  on school state feature & one hot encoding School_state feature
vectorizer = CountVectorizer(binary=True)
school_state_count = vectorizer.fit_transform(data['school_state'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",school_state_count.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks',
'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'n
y', '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 [75]:

```
# Finding the count of different values of teacher prefix feature
project_data_50['teacher_prefix'].value_counts()
```

Out[75]:

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

In [76]:

```
# check if we have any nan values are there in the column
print(project_data_50['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data_50['teacher_prefix'].isnull().values.sum())
```

```
True
number of nan values 3
```

In [77]:

```
#Replacing the Nan values with most frequent value in the column
project_data_50['teacher_prefix']=project_data_50['teacher_prefix'].fillna('Mrs.')
```

In [78]:

```
#Counting the values for different teacher prefix after removing Nan values
project_data['teacher_prefix'].value_counts()
```

Out[78]:

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

In [79]:

```
#Checking whether the Nan values have been removed from the copy of the data frame
project_data_50['teacher_prefix'].value_counts()
```

Out[79]:

```
Mrs.       57272
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
Name: teacher_prefix, dtype: int64
```

In [80]:

```
#One hot encoding the teacher prefix column
vectorizer = CountVectorizer(binary=True)
teacher_prefix_one = vectorizer.fit_transform(project_data_50['teacher_prefix'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",teacher_prefix_one.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encodig  (109248, 5)
```

In [81]:

```
#Checking the count of different values of project_grade_category
project_data['project_grade_category'].value_counts()
```

Out[81]:

```
Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
```

In [82]:

```
#Replacing spaces & hyphens in the text of project grade category with underscore
#converting Capital letters in the string to smaller letters
#Performing avalue count of project grade category
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-on-
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[82]:

```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

In [83]:

```
#One hot encoding project grade category feature
vectorizer = CountVectorizer(binary=True)
project_grade_one = vectorizer.fit_transform(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encoding ",project_grade_one.shape)
```

```
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix after one hot encoding  (109248, 4)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [84]:

```
# 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 encoding ",text_bow.shape)
```

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

1.4.2.2 Bag of Words on `project_title`

In [85]:

```
# Similarly you can vectorize for title also
# We are considering only the words which appeared in at least 10 documents(rows or projects).
#Vectorizing & one hot encoing project title feature
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",title_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3328)
```

1.4.2.3 TFIDF vectorizer

In [86]:

```
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 [87]:

```
#Vectorizing & one hot encoding project title using tfidf vectorization

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

```
Shape of matrix after one hot encoding  (109248, 3328)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [ ]:

```
'''
# 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-save-and-l

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


'''
```

In [88]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-l
# 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 [89]:

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

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

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248 [00:50<00:0
0, 2167.45it/s]

109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [90]:

```
# Similarly you can vectorize for title also
# Vectorizing project_title using avgw2v method
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
    avg_w2v_vectors.append(vector)

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

```
100%|███████████████████████████████████████████████████████████████████████| 109248/109248 [00:02<00:0
0, 43348.69it/s]

109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [91]:

```
# 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 [92]:
```python
# 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)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf val
            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 [05:18<00:
00, 343.54it/s]

109248
300
```

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

In [93]:
```python
# Similarly you can vectorize for title also
# vectorizing project_title using TFIDF weighted W2V pretrained model
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [94]:
```python
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 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)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf val
            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 [00:04<00:0
0, 21856.61it/s]

109248
300
```

1.4.3 Vectorizing Numerical features

In [98]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Standar
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data_70['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_70['price'].values.reshape(-1,1)) # finding the mean and standard deviation
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
price_standardized = price_scalar.transform(project_data_70['price'].values.reshape(-1, 1))
```

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

In [99]:
```
price_standardized
```

Out[99]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

In [101]:
```
teacher_prev_standardized
```

Out[101]:
```
array([[-0.40152481],
       [-0.14951799],
       [-0.36552384],
       ...,
       [-0.29352189],
       [-0.40152481],
       [-0.40152481]])
```

1.4.4 Merging all the above features

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

In [102]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

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

In [103]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
# Stacking all the features
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[103]:
```
(109248, 16663)
```

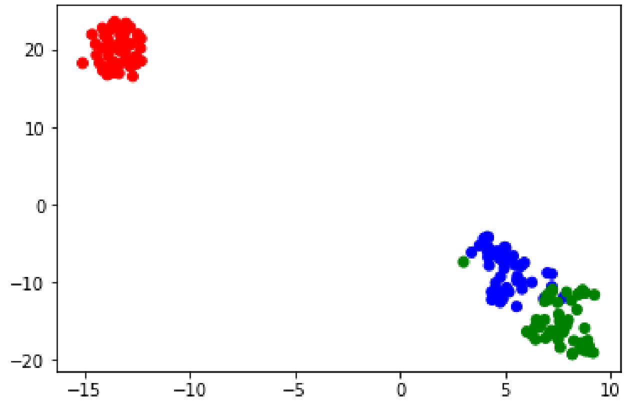
Assignment 2: Apply TSNE

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

1. In the above cells we have plotted and analyzed many features. Please observe the plots and write the observations in markdown cells below every plot.
2. EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects
3. Build the data matrix using these features
   - school_state : categorical data (one hot encoding)
   - clean_categories : categorical data (one hot encoding)
   - clean_subcategories : categorical data (one hot encoding)
   - teacher_prefix : categorical data (one hot encoding)
   - project_grade_category : categorical data (one hot encoding)
   - project_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)
   - price : numerical
   - teacher_number_of_previously_posted_projects : numerical
4. Now, plot FOUR t-SNE plots with each of these feature sets.
   A. categorical, numerical features + project_title(BOW)
   B. categorical, numerical features + project_title(TFIDF)
   C. categorical, numerical features + project_title(AVG W2V)
   D. categorical, numerical features + project_title(TFIDF W2V)
5. Concatenate all the features and Apply TNSE on the final data matrix
6. Note 1: The TSNE accepts only dense matrices
7. Note 2: Consider only 5k to 6k data points to avoid memory issues. If you run into memory error issues, reduce the number of data points but clearly state the number of datat-poins you are using

In [122]:

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

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

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Score'].apply(lambda x
plt.show()
```



One hot encoding all the features for 6k points

In [104]:

```
#considering first 6k rows from the data frame project_data
project_6k = project_data_70.head(6000)
project_6k.shape
```

Out[104]:

```
(6000, 20)
```

In [118]:

```
#one hot encoding project categories with 6k point
# 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_6k['clean_categories'].values)
print(vectorizer.get_feature_names())


categories_one_hot_6k = vectorizer.transform(project_6k['clean_categories'].values)
print("Shape of matrix after one hot encodig ",categories_one_hot_6k.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Spo
rts', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (6000, 9)
```

In [119]:

```
#one hot encoding project sub-categories with 6k point
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_6k['clean_subcategories'].values)
print(vectorizer.get_feature_names())
sub_categories_one_hot_6k = vectorizer.transform(project_6k['clean_subcategories'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_6k.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_G
overnment', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'Perfor
mingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geograph
y', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArt
s', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literac
y']
Shape of matrix after one hot encodig  (6000, 30)
```

In [120]:

```
# Applying count vectorizer  on school state feature for 6k points
vectorizer = CountVectorizer(binary=True)
school_state_count_6k = vectorizer.fit_transform(project_6k['school_state'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",school_state_count_6k.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks',
'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'n
y', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
Shape of matrix after one hot encodig  (6000, 51)
```

In [121]:

```
# Applying count vectorizer  on teacher prefix feature for 6k points
vectorizer = CountVectorizer(binary=True)
teacher_prefix_one_6k = vectorizer.fit_transform(project_6k['teacher_prefix'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",teacher_prefix_one_6k.shape)
```

```
['mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encodig  (6000, 4)
```

In [123]:

```
# Applying count vectorizer  on project grade feature for 6k points
project_6k['project_grade_category'] = project_6k['project_grade_category'].str.replace(' ','_')
project_6k['project_grade_category'] = project_6k['project_grade_category'].str.replace('-','_')
project_6k['project_grade_category'] = project_6k['project_grade_category'].str.lower()
project_6k['project_grade_category'].value_counts()
```

Out[123]:

```
grades_prek_2    2422
grades_3_5       2048
grades_6_8        933
grades_9_12       597
Name: project_grade_category, dtype: int64
```

In [124]:
```
# Applying count vectorizer  on project grade feature for 6k points
vectorizer = CountVectorizer(binary=True)
project_grade_one_6k = vectorizer.fit_transform(project_6k['project_grade_category'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encoding ",project_grade_one_6k.shape)
```

```
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix after one hot encoding  (6000, 4)
```

In [309]:
```
# Applying count vectorizer  on teacher previosuly submitted projects feature for 6k points
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data_70['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)

teacher_prev_scalar = StandardScaler()
teacher_prev_scalar.fit(project_6k['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)) #
print(f"Mean : {teacher_prev_scalar.mean_[0]}, Standard deviation : {np.sqrt(teacher_prev_scalar.var_[0])

# Now standardize the data with above maen and variance.
teacher_prev_standardized_6k = teacher_prev_scalar.transform(project_6k['teacher_number_of_previously_post

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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


Mean : 10.913666666666666, Standard deviation : 27.296310127113436

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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

In [127]:
```
teacher_prev_standardized_6k.shape
```

Out[127]:
```
(6000, 1)
```

In [128]:
```
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data_70['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_6k['price'].values.reshape(-1,1)) # finding the mean and standard deviation of th
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_6k = price_scalar.transform(project_6k['price'].values.reshape(-1, 1))
```

```
Mean : 300.482685, Standard deviation : 379.1594914082649
```

In [129]:
```
price_standardized_6k.shape
```

Out[129]:
```
(6000, 1)
```

In [105]:
```
#applying bow,tfidf,avgw2v,tfidf w2v for project title feature on 6k points
from tqdm import tqdm
preprocessed_titles_6k = []
# tqdm is for printing the status bar
for sentance in tqdm(project_6k['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles_6k.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████████| 6000/6000 [00:00<00:0
0, 16527.82it/s]
```

In [106]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[106]:
```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross 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 lunch despite disabiliti
es limitations students love coming school come eager learn explore have ever felt like ants pants neede
d groove move meeting this kids feel time the want able move learn say wobble chairs answer i love devel
op core enhances gross motor turn fine motor skills they also want learn games kids not want sit workshe
ets they want learn count jumping playing physical engagement key success the number toss color shape ma
ts make happen my students forget work fun 6 year old deserves nannan'
```

In [107]:
```
# Similarly you can vectorize for title also
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
title_bow_6k = vectorizer.fit_transform(preprocessed_titles_6k)
print("Shape of matrix after one hot encodig ",title_bow_6k.shape)
```

```
Shape of matrix after one hot encodig  (6000, 450)
```

In [145]:
```
#applying tfidf for project title feature on 6k points
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
title_tfidf_6k = vectorizer.fit_transform(preprocessed_titles_6k)
print("Shape of matrix after one hot encoding ",title_tfidf_6k.shape)
```

```
Shape of matrix after one hot encoding  (6000, 450)
```

In [110]:
```
#applying avgw2v for project title feature on 6k points
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [111]:

```
# average Word2Vec
# compute average word2vec for first 6k titles.
avg_w2v_vectors_6k = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_6k): # 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_6k.append(vector)

print(len(avg_w2v_vectors_6k))
print(len(avg_w2v_vectors_6k[0]))
```

```
100%|███████████████████████████████████████████████████████████████████████████| 6000/6000 [00:00<00:0
0, 30452.39it/s]

6000
300
```

In [112]:

```
#applying tfidf w2v for project title feature on 6k points
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles_6k)
# 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_6k = set(tfidf_model.get_feature_names())
```

In [113]:

```
tfidf_w2v_vectors_6k = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_6k): # 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)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf va
            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_6k.append(vector)

print(len(tfidf_w2v_vectors_6k))
print(len(tfidf_w2v_vectors_6k[0]))
```

```
100%|███████████████████████████████████████████████████████████████████████████| 6000/6000 [00:00<00:0
0, 18574.30it/s]

6000
300
```

In [130]:

```
#Checking the shapes of all the one hot encoded vectors
print(categories_one_hot_6k.shape)
print(sub_categories_one_hot_6k.shape)
print(title_bow_6k.shape)
print(price_standardized_6k.shape)
print(school_state_count_6k.shape)
print(teacher_prefix_one_6k.shape)
print(project_grade_one_6k.shape)
print(teacher_prev_standardized_6k.shape)
```

```
(6000, 9)
(6000, 30)
(6000, 450)
(6000, 1)
(6000, 51)
(6000, 4)
(6000, 4)
(6000, 1)
```

In [131]:

```
#One hot encoding Categorical & Numerical features
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Z = hstack((school_state_count_6k, categories_one_hot_6k, sub_categories_one_hot_6k, teacher_prefix_one_6l
Z.shape
```

Out[131]:

```
(6000, 100)
```

In [132]:

```
#One hot encoding Categorical & Numerical features with project title BOW
Z1 = hstack((Z, title_bow_6k ))
Z1.shape
```

Out[132]:

```
(6000, 550)
```

In [133]:

```
Z1 =  Z1.toarray()
print(Z1)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]]
```

In [140]:

```
Z1.shape
```

Out[140]:

```
(6000, 550)
```

In [135]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_6k['clean_categories'].values)
print(vectorizer.get_feature_names())

categories_one_hot_6k = vectorizer.transform(project_6k['clean_categories'].values)
print("Shape of matrix after one hot encodig ",categories_one_hot_6k.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Spo
rts', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (6000, 9)
```

In [100]:

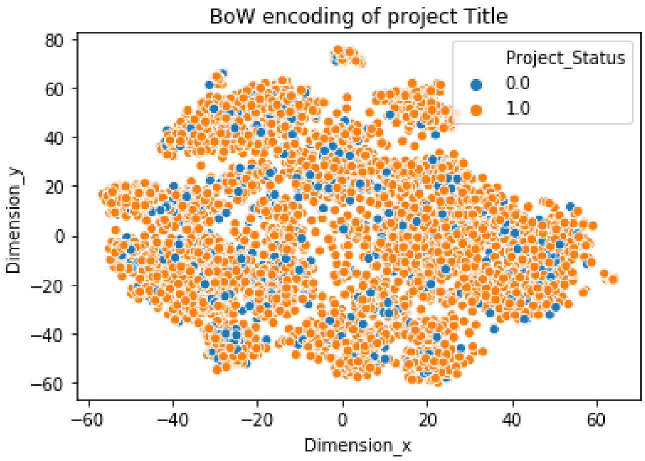
```
#Standardizing teacher prefix feature
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data_70['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)

teacher_prev_scalar = StandardScaler()
teacher_prev_scalar.fit(project_data_70['teacher_number_of_previously_posted_projects'].values.reshape(-1,
print(f"Mean : {teacher_prev_scalar.mean_[0]}, Standard deviation : {np.sqrt(teacher_prev_scalar.var_[0])}

# Now standardize the data with above mean and variance.
teacher_prev_standardized = teacher_prev_scalar.transform(project_data_70['teacher_number_of_previously_po
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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


Mean : 11.153165275336848, Standard deviation : 27.77702641477403

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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

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

In [221]:

```
#Applying tsne with BOW encoding of project title feature
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z1
y = project_6k['project_is_approved']

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("BoW encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```

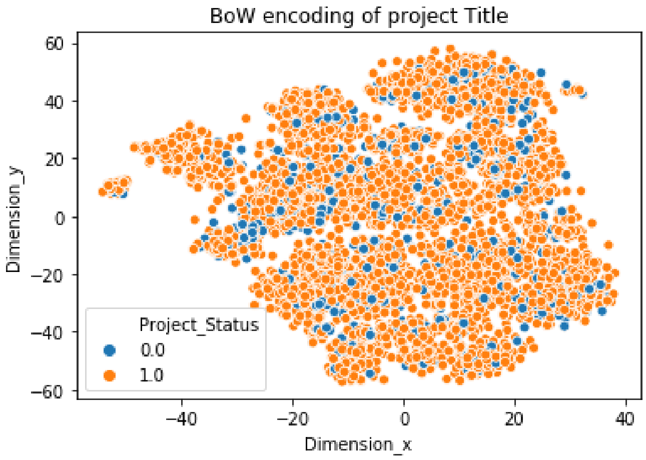


In [ ]:

```
Summary -
   1. The above plot shows  what is the shape of our data but there is no seperation of points & all the
      overlapping each other.

   2. So it is very difficult to come to make any conclusion with the above plot.
```

In [224]:

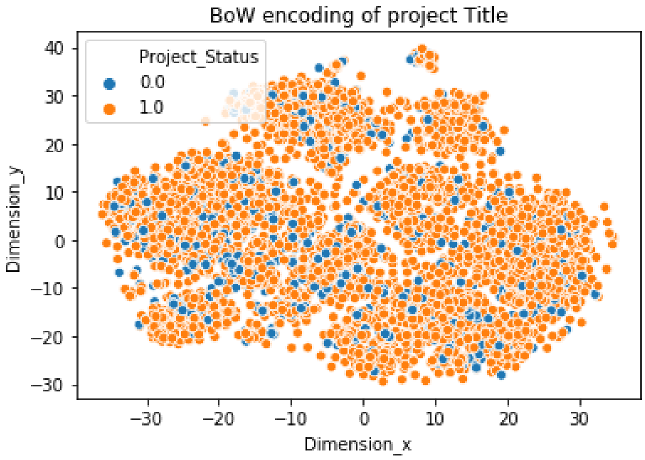
```
#Running TSNE with different values of perplexity
#with peplexity 50 & learning rate = 200
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z1
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=50, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("BoW encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)

plt.show()
```



```
Summary -
   1. With higher perplexity value the shape of the overall points changes.

   2. But still both the groups are intermixed with each & there is no seperation between these
different groups of points.
```

In [225]:

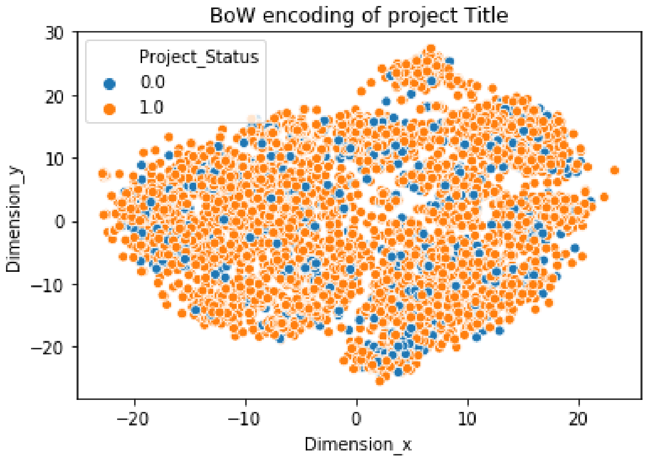
```
#with peplexity 100 & learning rate = 200
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z1
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("BoW encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

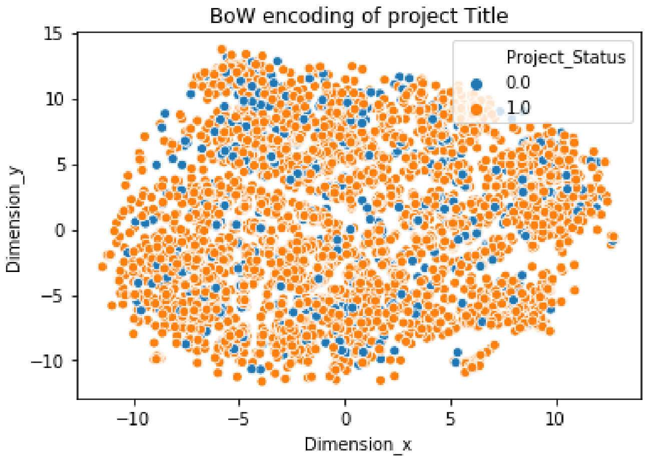
1. Even with perplexity value of 100 there is no seperation of points in the plot.

In [226]:
```
#with peplexity 200 & learning rate = 200
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z1
y = project_6k['project_is_approved']

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("BoW encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -
   1. Considering perplexity value of 200 still the plot barely shows any sepeartion of points which is still not good enough   to come to any conclusion.

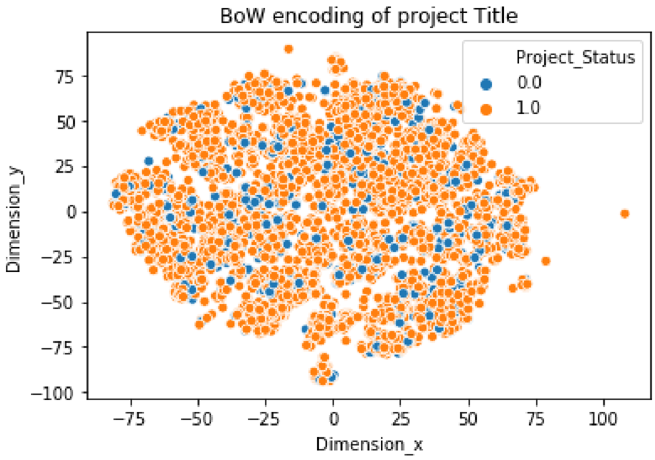
In [228]:
```
#with peplexity 500 & learning rate = 200
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z1
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=500, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("BoW encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)

plt.show()
```



Summary -
   1. Plotting with very high value of perplexity like 500 still shows no clear seperation in the plot.

In [227]:

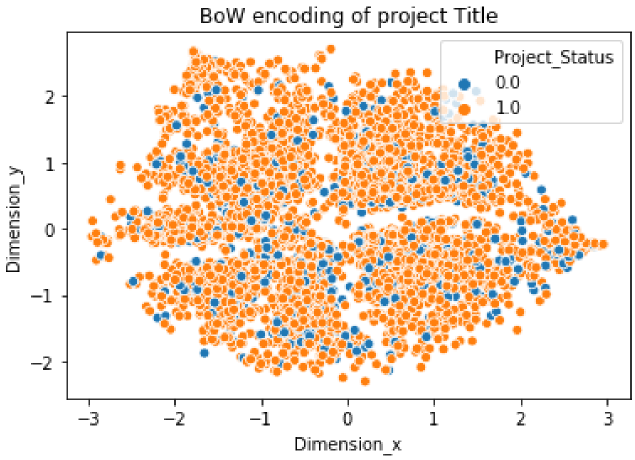
```
#with peplexity 10 & learning rate = 200
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z1
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=10, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("BoW encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)

plt.show()
```



```
Summary -
   1. The above plot shows  what is the shape of our data but there is no seperation of points & all the
points are totally
      overlapping each other.

   2. So it is very difficult to come to make any conclusion with the above plot.
```

In [246]:

```
#with 3k perplexity & learning rate 200
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

x = Z1
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=3000, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("BoW encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



In [ ]:

```
Summary -

   1. Even with lower value of perplexity the results are same with points almost spread everywhere with \
```

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

In [146]:

```
#Stacking Numerical features & Categorical features with tfidf of title
Z2 = hstack((Z, title_tfidf_6k ))
Z2.shape
```

Out[146]:

```
(6000, 550)
```

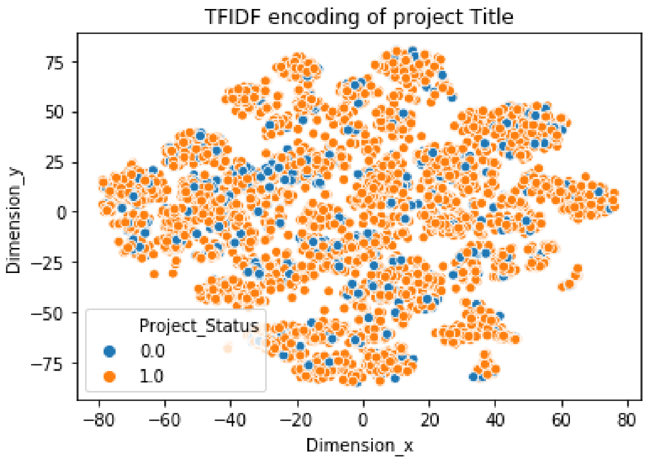
In [147]:

```
#Converting sparse matrix to dense matrix
Z2 =  Z2.toarray()
print(Z2)
```

```
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.35871538 0.         0.        ]]
```

In [310]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
#With perplexity values of 30 & learning rate of 200
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z2
y = project_6k['project_is_approved']

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

# produce a legend with the unique colors from the scatter

plt.title("TFIDF encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

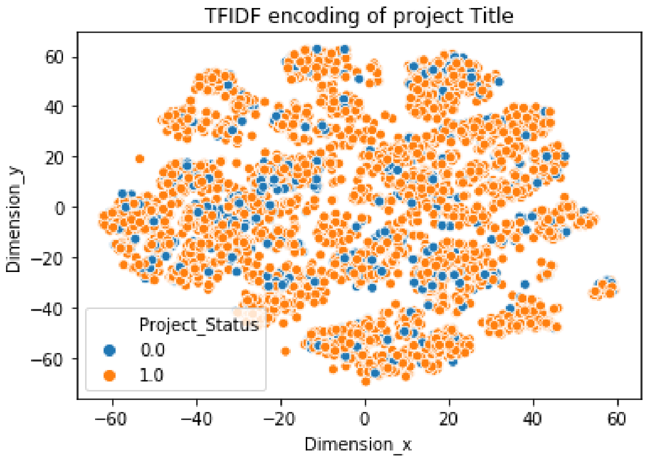
1. With TFIDF encoding plot there is some clusters of points that we can observse.

2. But even these clusters of points have intermixed points in them.

3. So we cannot interpret any conclusion from the above plot.

In [230]:

```
#with perplexity = 50 & learning rate =200

import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z2
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=50, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("TFIDF encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
# produce a legend with the unique colors from the scatter

plt.title("TFIDF encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```

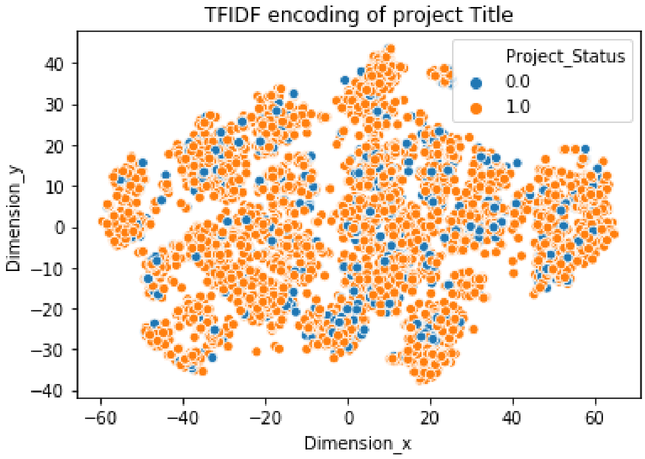


Summary -

1. With perplexity value of 50 the plot looks more or less similar & nothing can be interpreted from this plot.

In [231]:

```
#With perplexity=100 & learning rate=200

import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z2
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("TFIDF encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
# produce a legend with the unique colors from the scatter

plt.title("TFIDF encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```

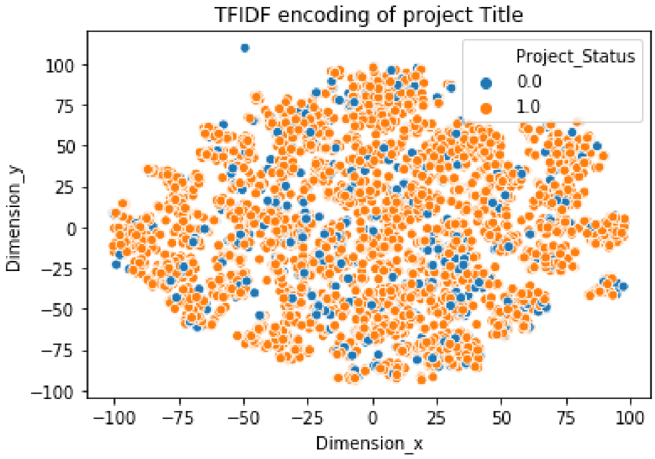


Summary -

1. Even With higher perplexity value of 100 there is no interpretability from the plot.

In [232]:

```
#With perplexity=10 & learning rate=200

import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

x = Z2
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=10, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("TFIDF encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
# produce a legend with the unique colors from the scatter

plt.title("TFIDF encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

1. With lower perplexity value of 10 the plot shows very intermixed points to deduct anything from it.

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

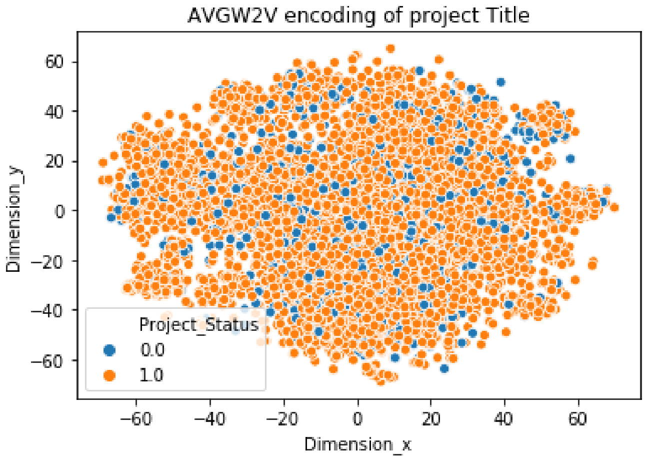
In [195]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
#One hot encoding numerical & categorical features with avgw2v vector
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
    #One hot encoding Categorical & Numerical features with AVG W2V BOW
Z3 = hstack((Z, avg_w2v_vectors_6k ))
Z3.shape
```

Out[195]:
```
(6000, 400)
```

In [197]:
```
#Converting sparse matrix to dense matrix
Z3 =  Z3.toarray()
print(Z3)
```

```
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  3.57094000e-01
   2.94482000e-01  8.56000000e-05]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  2.94676250e-01
   5.68865000e-02 -3.51785000e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  1.73265033e-01
   1.59915833e-01  8.36399500e-02]
 ...
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  1.65789667e-01
  -6.30140000e-02 -8.15696667e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  1.72703857e-01
   2.82288143e-01  4.59874286e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  2.31653000e-02
  -5.22875000e-02 -1.29356000e-02]]
```

In [233]:
```
#Applying TSNE with avgw2v encoding of project title
x = Z3
y = project_6k['project_is_approved']

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("AVGW2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



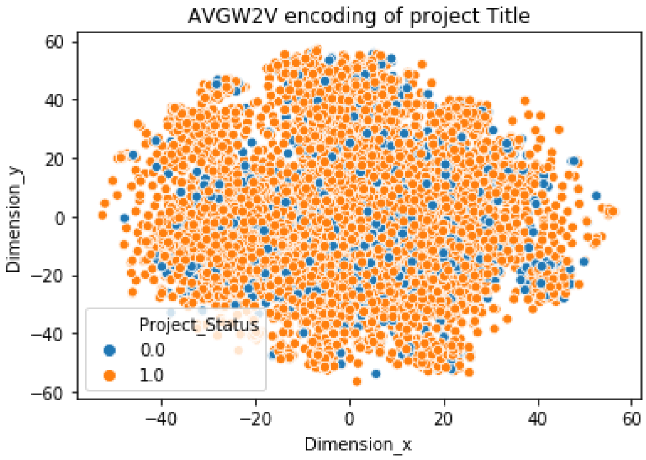
Summary -

1. With perplexity value of 30 the plot does not show any seperation.

2. It is very difficult to interpret anything from this plot.

In [234]:

```
#with perplexity =50 & learning rate = 200
x = Z3
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=50, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("AVGW2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```

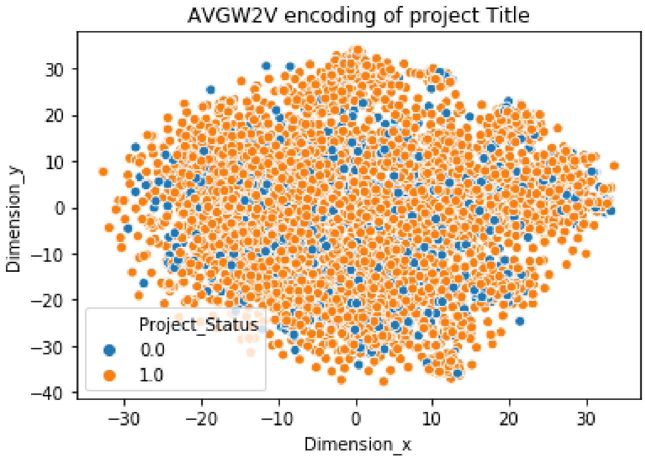


Summary -

1. With perplexity value of 50 the plot is still having intermixed points.

In [235]:

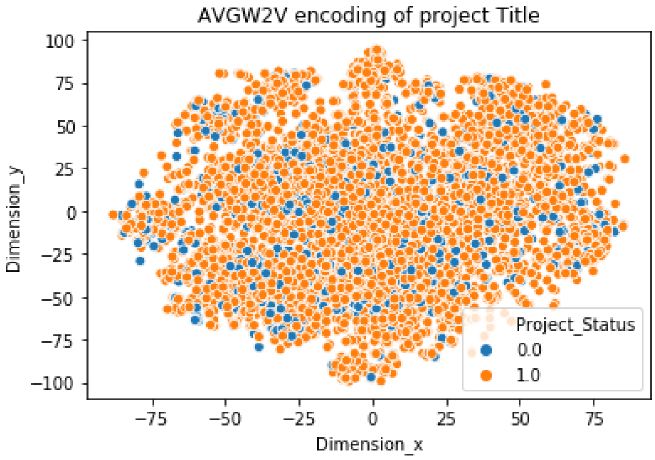
```
#with perplexity 100 & learning rate 200
x = Z3
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("AVGW2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

1. Even With higher perplexity value of 100 there is no interpretability from the plot.

In [236]:
```
#with perplexity = 10 & learning rate = 200
x = Z3
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=10, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("AVGW2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



In [ ]:
```
Summary -

  1. Even With lower value of perplexity the graph remains the same & shows no sign of seperated clusters
```

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

In [199]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
  #One hot encoding Categorical & Numerical features with TFIDF Weighted W2V
Z4 = hstack((Z, tfidf_w2v_vectors_6k ))
Z4.shape
```

Out[199]:
```
(6000, 400)
```

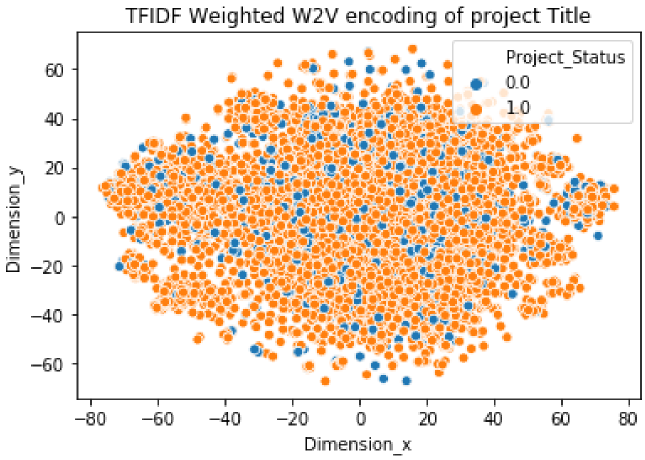
In [201]:

```
#Converting sparse matrix to dense matrix
Z4 =  Z4.toarray()
print(Z4)
```

```
[[ 0.          0.          0.         ...  0.37000468  0.2883615
   0.04542266]
 [ 0.          0.          0.         ...  0.30172159 -0.0050524
  -0.33957097]
 [ 0.          0.          0.         ...  0.134545    0.12567533
   0.09314133]
 ...
 [ 0.          0.          0.         ...  0.17403686 -0.04301274
  -0.06784444]
 [ 0.          0.          0.         ...  0.11711675  0.19292882
   0.03768657]
 [ 0.          0.          0.         ...  0.02945161 -0.09579966
  -0.02896174]]
```

In [237]:

```
#Running TSNE With perplexity 30 & learning rate = 200
x = Z4
y = project_6k['project_is_approved']

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("TFIDF Weighted W2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



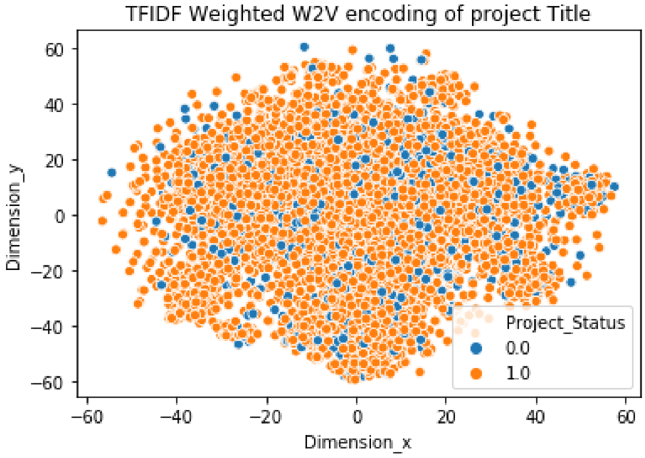
Summary -

1. TSNE with TFIDF weighted Word2Vec shows similar results as previous plots.

2. The points are scatterred all over the place with high overlapping.

In [238]:

```
#With perplexity 50 & learning rate = 200
x = Z4
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=50, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("TFIDF Weighted W2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```

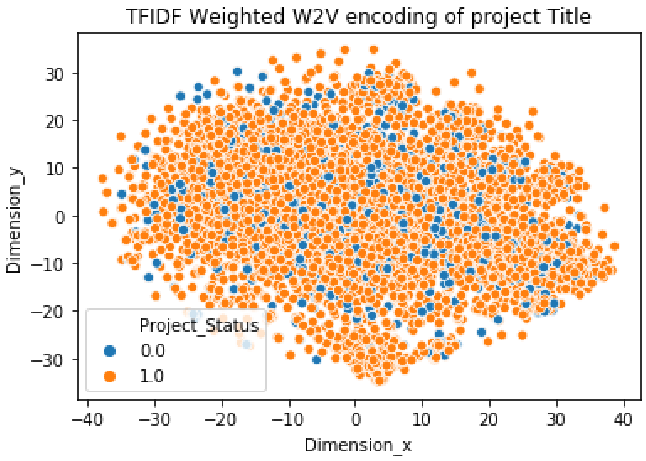


Summary -

1. With perplexity value of 50 shows very similar plot to the previous one.

2. We cannot come to any conclusion from the this plots.

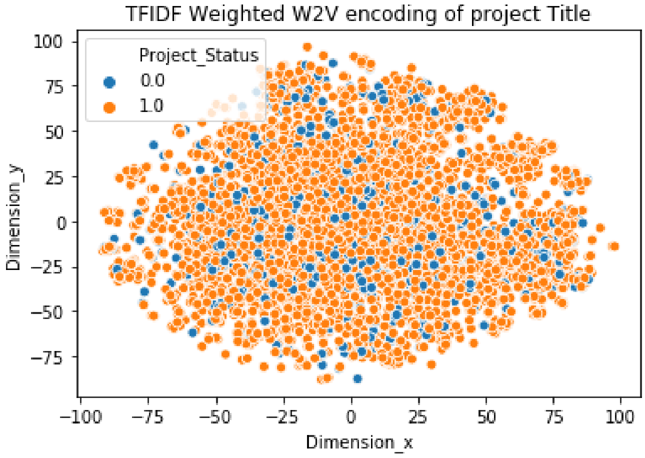
In [239]:
```
#With perplexity 100 & learning rate = 200

x = Z4
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=100, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("TFIDF Weighted W2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

1. Even With higher perplexity value of 100 there is no interpretability from the plot.

In [240]:
```
#With perplexity 10 & learning rate = 200
x = Z4
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=10, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply
plt.title("TFIDF Weighted W2V encoding of project Title")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

1. Even lowering the perplexity values to 10 shows no signs of interpretability from the plot.

2. Points are very much overlapping to each other to find any conclusion from the plot.

TSNE with all the features combined

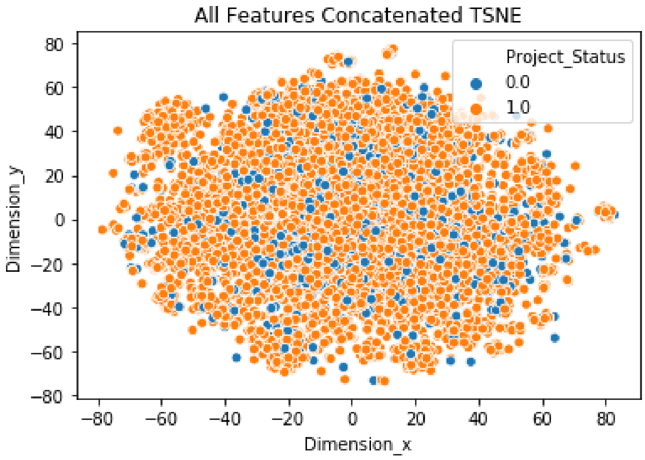
In [203]:
```
#TSNE with all the combined features
#One hot encoding all the features
Z5 = hstack((Z, title_bow_6k, title_tfidf_6k, avg_w2v_vectors_6k, tfidf_w2v_vectors_6k ))
Z5.shape
```

Out[203]:
```
(6000, 1600)
```

In [204]:
```
#converting sparse vector into dense matrix
Z5 =  Z5.toarray()
print(Z5)
```

```
[[ 0.          0.          0.         ...  0.37000468  0.2883615
   0.04542266]
 [ 0.          0.          0.         ...  0.30172159 -0.0050524
  -0.33957097]
 [ 0.          0.          0.         ...  0.134545    0.12567533
   0.09314133]
 ...
 [ 0.          0.          0.         ...  0.17403686 -0.04301274
  -0.06784444]
 [ 0.          0.          0.         ...  0.11711675  0.19292882
   0.03768657]
 [ 0.          0.          0.         ...  0.02945161 -0.09579966
  -0.02896174]]
```

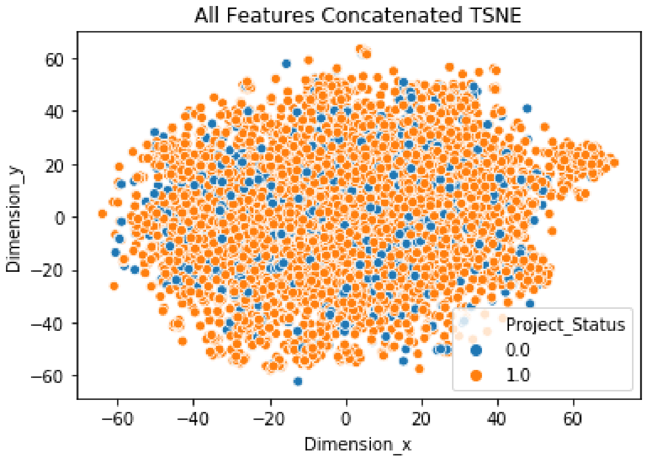
In [241]:
```
#Concatenating all the features and Apply TNSE on the final data matrix

x = Z5
y = project_6k['project_is_approved']

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

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("All Features Concatenated TSNE")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



In [ ]:
```
Summary -

   1. With all the features combined the points int the graph are still very much intermixed.

   2. It is very difficult to find any conclusion from this plot.
```

In [243]:

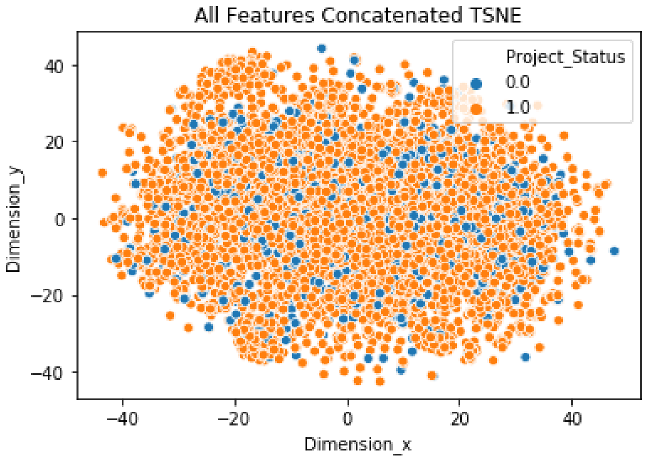
```
#with perplexity 50 & learning rate = 200

x = Z5
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=50, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("All Features Concatenated TSNE")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

1. With higher perplexity of value 50 the plot is still unpredictable.

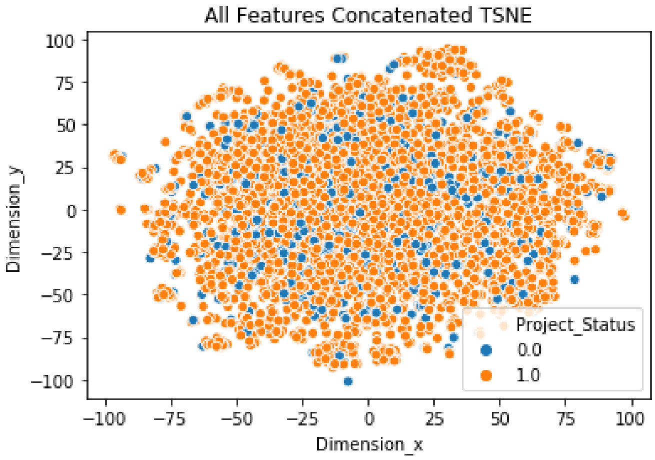
```
In [244]: #with perplexity 100 & learning rate = 200

          x = Z5
          y = project_6k['project_is_approved']

          tsne = TSNE(n_components=2, perplexity=100, learning_rate=200)

          X_embedding = tsne.fit_transform(x)
          # if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

          for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
          for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
          colors = {0:'red', 1:'blue', 2:'green'}
          #plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

          plt.title("All Features Concatenated TSNE")
          plt.xlabel("Dimension x")
          plt.ylabel("Diension y")
          for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
          for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
          for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
          ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
          plt.show()
```



```
In [ ]: Summary -

          1. Even changing the perplexity to 100 we don not observe much change in the plot so the plot is still
             any insight.
```

In [245]:

```
#with perplexity 10 & learning rate = 200

x = Z5
y = project_6k['project_is_approved']

tsne = TSNE(n_components=2, perplexity=10, learning_rate=200)

X_embedding = tsne.fit_transform(x)
# if x is a sparse matrix you need to pass it as X_embedding = tsne.fit_transform(x.toarray()) , .toarray(

for_tsne = np.hstack((X_embedding, y.values.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Project_Status'])
colors = {0:'red', 1:'blue', 2:'green'}
#plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'], c=for_tsne_df['Project_Status'].apply

plt.title("All Features Concatenated TSNE")
plt.xlabel("Dimension x")
plt.ylabel("Diension y")
for_tsne_df.Dimension_x=for_tsne_df.Dimension_x.astype('float')
for_tsne_df.Dimension_y=for_tsne_df.Dimension_y.astype('float')
for_tsne_df.Project_Status=for_tsne_df.Project_Status.astype('category')
ax = sns.scatterplot(x="Dimension_x", y="Dimension_y", hue="Project_Status", data=for_tsne_df)
plt.show()
```



Summary -

1. Even with lower value of perplexity the plot is pretty much unpredictable & does not provide any insight.

2.5 Summary

In [ ]:

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

Observations :-

1. The data points in all the above plots are widely spread.

2. Points from the both the classes are widely spread & very much overlapping into each other.

3. Also the number of Not approved projects are much less if we compare them to approved projects.

4. From this we can conclude that the dataset is imbalanced.

5. We have also observed that considering first 6000 data points all the TSNE plot are very much overlapping or intermixed
with each other.

6. The data points remain very much overlapping even with various values of perplexity.

7. Higher or lower perplexity is not affecting the plots or it does not show any signs of cluster forming or seperation between
the different types of data points.

```
8. We just get a approximate idea of the local structure of the points.

9. With such high overlapping of points after considering so many different combination of features &
values of perplexity
   it is difficult to arrive at any conclusion or finding any insights from the plots.
```