prabhudayala@gmail.com_2

April 2, 2019

1 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

How to scale current manual processes and resources to screen 500,000 projects so that they ca
<li>How to increase the consistency of project vetting across different volunteers to improve
<li>How to focus volunteer time on the applications that need the most assistance</li>
</ul>
```

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.

1.1 About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`

`project_title` | Title of the project. **Examples:**
Art Will Make You Happy!
First Grade Fun
`project_grade_category` | Grade level of students for which the project is targeted. One of the following enumerated values:
Grades PreK-2
Grades 3-5
Grades 6-8
Grades 9-12
`project_subject_categories` | One or more (comma-separated) subject categories for the project from the following enumerated list of values:

Applied Learning
Care & Hunger
Health & Sports
History & Civics
Literacy & Language
Math & Science
Music & The Arts
Special Needs
Warmth
Examples:
Music & The Arts
Literacy & Language, Math & Science

`school_state` | State where school is located (Two-letter U.S. postal code). **Example:** `WY`

`project_subject_subcategories` | One or more (comma-separated) subject subcategories for the project. **Examples:**

Literacy
Literature & Writing, Social Sciences

`project_resource_summary` | An explanation of the resources needed for the project. **Example:**

My students need hands on literacy materials to manage sensory needs!

`project_essay_1` | First application essay

`project_essay_2` | *Second application essay* `project_essay_3` | Third application essay ***`project_essay_4`*** | *Fourth application essay* `project_submitted_datetime` | Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`

`teacher_id` | A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`

`teacher_prefix` | Teacher's title. One of the following enumerated values:

nan
Dr.
Mr.
Mrs.
Ms.
Teacher.

`teacher_number_of_previously_posted_projects` | Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`

Feature	Description
`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.

1.1.1 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 [28]: %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
```

1.2 1.1 Reading Data

```
In [29]: project_data = pd.read_csv('train_data.csv')
         resource_data = pd.read_csv('resources.csv')
```

```
In [30]: 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 [31]: 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[31]:         id                                           description  quantity  \
         0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
         1  p069063         Bouncy Bands for Desks (Blue support pipes)         3

             price
         0  149.00
         1   14.95
```

```
In [32]: print("avarage resourses needed for a project is: ",resource_data.shape[0]/project_dat
```

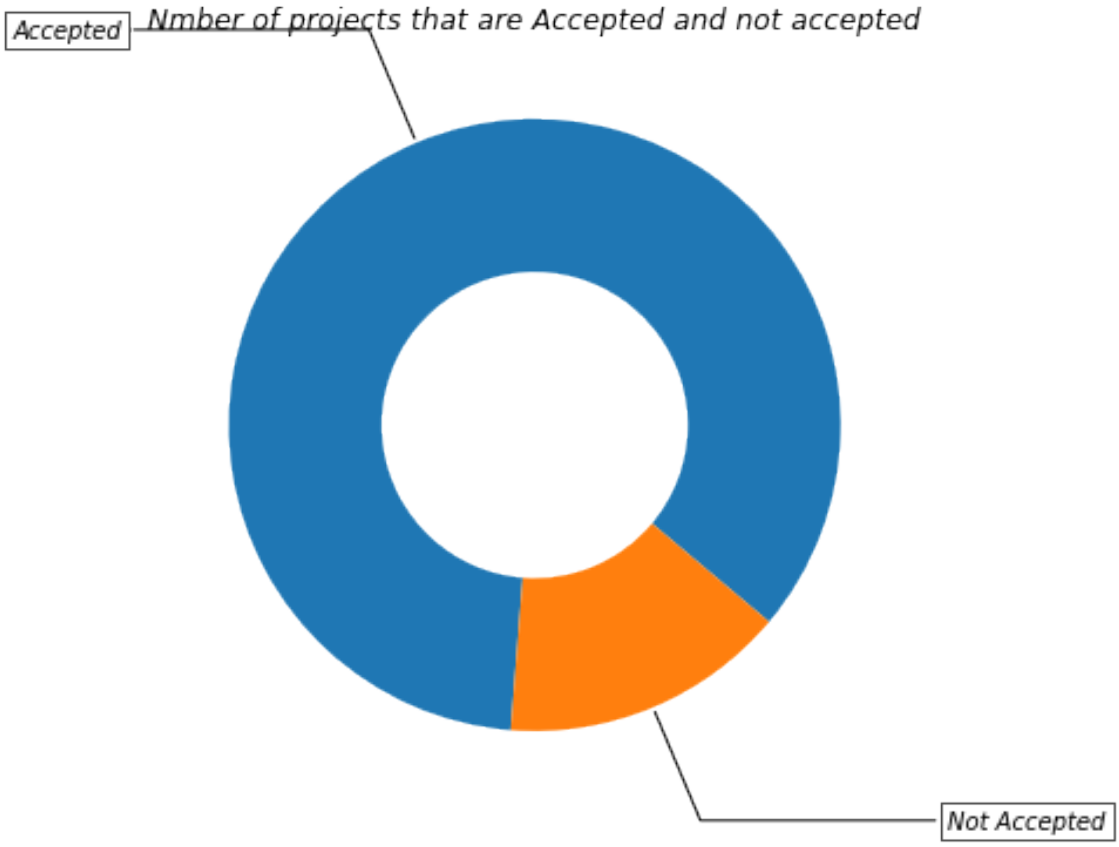
```
avarage resourses needed for a project is:  14.108011130638547
```

2 1.2 Data Analysis

```
In [33]: # 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-

         y_value_counts = project_data['project_is_approved'].value_counts()
         print("Number of projects thar are approved for funding ", y_value_counts[1], ", (",
         print("Number of projects thar are not approved for funding ", 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("Nmber of projects that are Accepted and not accepted")

        plt.show()
```

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



Observation 1: 84.85 % of submited projects are approved.

2.0.1 1.2.1 Univariate Analysis: School State

In [34]:
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].apply
# if you have data which contain only 0 and 1, then the mean = percentage (think abou
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 2: Country with code DE which is Delaware has maximum project acceptance rate.

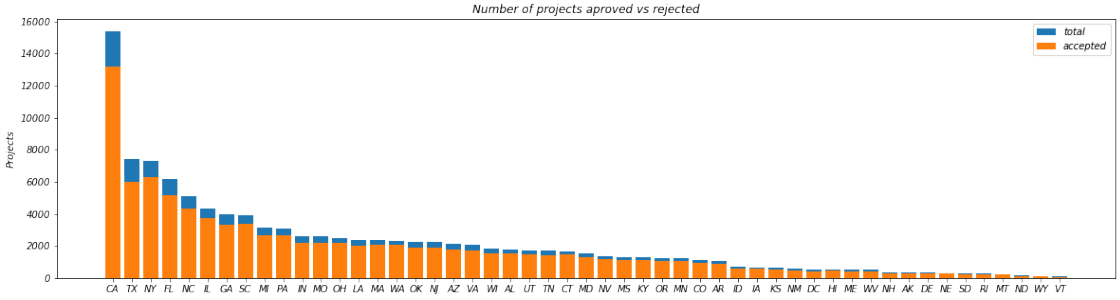
In [35]:
```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
print('='*50)
print("States with highest % approvals")
print(temp.tail(5))
```

```
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 [36]:
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/
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 [37]:
```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/
    temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum()))

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

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



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

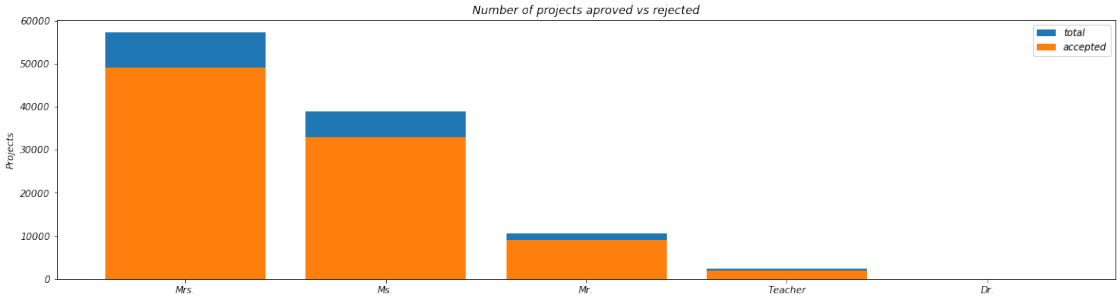
==========

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

Observation 3: Every state has greater than 80% success rate in approval

2.0.2 1.2.2 Univariate Analysis: teacher_prefix

```
In [39]: univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)
```

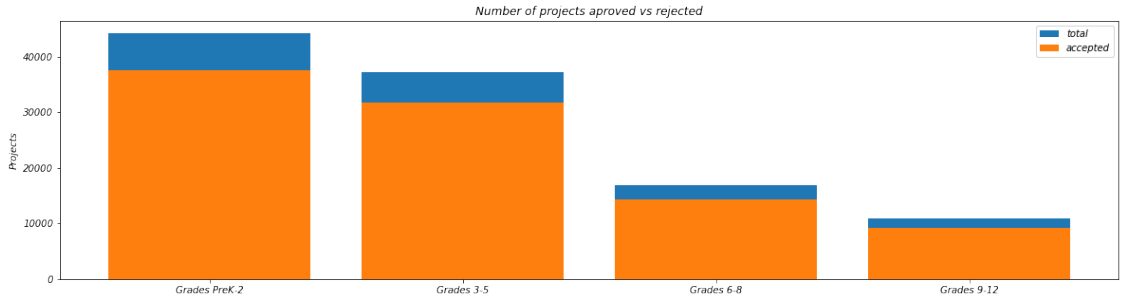


```
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.     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

Observation 4: Teacher who is having Mrs. as prefix have the maximum average of project acceptance. Teacher who is having Dr. as prefix have the minimum average of project acceptance.

2.0.3 1.2.3 Univariate Analysis: project_grade_category

```
In [41]: univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', to
```




```
  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
```

Observation 5: Heighest number of projects are submitted for Grades PreK-2 and accepted too.

Observation 6: Project submition and acceptance rate both decreases on increase of grade.

2.0.4 1.2.4 Univariate Analysis: project_subject_categories

In [42]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-st
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-py
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", "Warm
        if 'The' in j.split(): # this will split each of the catogory based on space
            j=j.replace('The','') # if we have the words "The" we are going to replac
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing sp
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

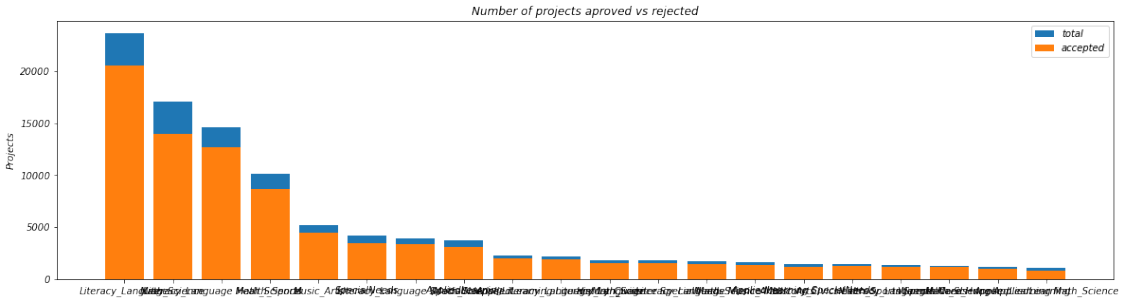
In [43]:
```
#print(cat_list[0])
#print(cat_list[1])
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[43]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_subject_subcategories	project_title	project_essay_1	project_essay_2	project_essay_3
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	ESL, Literacy	Educational Support for English Learners at Home	My students are English learners that are work...	\"The limits of your language are the limits o...	NaN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Civics & Government, Team Sports	Wanted: Projector for Hungry Learners	Our students arrive to our school eager to lea...	The projector we need for our school is very c...	NaN

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

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0
1                                             7                    1

              clean_categories
0            Literacy_Language
1  History_Civics Health_Sports
```

In [44]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=20)
```

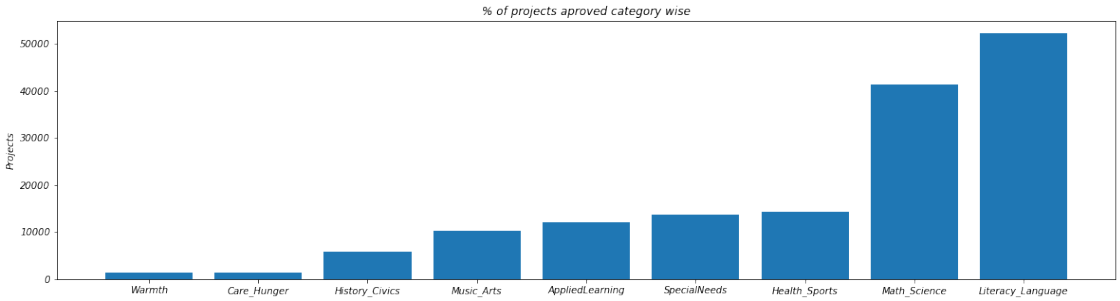



```
                 clean_categories  project_is_approved  total       Avg
24                Literacy_Language                20520  23655  0.867470
32                     Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12725  14636  0.869432
8  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

Observation 7: Projects with category Warmth Care_Hunger tend to have more acceptance rate where as Math_Science projects have approval rate as minimum.

In [46]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/40840
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

```
In [50]: # 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 [51]: 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
```

Observation 8: As we have observed that one project can belong to multiple category, After getting projects sumitted in each category I found that Literacy_Language category has maximum number of project submited.

2.0.5 1.2.5 Univariate Analysis: project_subject_subcategories

```
In [52]: sub_catogories = list(project_data['project_subject_subcategories'].values)
         # remove special characters from list of strings python: https://stackoverflow.com/a/
```

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

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", "Warm
        if 'The' in j.split(): # this will split each of the catogory based on space
            j=j.replace('The','') # if we have the words "The" we are going to replac
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing s
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [53]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[53]:

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

	school_state	project_submitted_datetime	project_grade_category	\
0	IN	2016-12-05 13:43:57	Grades PreK-2	
1	FL	2016-10-25 09:22:10	Grades 6-8	

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

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

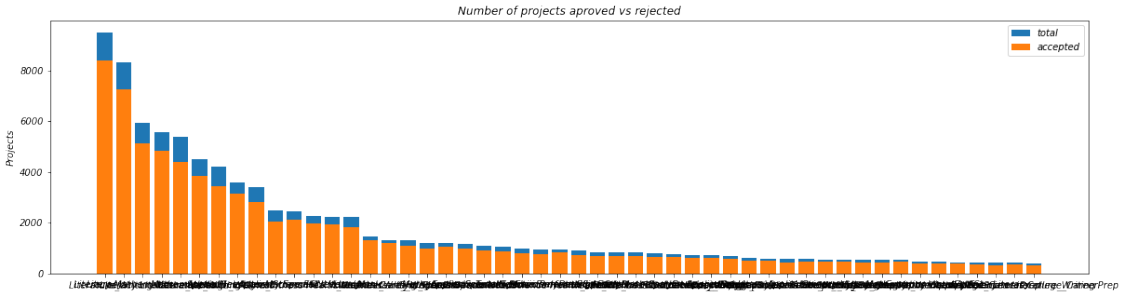
	project_essay_2	project_essay_3	\
0	\"The limits of your language are the limits o...	NaN	
1	The projector we need for our school is very c...	NaN	

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

	teacher_number_of_previously_posted_projects	project_is_approved	\
0	0	0	
1	7	1	

```
                clean_categories           clean_subcategories
0             Literacy_Language                   ESL Literacy
1  History_Civics Health_Sports  Civics_Government TeamSports
```

```
In [54]: univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top=5
```



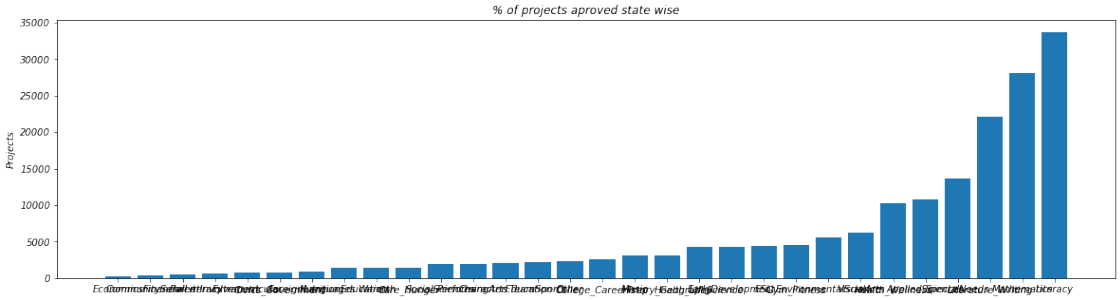

```
                clean_subcategories  project_is_approved  total       Avg
317                         Literacy                 8371   9486  0.882458
319            Literacy Mathematics                 7260   8325  0.872072
331  Literature_Writing Mathematics                 5140   5923  0.867803
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
```

Observation 8: For all project sub category the acceptance rate is more than 80% and projects belonging to this mix category AppliedSciences College_CareerPrep has the lowest rate of average approval (81.48%).

```
In [56]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/40840
         from collections import Counter
         my_counter = Counter()
         for word in project_data['clean_subcategories'].values:
             my_counter.update(word.split())
```

```
In [57]: # 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 [58]: 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
```

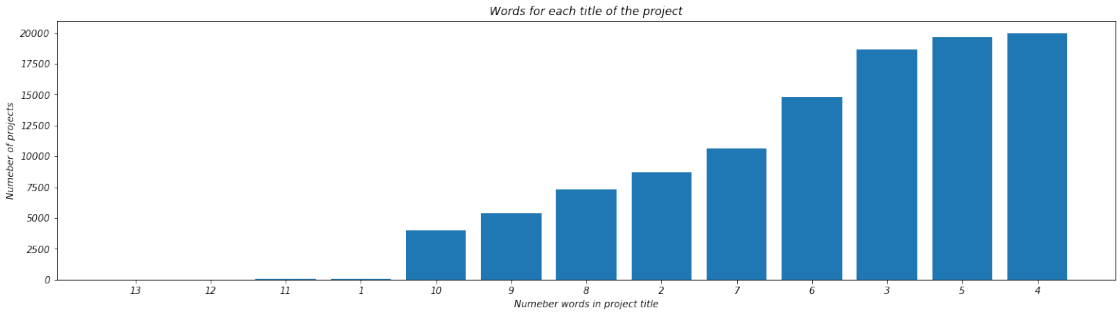
```
VisualArts           :      6278
Health_Wellness      :     10234
AppliedSciences      :     10816
SpecialNeeds         :     13642
Literature_Writing   :     22179
Mathematics          :     28074
Literacy             :     33700
```

Observation 9: Maximum number of projects (33700) belong to sub category Literacy.

2.0.6 1.2.6 Univariate Analysis: Text features (Title)

```
In [59]: #How to calculate number of words in a string in DataFrame: https://stackoverflow.com
         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))
         print(ind)
         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()
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12]
```

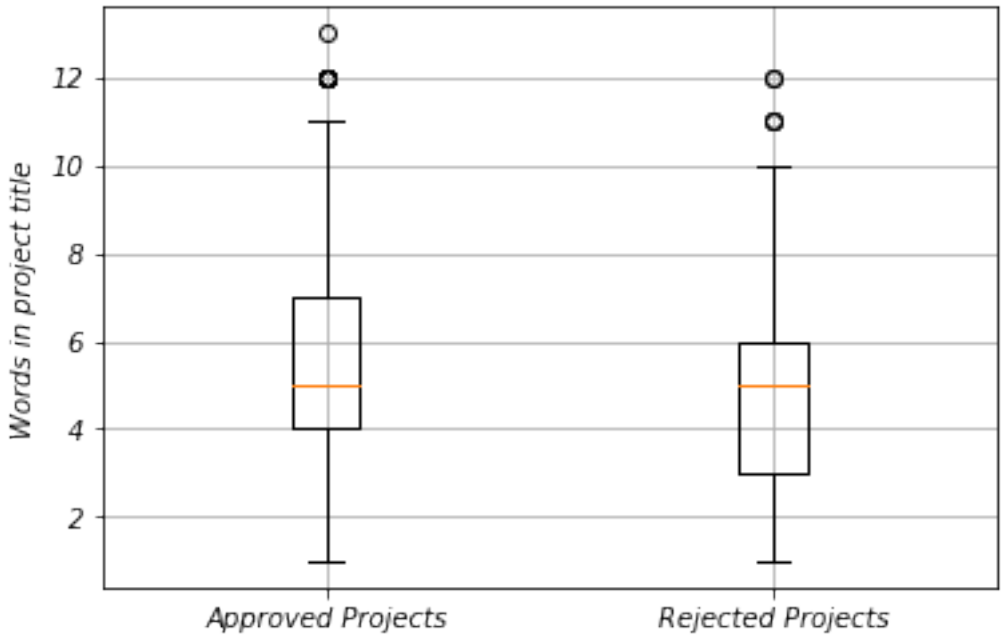


Observation 9: Maximum project title consists of 3 or 4 or 5 words.

```
In [60]: approved_title_word_count = project_data[project_data['project_is_approved']==1]['pro
         approved_title_word_count = approved_title_word_count.values
```

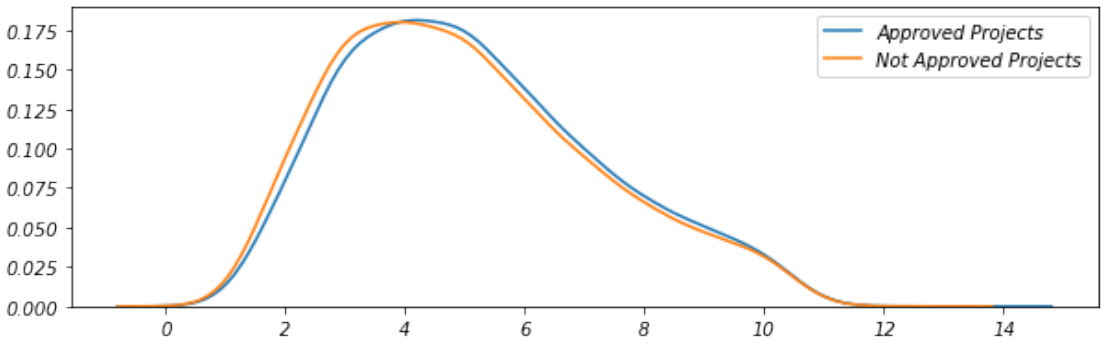
```
rejected_title_word_count = project_data[project_data['project_is_approved']==0]['proj
rejected_title_word_count = rejected_title_word_count.values
```

In [61]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



Observation 10: For approved projects it seems that there is a small increase in number of words.

In [62]:
```
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()
```

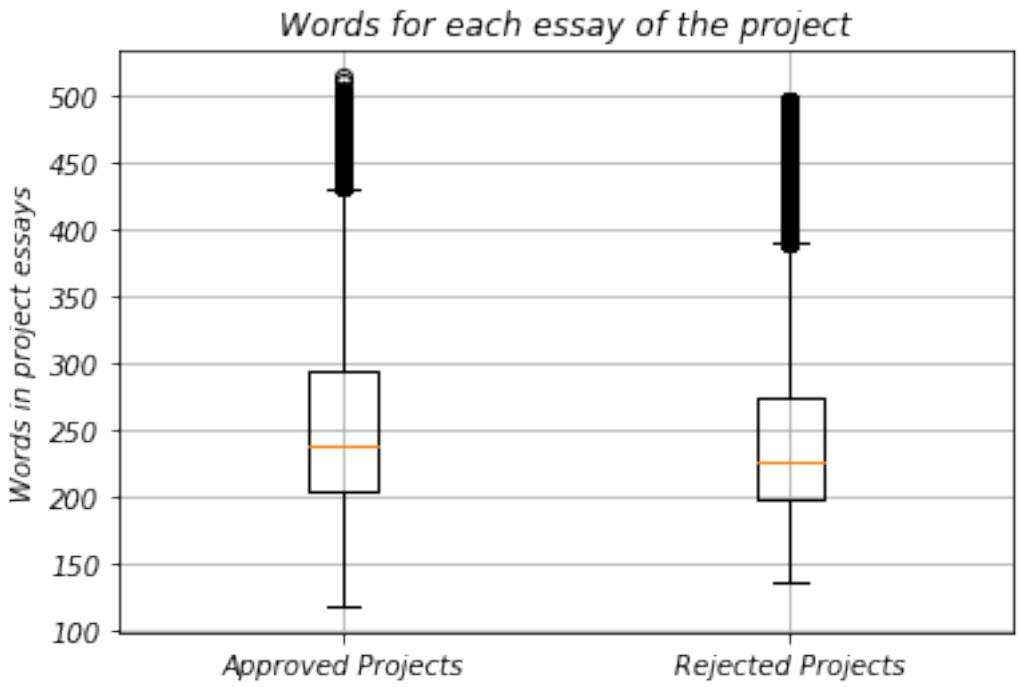


2.0.7 1.2.7 Univariate Analysis: Text features (Project Essay's)

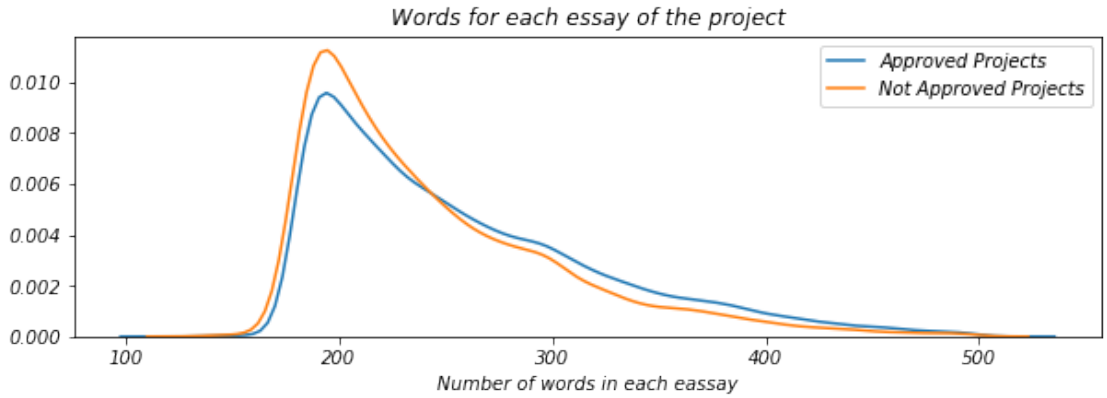
In [63]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [64]:
```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].st
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0]['essay'].st
rejected_word_count = rejected_word_count.values
```

In [65]:
```
# 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 [66]: 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()
```



Observation 11: Number of words in essay does not have much impact on project acceptance.

2.0.8 1.2.8 Univariate Analysis: Cost per project

In [67]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[67]:

	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 [68]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_
price_data.head(2)
```

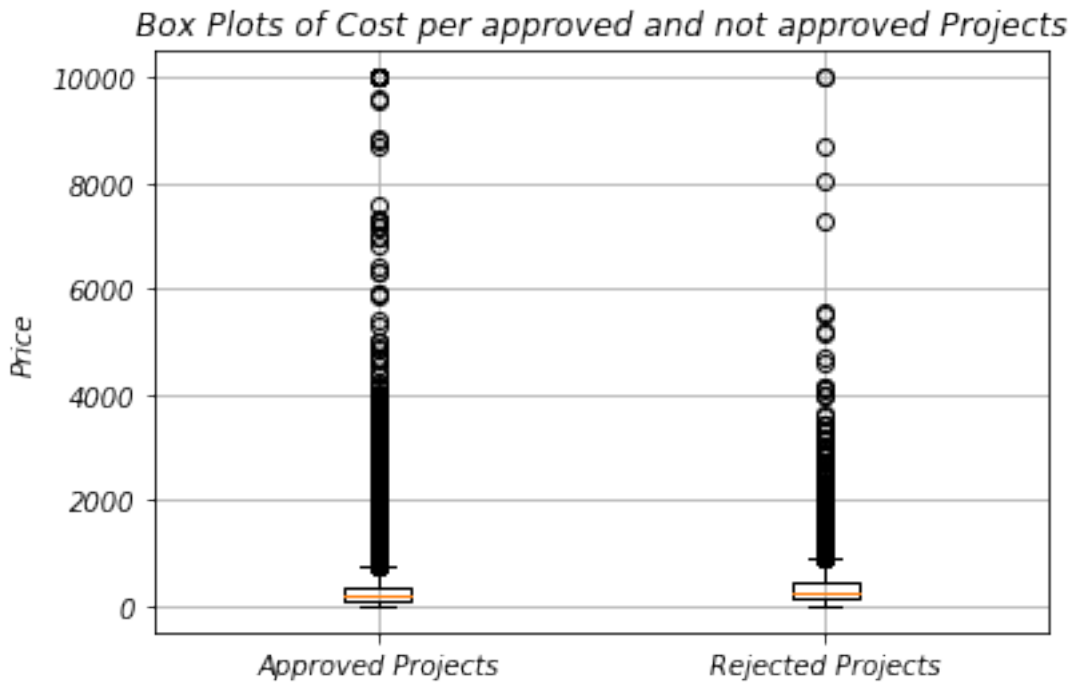
Out[68]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

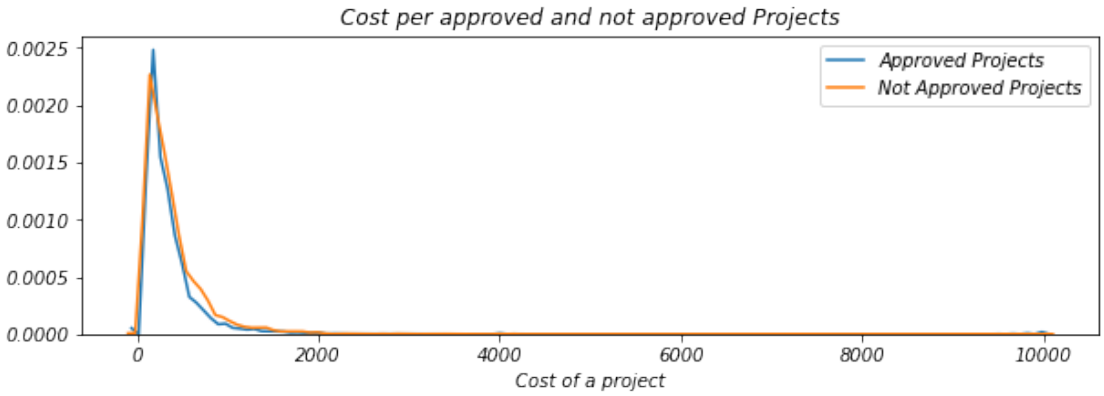
In [69]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [70]:
```
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 [71]:
```
# 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 [72]:
```
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 [40]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
```

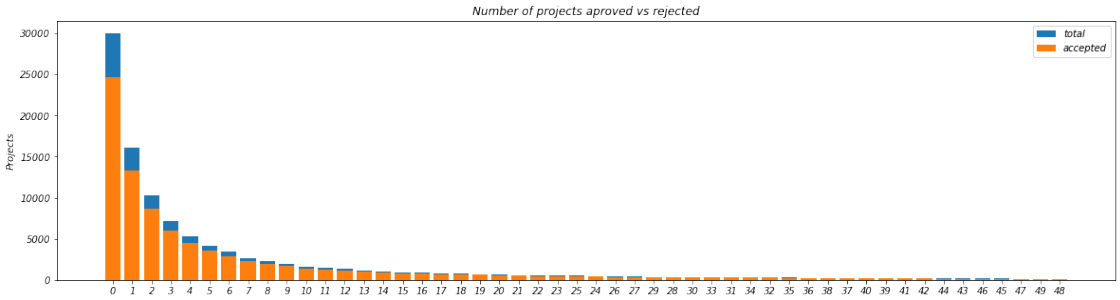
```
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install pre

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
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

Observation 12 : Though there is a small difference in each percentile of accepted and rejected project, it does not seem to varry enough to differenciate between acceptance and rejection of a project.

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

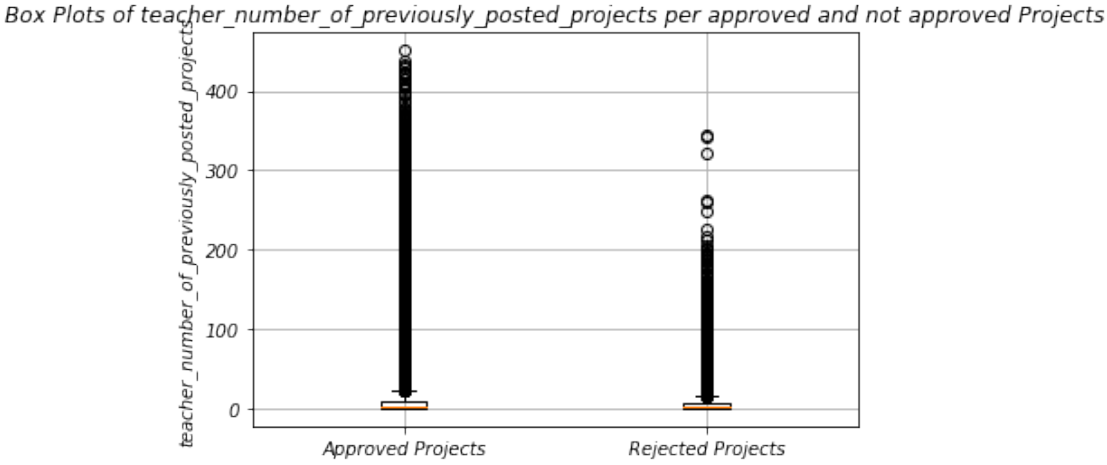
```
In [73]: univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects', 'pr
```



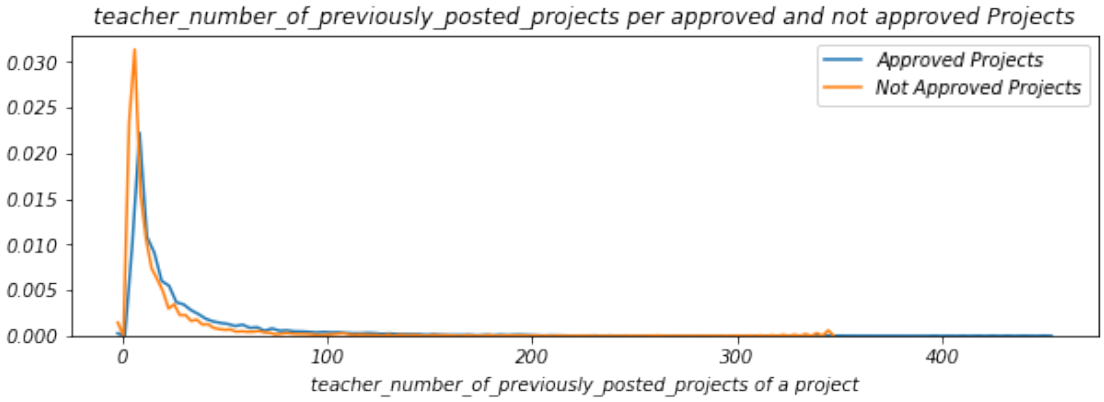

```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                24652  30014
1                                             1                13329  16058
2                                             2                 8705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

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

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

```
In [74]: approved_teacher_number_of_previously_posted_projects = project_data[project_data['pro

         rejected_teacher_number_of_previously_posted_projects = project_data[project_data['pro

In [75]: plt.boxplot([approved_teacher_number_of_previously_posted_projects, rejected_teacher_n
         plt.title('Box Plots of teacher_number_of_previously_posted_projects per approved and
         plt.xticks([1,2],('Approved Projects','Rejected Projects'))
```

```
plt.ylabel('teacher_number_of_previously_posted_projects')
plt.grid()
plt.show()
```



In [76]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_teacher_number_of_previously_posted_projects, hist=False, label=
sns.distplot(rejected_teacher_number_of_previously_posted_projects, hist=False, label=
plt.title('teacher_number_of_previously_posted_projects per approved and not approved
plt.xlabel('teacher_number_of_previously_posted_projects of a project')
plt.legend()
plt.show()
```



In [77]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

```
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_teacher_number_of_previously_posted_p
print(x)
```

```
+------------+-------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+-------------------+-----------------------+
|     0      |        0.0        |          0.0          |
|     5      |        0.0        |          0.0          |
|     10     |        0.0        |          0.0          |
|     15     |        0.0        |          0.0          |
|     20     |        0.0        |          0.0          |
|     25     |        0.0        |          0.0          |
|     30     |        1.0        |          0.0          |
|     35     |        1.0        |          1.0          |
|     40     |        1.0        |          1.0          |
|     45     |        2.0        |          1.0          |
|     50     |        2.0        |          2.0          |
|     55     |        3.0        |          2.0          |
|     60     |        4.0        |          3.0          |
|     65     |        5.0        |          3.0          |
|     70     |        7.0        |          4.0          |
|     75     |        9.0        |          6.0          |
|     80     |        13.0       |          8.0          |
|     85     |        19.0       |          11.0         |
|     90     |        30.0       |          17.0         |
|     95     |        57.0       |          31.0         |
|    100     |       451.0       |         345.0         |
+------------+-------------------+-----------------------+
```

Observation 13: The fact whether the teacher has any previously submited projects by the company really does not impact on the aceeptance of a new one

Observation 14: Almost half of the teachers have only 2 previously submited 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.

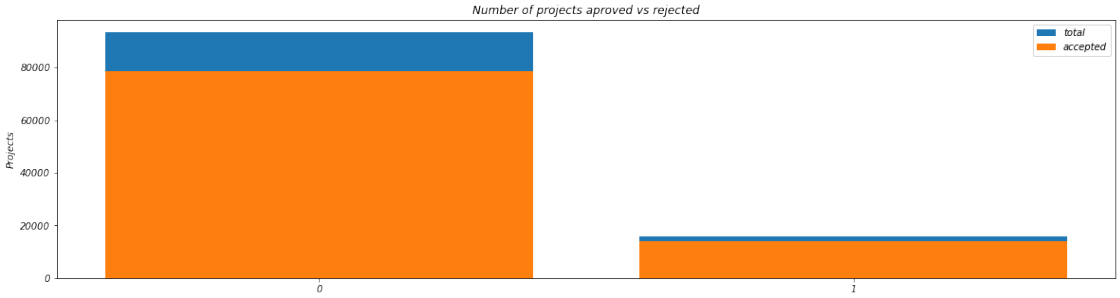
```
In [83]: def hasNumbers(inputString):
             value = bool(re.search(r'\d', inputString))
             if(value):
                 return 1
```

```
            else:
                return 0
        project_data['is_digit_present_in_project_summary']=project_data.project_resource_summ
```

In [84]:
```
project_data.is_digit_present_in_project_summary.value_counts()
```

Out[84]:
```
0    93492
1    15756
Name: is_digit_present_in_project_summary, dtype: int64
```

In [85]:
```
univariate_barplots(project_data, 'is_digit_present_in_project_summary', 'project_is_a
```



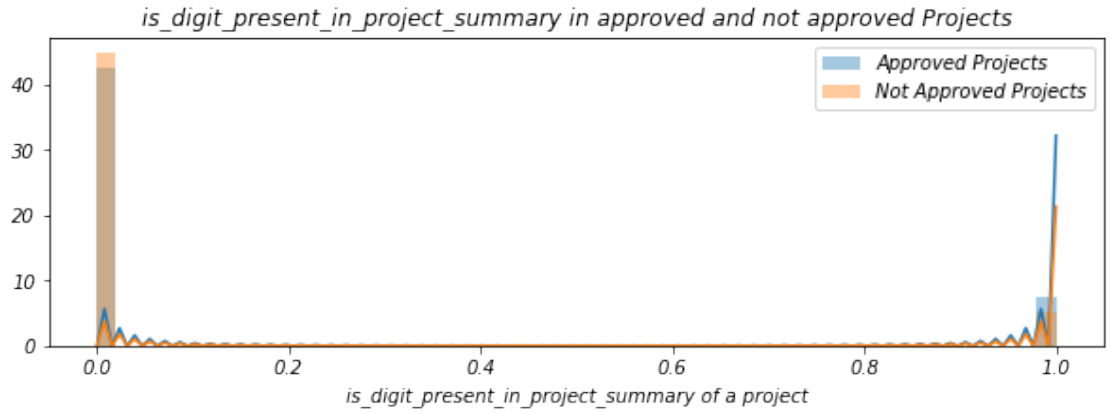
	is_digit_present_in_project_summary	project_is_approved	total	Avg
0	0	78616	93492	0.840885
1	1	14090	15756	0.894263

==========

	is_digit_present_in_project_summary	project_is_approved	total	Avg
0	0	78616	93492	0.840885
1	1	14090	15756	0.894263

In [86]:
```
approved_is_digit_present_in_project_summary = project_data[project_data['project_is_a

        rejected_is_digit_present_in_project_summary = project_data[project_data['project_is_a
```

In [87]:
```
plt.figure(figsize=(10,3))
        sns.distplot(approved_is_digit_present_in_project_summary, hist=True, label="Approved
        sns.distplot(rejected_is_digit_present_in_project_summary, hist=True, label="Not Appr
        plt.title('is_digit_present_in_project_summary in approved and not approved Projects'
        plt.xlabel('is_digit_present_in_project_summary of a project')
        plt.legend()
        plt.show()
```



Observation: Presence of numericals in project_resource_summary does not affect the acceptance much.

2.1 1.3 Text preprocessing

2.1.1 1.3.1 Essay Text

```
In [88]: project_data.head(2)
```

```
Out[88]:    Unnamed: 0       id                         teacher_id teacher_prefix  \
         0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
         1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.

           school_state project_submitted_datetime project_grade_category  \
         0           IN        2016-12-05 13:43:57          Grades PreK-2
         1           FL        2016-10-25 09:22:10             Grades 6-8

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

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

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

                        ...                  project_essay_4  \
         0              ...                              NaN
         1              ...                              NaN

                                 project_resource_summary  \
```

```
0  My students need opportunities to practice beg...
1  My students need a projector to help with view...

  teacher_number_of_previously_posted_projects  project_is_approved  \
0                                            0                   0
1                                            7                   1

               clean_categories           clean_subcategories  \
0             Literacy_Language                   ESL Literacy
1  History_Civics Health_Sports  Civics_Government TeamSports

                                                 essay  price  quantity  \
0  My students are English learners that are work...  154.6        23
1  Our students arrive to our school eager to lea...  299.0         1

   is_digit_present_in_project_summary
0                                    0
1                                    0

[2 rows x 21 columns]
```

In [89]:
```
# 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
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, a
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogn
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The g
==================================================
```

In [90]:
```
# 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 [91]: sent = decontracted(project_data['essay'].values[20000])
         print(sent)
         print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogn
==================================================
```

```
In [92]: # \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-p
         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, cogn
```

```
In [93]: #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 cogni
```

```
In [94]: # 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'
                     "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him'
                     'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
                     'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "
                     'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', '
                     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
                     'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'throug
```

```
                    'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', '
                    'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'a
                    'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'to
                    's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", '
                    've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't
                    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mi
                    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
                    'won', "won't", 'wouldn', "wouldn't"]
```

```
In [95]: # 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 [00:44<00:00, 2470.86it/s]
```

```
In [96]: # after preprocesing
         preprocessed_essays[20000]
```

```
Out[96]: 'my kindergarten students varied disabilities ranging speech language delays cognitive
```

```
In [97]: print(project_data['essay'].values[0])
         project_data['essay']=preprocessed_essays
         print("*"*50)
         print(project_data['essay'].values[0])
```

```
My students are English learners that are working on English as their second or third languages
**************************************************
my students english learners working english second third languages we melting pot refugees imm
```

1.3.2 Project title Text

```
In [98]: from tqdm import tqdm
         preprocessed_project_title = []
         # 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_project_title.append(sent.lower().strip())
```

```
100%|| 109248/109248 [00:01<00:00, 56400.03it/s]
```

```
In [99]: print(project_data['project_title'].values[20001])
         project_data['project_title']=preprocessed_project_title
         print(project_data['project_title'].values[20001])
```

```
The Beautiful Life of a Butterfly
the beautiful life butterfly
```

2.2 1. 4 Preparing data for models

```
In [100]: project_data.columns
```

```
Out[100]: 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',
                 'is_digit_present_in_project_summary'],
                dtype='object')
```

we are going to consider

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

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

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

2.2.1 1.4.1 Vectorizing Categorical data

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

```
In [101]: # 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=Fals
          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', '
Shape of matrix after one hot encodig  (109248, 9)
```

```
In [102]: # we use count vectorizer to convert the values into one hot encoded features
          vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=
          vectorizer.fit(project_data['clean_subcategories'].values)
          print(vectorizer.get_feature_names())


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
Shape of matrix after one hot encodig  (109248, 30)
```

```
In [103]: # we use count vectorizer to convert the values into one hot encoded features
          vectorizer = CountVectorizer(vocabulary=list(project_data['school_state'].unique()),
          vectorizer.fit(project_data['school_state'].values)
          print(vectorizer.get_feature_names())


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

```
['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY', 'OK', 'MA', 'NV', 'OH', 'PA
Shape of matrix after one hot encodig  (109248, 51)
```

```
In [104]: project_data.teacher_prefix = project_data.teacher_prefix.replace(np.nan, '', regex=
          print(project_data.teacher_prefix.value_counts())
```

```
Mrs.       57269
Ms.        38955
```

```
Mr.        10648
Teacher     2360
Dr.           13
               3
Name: teacher_prefix, dtype: int64
```

```
In [105]: # we use count vectorizer to convert the values into one hot encoded features
          vectorizer = CountVectorizer(vocabulary=list(project_data['teacher_prefix'].unique()),
          vectorizer.fit(project_data['teacher_prefix'].values)
          print(vectorizer.get_feature_names())
          teacher_prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].values)
          print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
```

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

```
In [106]: # we use count vectorizer to convert the values into one hot encoded features
          vectorizer = CountVectorizer(vocabulary=list(project_data['project_grade_category'].u
          vectorizer.fit(project_data['project_grade_category'].values)
          print(vectorizer.get_feature_names())
          project_grade_category_one_hot = vectorizer.transform(project_data['project_grade_cat
          print("Shape of matrix after one hot encodig ",project_grade_category_one_hot.shape)
```

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

2.2.2 1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

```
In [107]: # We are considering only the words which appeared in at least 10 documents(rows or
          vectorizer = CountVectorizer(min_df=10)
          text_bow = vectorizer.fit_transform(preprocessed_essays)
          print("Shape of matrix after one hot encodig ",text_bow.shape)
```

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

1.4.2.2 Bag of Words on `project_title`

```
In [108]: # you can vectorize the title also
          # before you vectorize the title make sure you preprocess it
```

```
In [109]: # Similarly you can vectorize for title also
```

```
In [110]: # We are considering only the words which appeared in at least 10 documents(rows or
          vectorizer = CountVectorizer(min_df=10)
          title_text_bow = vectorizer.fit_transform(preprocessed_project_title)
          print("Shape of matrix after one hot encodig ",title_text_bow.shape)
```

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

1.4.2.3 TFIDF vectorizer

```
In [111]: 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 [112]: # Similarly you can vectorize for title also
          from sklearn.feature_extraction.text import TfidfVectorizer
          vectorizer = TfidfVectorizer(min_df=10)
          title_text_tfidf = vectorizer.fit_transform(preprocessed_project_title)
          print("Shape of matrix after one hot encodig ",title_text_tfidf.shape)
```

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

1.4.2.5 Using Pretrained Models: Avg W2V

```
In [113]: # 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
          # borrowed from https://therenegadecoder.com/code/how-to-check-if-a-file-exists-in-p
          import os
          exists = os.path.isfile('./glove_vectors')
          if(not exists):
              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-t

            import pickle
            with open('glove_vectors', 'wb') as f:
                pickle.dump(words_courpus, f)
        else:
            print("glove already exists. No need to compute")
```

```
glove already exists. No need to compute
```

```
In [114]: # stronging variables into pickle files python: http://www.jessicayung.com/how-to-us
          # 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 [115]: # average Word2Vec
          # compute average word2vec for each review.
          avg_w2v_vectors_essay = []; # the avg-w2v for each sentence/review is stored in this
          for sentence in tqdm(preprocessed_essays): # for each review/sentence
              vector = np.zeros(300) # as word vectors are of zero length
```

```
            cnt_words =0; # num of words with a valid vector in the sentence/review
            for word in sentence.split(): # for each word in a review/sentence
                if word in glove_words:
                    vector += model[word]
                    cnt_words += 1
            if cnt_words != 0:
                vector /= cnt_words
            avg_w2v_vectors_essay.append(vector)

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

```
100%|| 109248/109248 [00:23<00:00, 4735.21it/s]


109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

```
In [116]: # average Word2Vec
          # compute average word2vec for each title.
          title_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this
          for sentence in tqdm(preprocessed_project_title): # for each review/sentence
              vector = np.zeros(300) # as word vectors are of zero length
              cnt_words =0; # num of words with a valid vector in the sentence/review
              for word in sentence.split(): # for each word in a review/sentence
                  if word in glove_words:
                      vector += model[word]
                      cnt_words += 1
              if cnt_words != 0:
                  vector /= cnt_words
              title_avg_w2v_vectors.append(vector)

          print(len(title_avg_w2v_vectors))
          print(len(title_avg_w2v_vectors[0]))
```

```
100%|| 109248/109248 [00:01<00:00, 91725.21it/s]


109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

```
In [117]: # 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_)))
essay_tfidf_words = set(tfidf_model.get_feature_names())
```

In [118]:
```
# average Word2Vec
# compute average word2vec for each review.
essay_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in th
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 essay_tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors.append(vector)

print(len(essay_tfidf_w2v_vectors))
print(len(essay_tfidf_w2v_vectors[0]))
```

```
100%|| 109248/109248 [02:52<00:00, 632.18it/s]


109248
300
```

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

In [119]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_project_title)
# 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_)))
title_tfidf_words = set(tfidf_model.get_feature_names())
```

In [120]:
```
# average Word2Vec
# compute average word2vec for each review.
title_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in th
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 title_tfidf_words):
```

```
                    vec = model[word] # getting the vector for each word
                    # here we are multiplying idf value(dictionary[word]) and the tf value((
                    tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
                    vector += (vec * tf_idf) # calculating tfidf weighted w2v
                    tf_idf_weight += tf_idf
            if tf_idf_weight != 0:
                vector /= tf_idf_weight
            title_tfidf_w2v_vectors.append(vector)

        print(len(title_tfidf_w2v_vectors))
        print(len(title_tfidf_w2v_vectors[0]))
```

```
100%|| 109248/109248 [02:51<00:00, 638.11it/s]


109248
300
```

2.2.3 1.4.3 Vectorizing Numerical features

```
In [121]: # check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
          # standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn
          from sklearn.preprocessing import StandardScaler

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

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

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

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

```
In [122]: price_standardized
```

```
Out[122]: array([[-0.3905327 ],
                 [ 0.00239637],
                 [ 0.59519138],
                 ...,
                 [-0.15825829],
                 [-0.61243967],
                 [-0.51216657]])
```

```
In [123]: project_data.columns
```

```
Out[123]: 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',
                 'is_digit_present_in_project_summary'],
                dtype='object')
```

```
In [124]: # check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
          # standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn
          from sklearn.preprocessing import StandardScaler

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

          price_scalar = StandardScaler()
          price_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values
          print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.va

          # Now standardize the data with above maen and variance.
          teacher_number_of_previously_posted_projects_standardized = price_scalar.transform(pr
```

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

2.2.4 1.4.4 Merging all the above features

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

```
In [125]: 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 [126]: # 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 mat
          X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized)
          X.shape
```

Out[126]: (109248, 16663)

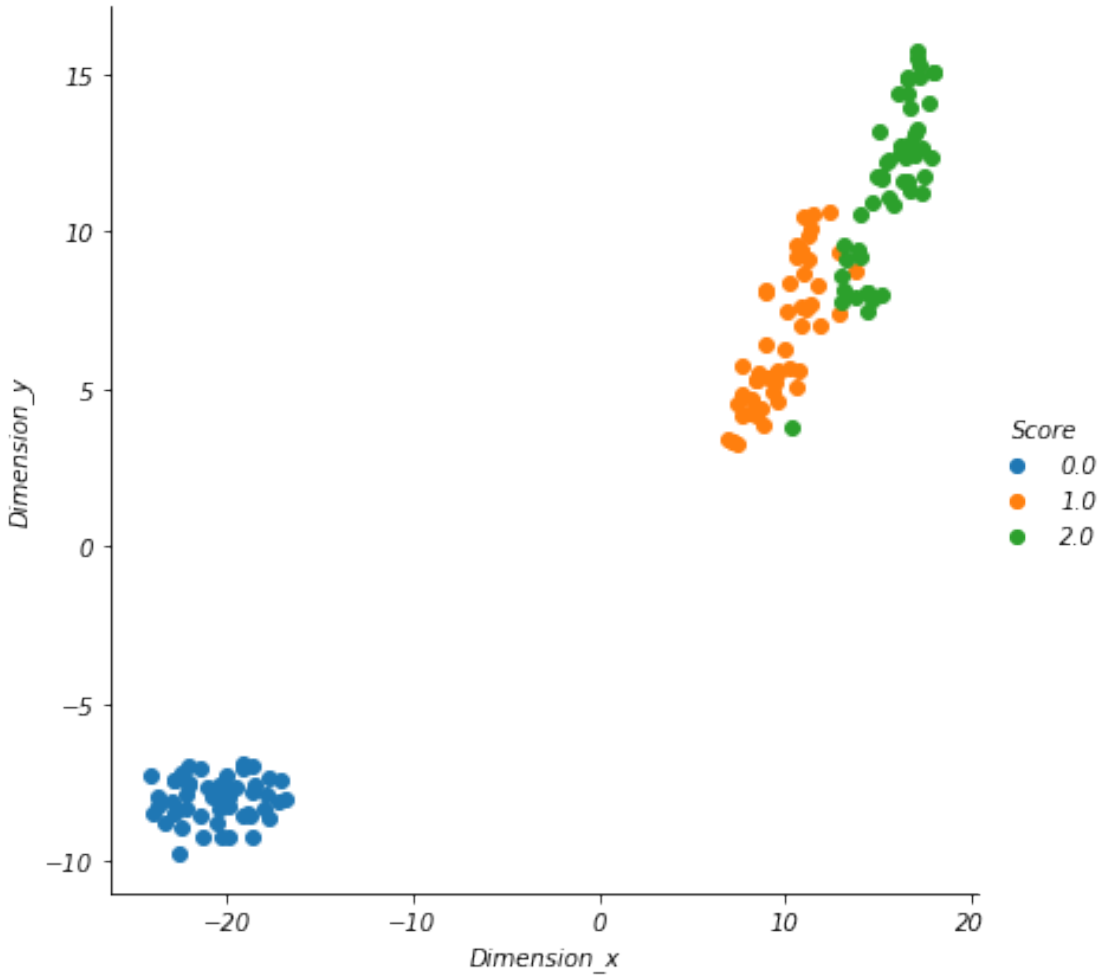
Assignment 2: Apply TSNE

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.

```
<li> In the above cells we have plotted and analyzed many features. Please observe the plots ar
<li> EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_proj
<li>
    <ul>Build the data matrix using these features
        <li>school_state : categorical data (one hot encoding)</li>
        <li>clean_categories : categorical data (one hot encoding)</li>
        <li>clean_subcategories : categorical data (one hot encoding)</li>
        <li>teacher_prefix : categorical data (one hot encoding)</li>
        <li>project_grade_category : categorical data (one hot encoding)</li>
        <li>project_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)</li>
        <li>price : numerical</li>
        <li>teacher_number_of_previously_posted_projects : numerical</li>
     </ul>
</li>
<li> Now, plot FOUR t-SNE plots with each of these feature sets.
    <ol>
        <li>categorical, numerical features + project_title(BOW)</li>
        <li>categorical, numerical features + project_title(TFIDF)</li>
        <li>categorical, numerical features + project_title(AVG W2V)</li>
        <li>categorical, numerical features + project_title(TFIDF W2V)</li>
    </ol>
</li>
<li> Concatenate all the features and Apply TNSE on the final data matrix </li>
<li> <font color='blue'>Note 1: The TSNE accepts only dense matrices</font></li>
<li> <font color='blue'>Note 2: Consider only 5k to 6k data points to avoid memory issues. If y
```

In [137]:
```
# 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,random_state=42)

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

```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score
sn.FacetGrid(for_tsne_df,hue='Score',height=6).map(plt.scatter,'Dimension_x','Dimens
plt.show()
```



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

```
In [159]: # please write all of the code with proper documentation and proper titles for each
          # 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
```

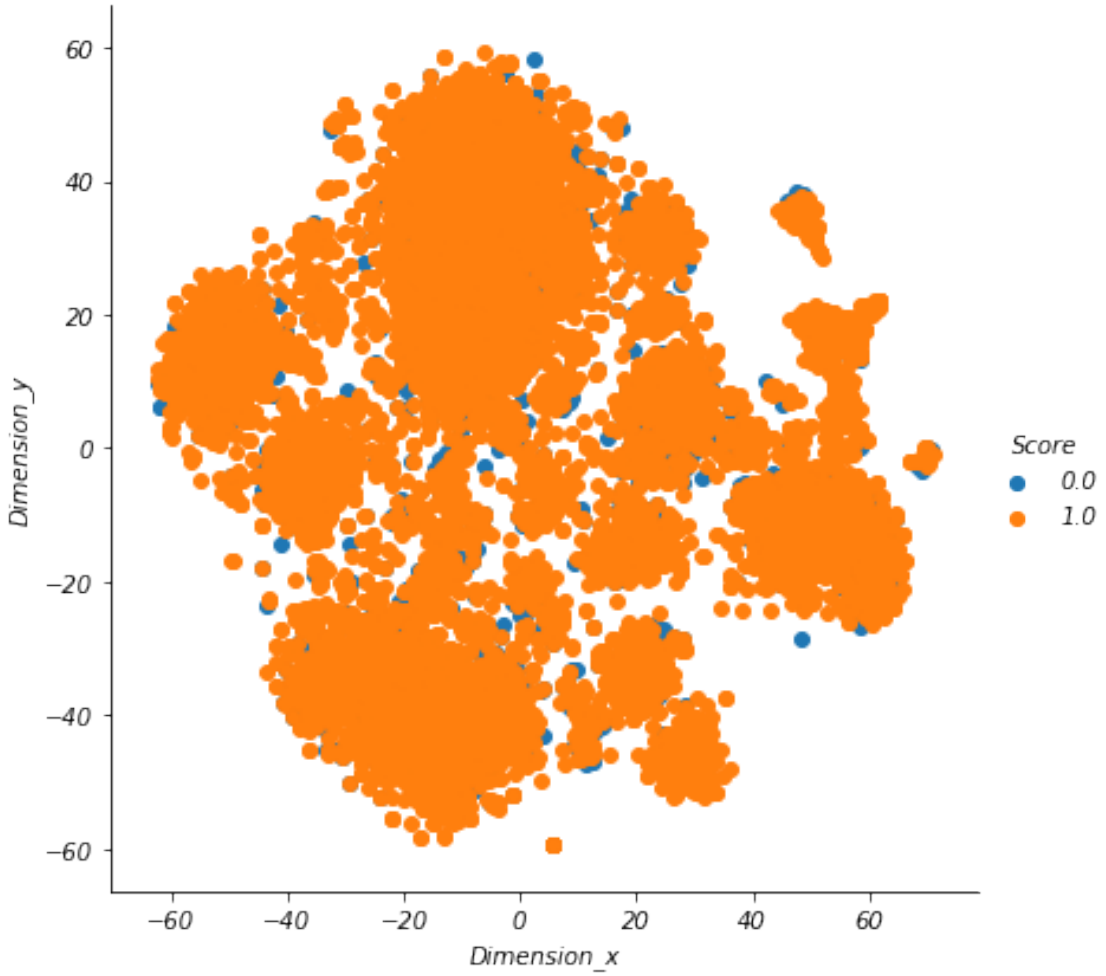
```
In [160]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
          from scipy.sparse import hstack
```

```
# with the same hstack function we are concatinating a sparse matrix and a dense mat
BOW = hstack((categories_one_hot, sub_categories_one_hot,school_state_one_hot,teache
BOW.shape
```

Out[160]: (109248, 3427)

In [161]:
```
# 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

BOW=BOW.todense()
data_size=10000
small_bow=BOW[:data_size,:]
smallY=project_data['project_is_approved'].values[:data_size]
print(small_bow.shape)
print(smallY.shape)

x = small_bow
y = smallY

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.

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
print(for_tsne.shape)
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score
sn.FacetGrid(for_tsne_df,hue='Score',height=6).map(plt.scatter,'Dimension_x','Dimensi
plt.show()
```

```
(10000, 3427)
(10000,)
(10000, 3)
```



Insight: The data is not linearly spearable in 2 dimention

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

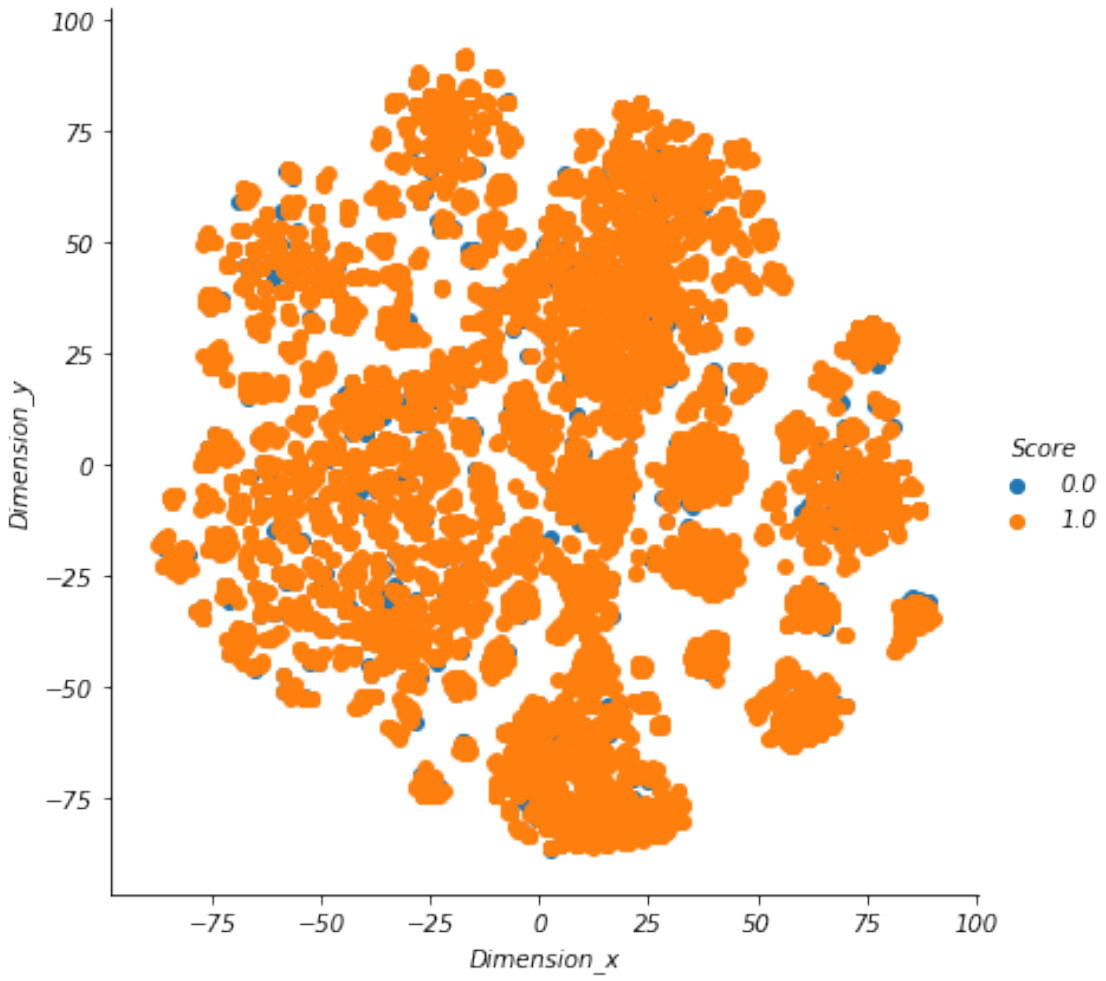
```
In [162]: # please write all the code with proper documentation, and proper titles for each su
          # when you plot any graph make sure you use
              # a. Title, that describes your plot, this will be very helpful to the reader
              # b. Legends if needed
              # c. X-axis label
              # d. Y-axis label
```

```
In [163]: # 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 mat
          TFIDF = hstack((categories_one_hot, sub_categories_one_hot,school_state_one_hot,teach
          TFIDF.shape
```

```
Out[163]: (109248, 3427)
```

In [164]:
```
# 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

TFIDF=TFIDF.todense()
data_size=10000
small_bow=TFIDF[:data_size,:]
smallY=project_data['project_is_approved'].values[:data_size]
print(small_bow.shape)
print(smallY.shape)

x = small_bow
y = smallY

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.

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
print(for_tsne.shape)
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score
sn.FacetGrid(for_tsne_df,hue='Score',height=6).map(plt.scatter,'Dimension_x','Dimens
plt.show()
```

```
(10000, 3427)
(10000,)
(10000, 3)
```



Insight: The data is not linearly spearable in 2 dimention

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

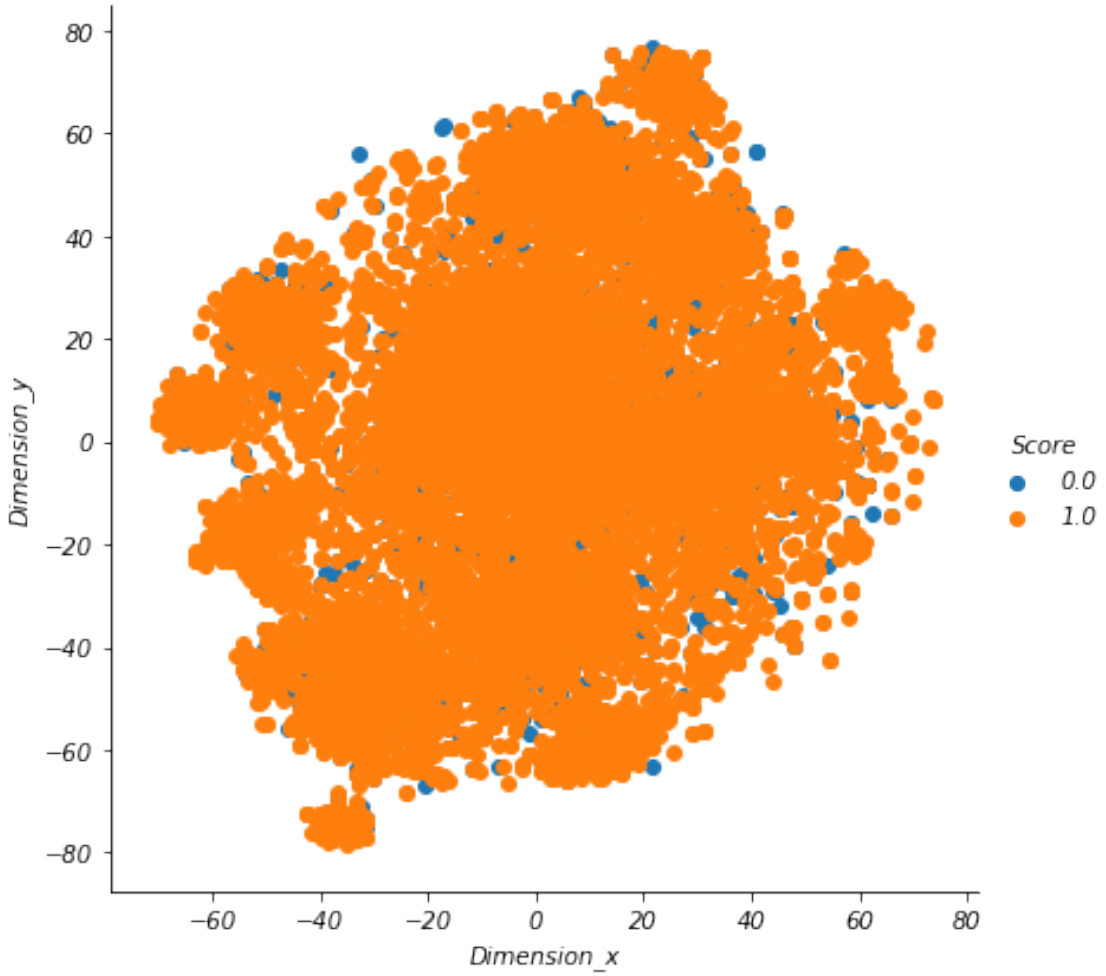
```
In [165]: # please write all the code with proper documentation, and proper titles for each su
          # when you plot any graph make sure you use
              # a. Title, that describes your plot, this will be very helpful to the reader
              # b. Legends if needed
              # c. X-axis label
              # d. Y-axis label
```

```
In [166]: # 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 mat
          AVG_W2V = hstack((categories_one_hot, sub_categories_one_hot,school_state_one_hot,tea
          AVG_W2V.shape
```

```
Out[166]: (109248, 398)
```

In [167]:
```
# 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

AVG_W2V=AVG_W2V.todense()
data_size=10000
small_bow=AVG_W2V[:data_size,:]
smallY=project_data['project_is_approved'].values[:data_size]
print(small_bow.shape)
print(smallY.shape)

x = small_bow
y = smallY

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.

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
print(for_tsne.shape)
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score
sn.FacetGrid(for_tsne_df,hue='Score',height=6).map(plt.scatter,'Dimension_x','Dimens
plt.show()
```

```
(10000, 398)
(10000,)
(10000, 3)
```



Insight: The data is not linearly spearable in 2 dimention

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

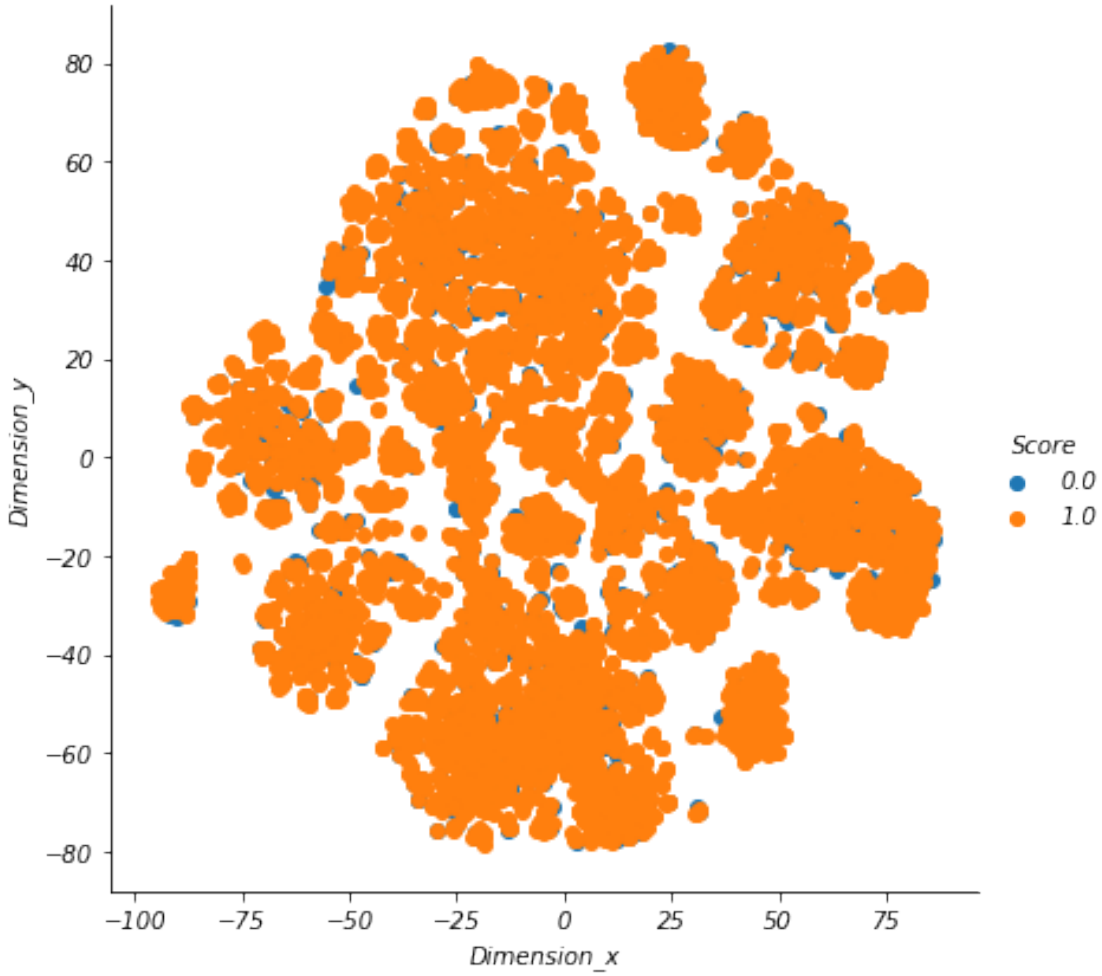
```
In [168]: # please write all the code with proper documentation, and proper titles for each su
          # when you plot any graph make sure you use
              # a. Title, that describes your plot, this will be very helpful to the reader
              # b. Legends if needed
              # c. X-axis label
              # d. Y-axis label
```

```
In [169]: # 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 mat
          TFIDF_Weighted_W2V = hstack((categories_one_hot, sub_categories_one_hot,school_state_
          TFIDF_Weighted_W2V.shape
```

```
Out[169]: (109248, 398)
```

In [170]:
```
# 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

TFIDF_Weighted_W2V=TFIDF_Weighted_W2V.todense()
data_size=10000
small_bow=TFIDF_Weighted_W2V[:data_size,:]
smallY=project_data['project_is_approved'].values[:data_size]
print(small_bow.shape)
print(smallY.shape)

x = small_bow
y = smallY

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.

for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
print(for_tsne.shape)
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Score
sn.FacetGrid(for_tsne_df,hue='Score',height=6).map(plt.scatter,'Dimension_x','Dimens
plt.show()
```

```
(10000, 398)
(10000,)
(10000, 3)
```



Insight: The data is not linearly spearable in 2 dimention
2.5 Summary

```
In [172]: project_data.columns
```

```
Out[172]: 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',
                 'is_digit_present_in_project_summary'],
                dtype='object')
```

Final Summary: * As TSNE on 10k points using title column did not show much result, it is obvious that the data is less likely to be separable in 2 dimentions using title as a feature. May be combination of all text coulmns can help.

- project_grade_category, essay(which is a combination of colums 'project_essay_1', 'project_essay_2', 'project_essay_3', 'project_essay_4), category, subcategory fileds seems powerful, which can help us on building ML models