Projects_Auto_Approval_tSNE

June 23, 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, 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

```
<li>How to increase the consistency of project vetting across different volunteers to improve t
<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!</code

`project_essay_1` | First application essay
`project_essay_2` | Second application essay `project_essay_3` | Third application essay *`project_essay_4` | Fourth application essay* `project_submitted_datetime` | Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id` | A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix` | Teacher's title. One of the following enumerated values:

nan
Dr.
Mr.
Mrs.
Ms.
Teacher.

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

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

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

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

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

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

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

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.

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

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

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

import re
```

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

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

from tqdm import tqdm
import os

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

warnings.filterwarnings("ignore")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning:
paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install
paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled.
`pip install paramiko` to suppress')
C:\ProgramData\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning:
detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.2 1.1 Reading Data

[2]:

```
project_data = pd.read_csv("C:\\VipinML\\Assignment␣
 ↪2\\Assignments_DonorsChoose_2018\\train_data.csv")
resource_data = pd.read_csv("C:\\VipinML\Assignment␣
 ↪2\\Assignments_DonorsChoose_2018\\resources.csv")

project_data = project_data.head(7000)
resource_data = resource_data.head(7000)
```

[3]:

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

project_data.head(4)
```

```
Number of data points in train data (7000, 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']
```

[3]:

```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.
2       21895  p182444  3465aaf82da834c0582ebd0ef8040ca0            Ms.
3          45  p246581  f3cb9bffbba169bef1a77b243e620b60           Mrs.

  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
2           AZ        2016-08-31 12:03:56             Grades 6-8
3           KY        2016-10-06 21:16:17          Grades PreK-2

            project_subject_categories     project_subject_subcategories  \
0                  Literacy & Language                    ESL, Literacy
1    History & Civics, Health & Sports  Civics & Government, Team Sports
2                      Health & Sports    Health & Wellness, Team Sports
3  Literacy & Language, Math & Science             Literacy, Mathematics

                                       project_title  \
0   Educational Support for English Learners at Home
1              Wanted: Projector for Hungry Learners
2  Soccer Equipment for AWESOME Middle School Stu...
3                             Techie Kindergarteners

                                     project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...
2  \r\n\"True champions aren't always the ones th...
3  I work at a unique school filled with both ESL...

                                     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
2  The students on the campus come to school know...             NaN
3  My students live in high poverty conditions wi...             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...
2             NaN  My students need shine guards, athletic socks,...
3             NaN  My students need to engage in Reading and Math...

   teacher_number_of_previously_posted_projects  project_is_approved
0                                             0                    0
1                                             7                    1
2                                             1                    0
3                                             4                    1
```

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

```
Number of data points in train data (7000, 4)
['id' 'description' 'quantity' 'price']
```

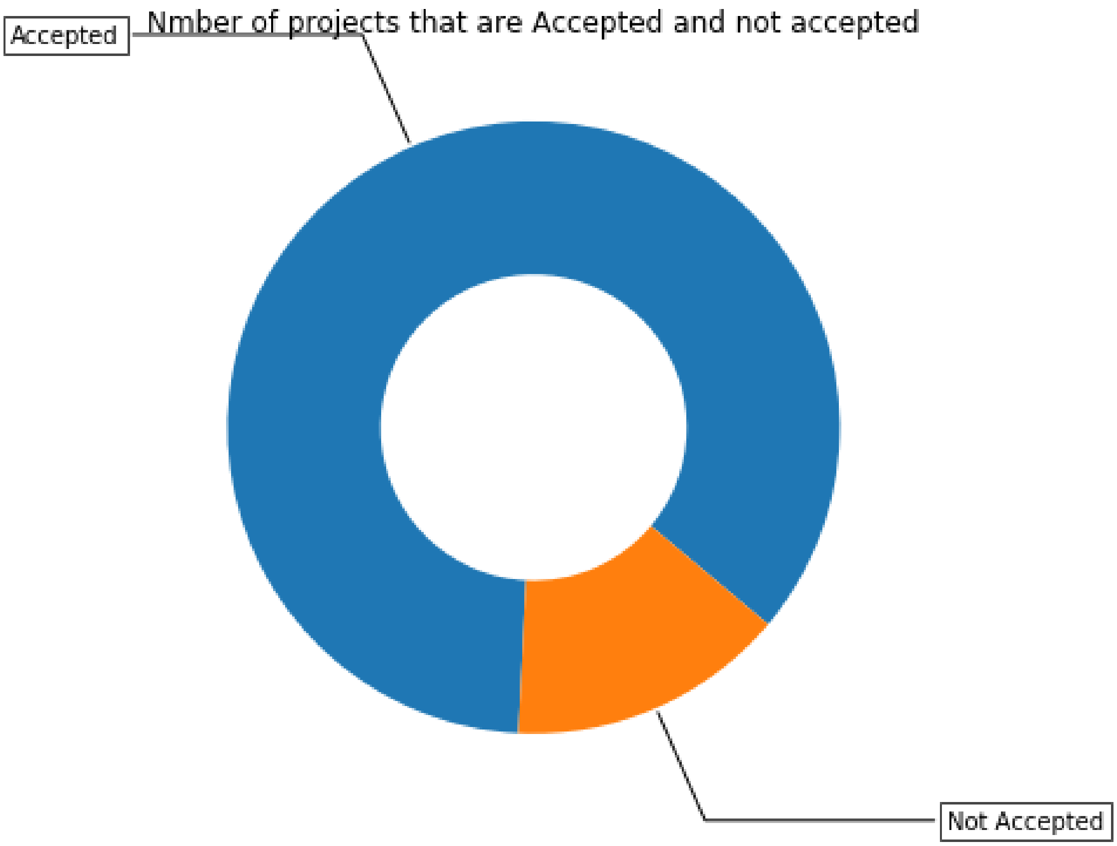
[4]:
```
        id                                        description  quantity  \
0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
1  p069063        Bouncy Bands for Desks (Blue support pipes)         3
2  p069063  Cory Stories: A Kid's Book About Living With Adhd         1
3  p069063  Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...         2

    price
0  149.00
1   14.95
2    8.45
3   13.59
```

2 1.2 Data Analysis

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

# project is approved or not approved.
y_value_counts = project_data['project_is_approved'].value_counts()
# print all cpunts of approved and non-approved projects.
print (y_value_counts)


print("Number of projects thar are approved for funding ", y_value_counts[1],␣
 ↪", (", (y_value_counts[1]/(y_value_counts[1]+y_value_counts[0]))*100,"%)")
```

```
print("Number of projects thar are not approved for funding ",␣
 ↪y_value_counts[0], ", (", (y_value_counts[0]/
 ↪(y_value_counts[1]+y_value_counts[0]))*100,"%)")

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

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

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

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

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1

    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))

    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                 horizontalalignment=horizontalalignment, **kw)

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

plt.show()
```

```
1    5972
0    1028
Name: project_is_approved, dtype: int64
Number of projects thar are approved for funding  5972 , ( 85.3142857142857 %)
Number of projects thar are not approved for funding  1028 , (
14.685714285714285 %)
```



2.0.1 1.2.1 Univariate Analysis: School State

[6]:

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

temp = pd.DataFrame(project_data.groupby("school_state")["project_is_approved"].
 ↪apply(np.mean)).reset_index()

# if you have data which contain only 0 and 1, then the mean = percentage␣
 ↪(think about it)
temp.columns = ['state_code', 'num_proposals']

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

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

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

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

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

[6]:

```
'# How to plot US state heatmap:
https://datascience.stackexchange.com/a/9620\n\nscl = [[0.0,
\'rgb(242,240,247)\'],[0.2, \'rgb(218,218,235)\'],[0.4, \'rgb(188,189,220)\'],
[0.6, \'rgb(158,154,200)\'],[0.8, \'rgb(117,107,177)\'],[1.0,
\'rgb(84,39,143)\']]\n\ndata = [ dict(\n        type=\'choropleth\',\n
colorscale = scl,\n        autocolorscale = False,\n        locations =
temp[\'state_code\'],\n        z = temp[\'num_proposals\'].astype(float),\n
locationmode = \'USA-states\',\n        text = temp[\'state_code\'],\n
marker = dict(line = dict (color = \'rgb(255,255,255)\',width = 2)),\n
colorbar = dict(title = "% of pro")\n    ) ]\n\nlayout = dict(\n        title =
\'Project Proposals % of Acceptance Rate by US States\',\n        geo = dict(\n
scope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n
showlakes = True,\n            lakecolor = \'rgb(255, 255, 255)\',\n        ),\n
)\n\nfig = go.Figure(data=data, layout=layout)\noffline.iplot(fig,
filename=\'us-map-heat-map\')\n'
```

[7]:

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

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.500000
41         SD       0.714286
50         WY       0.727273
7          DC       0.766667
0          AK       0.782609
==================================================
States with highest % approvals
   state_code  num_proposals
23         MN       0.919355
32         NM       0.935484
16         KS       0.977273
28         ND       1.000000
8          DE       1.000000
```

[8]:

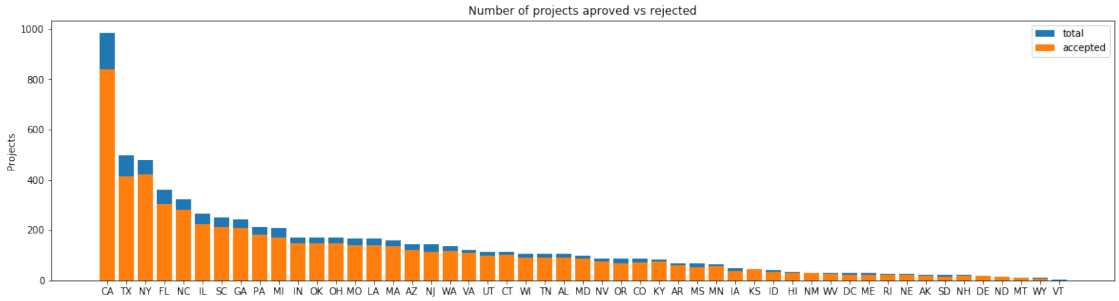
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/
 ↪lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

    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()
```

[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 group by count: https://stackoverflow.com/a/19385591/
 ↪4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'total':
 ↪'count'})).reset_index()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':
 ↪'mean'})).reset_index()['Avg']
```

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

    if top:
        temp = temp[0:top]

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

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

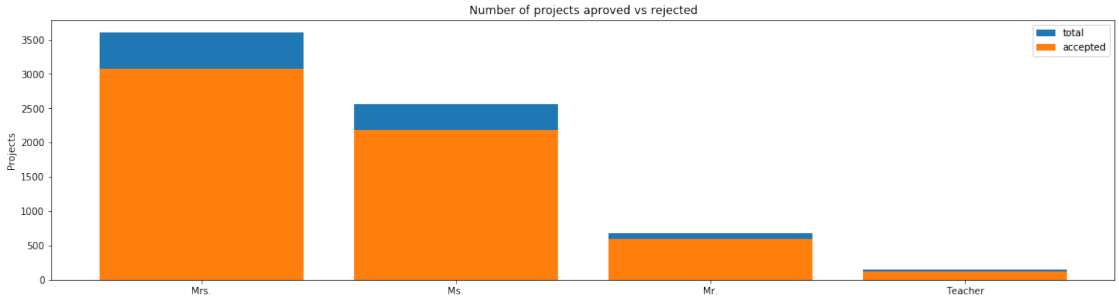


```
   school_state  project_is_approved  total       Avg
4            CA                  841    984  0.854675
43           TX                  412    496  0.830645
34           NY                  421    479  0.878914
9            FL                  302    361  0.836565
27           NC                  279    322  0.866460
==================================================
   school_state  project_is_approved  total       Avg
8            DE                   18     18  1.000000
28           ND                   13     13  1.000000
26           MT                    9     11  0.818182
50           WY                    8     11  0.727273
46           VT                    1      2  0.500000
```

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

2.0.2 1.2.2 Univariate Analysis: teacher_prefix

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

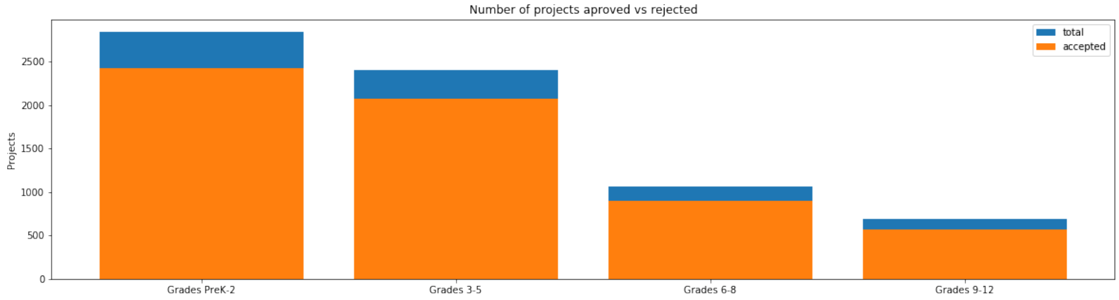


```
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 3082   3606  0.854687
2            Ms.                 2180   2561  0.851230
0            Mr.                  593    684  0.866959
3        Teacher                  117    149  0.785235
==================================================
```

```
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 3082   3606  0.854687
2            Ms.                 2180   2561  0.851230
0            Mr.                  593    684  0.866959
3        Teacher                  117    149  0.785235
```

2.0.3 1.2.3 Univariate Analysis: project_grade_category

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



```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 2423   2842  0.852569
0             Grades 3-5                 2078   2406  0.863674
1             Grades 6-8                  903   1068  0.845506
2            Grades 9-12                  568    684  0.830409
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 2423   2842  0.852569
0             Grades 3-5                 2078   2406  0.863674
1             Grades 6-8                  903   1068  0.845506
2            Grades 9-12                  568    684  0.830409
```

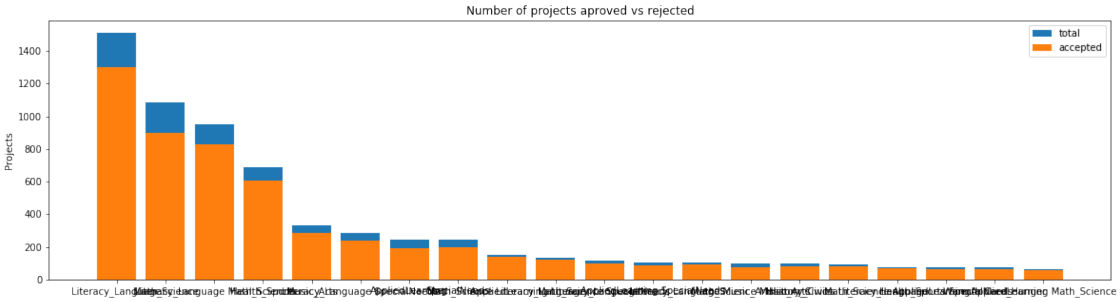
2.0.4 1.2.4 Univariate Analysis: project_subject_categories

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

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

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

[14]:
```
   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_subject_subcategories  \
0                      ESL, Literacy
1  Civics & Government, Team Sports

                                      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
0             Literacy_Language
1  History_Civics Health_Sports
```

```
[15]: univariate_barplots(project_data, 'clean_categories', 'project_is_approved',␣
      ↪top=20)
```



```
                  clean_categories  project_is_approved  total       Avg
23               Literacy_Language                 1299   1511  0.859696
31                    Math_Science                  901   1086  0.829650
27  Literacy_Language Math_Science                  831    948  0.876582
8                    Health_Sports                  605    691  0.875543
39                      Music_Arts                  287    332  0.864458
```

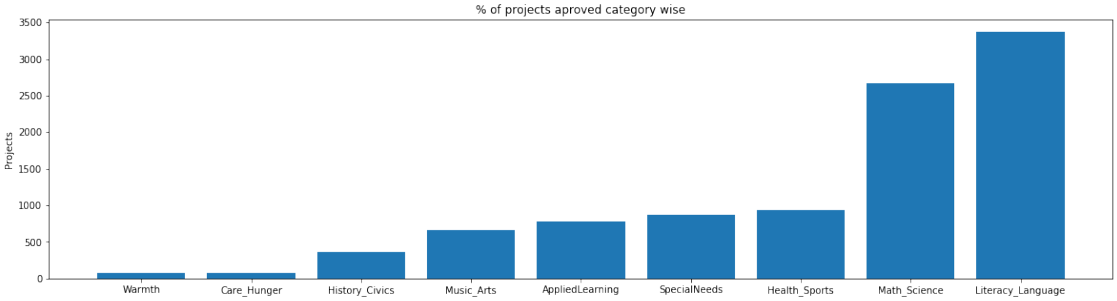
```
==================================================
                   clean_categories  project_is_approved  total       Avg
19  History_Civics Literacy_Language                   84     90  0.933333
32      Math_Science AppliedLearning                   67     76  0.881579
14        Health_Sports SpecialNeeds                   65     76  0.855263
49                Warmth Care_Hunger                   64     73  0.876712
4       AppliedLearning Math_Science                   56     66  0.848485
```

[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())
```

[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()
```



[18]:

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

```
Warmth               :        83
Care_Hunger          :        83
History_Civics       :       364
Music_Arts           :       657
```

```
AppliedLearning      :       779
SpecialNeeds         :       868
Health_Sports        :       940
Math_Science         :      2662
Literacy_Language    :      3371
```

2.0.5 1.2.5 Univariate Analysis: project_subject_subcategories

[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"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to␣
 ↪replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with␣
 ↪''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the␣
 ↪trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

[20]:

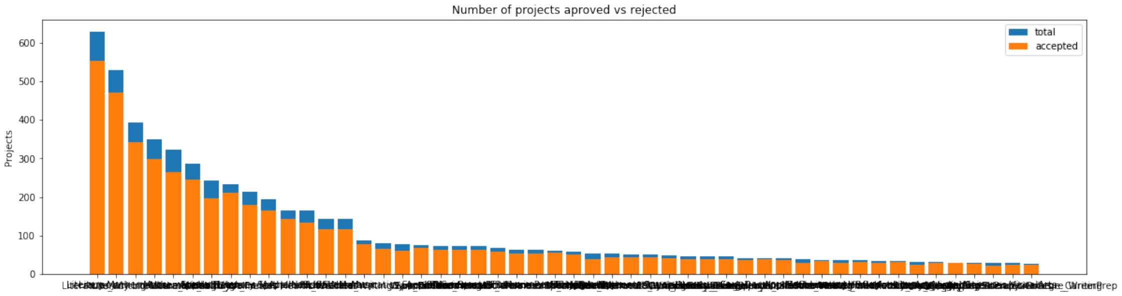
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

[20]:

```
   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
```

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

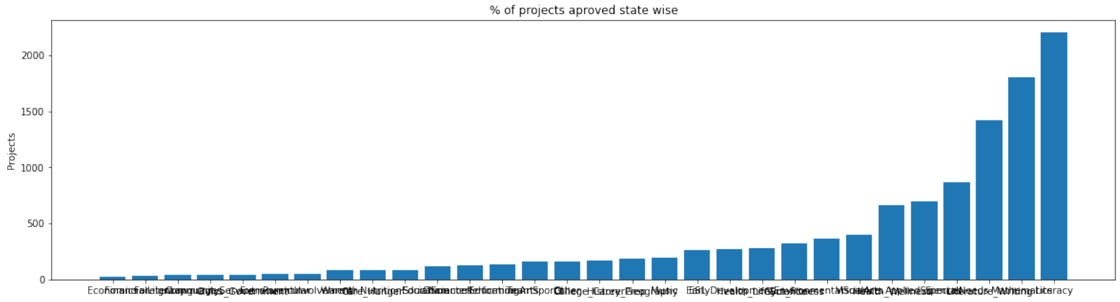


```
                 clean_subcategories  project_is_approved  total       Avg
210                         Literacy                  553    629  0.879173
212             Literacy Mathematics                  470    530  0.886792
222  Literature_Writing Mathematics                  342    394  0.868020
211     Literacy Literature_Writing                  299    350  0.854286
231                      Mathematics                  264    323  0.817337
==================================================
                     clean_subcategories  project_is_approved  total  \
202             History_Geography Literacy                   28     30
```

```
227      Literature_Writing SocialSciences                  26      29
23              AppliedSciences VisualArts                  21      29
65    College_CareerPrep Literature_Writing                 24      28
3       AppliedSciences College_CareerPrep                  23      26

          Avg
202  0.933333
227  0.896552
23   0.724138
65   0.857143
3    0.884615
```

[22]:
```python
# 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())
```

[23]:
```python
# 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()
```



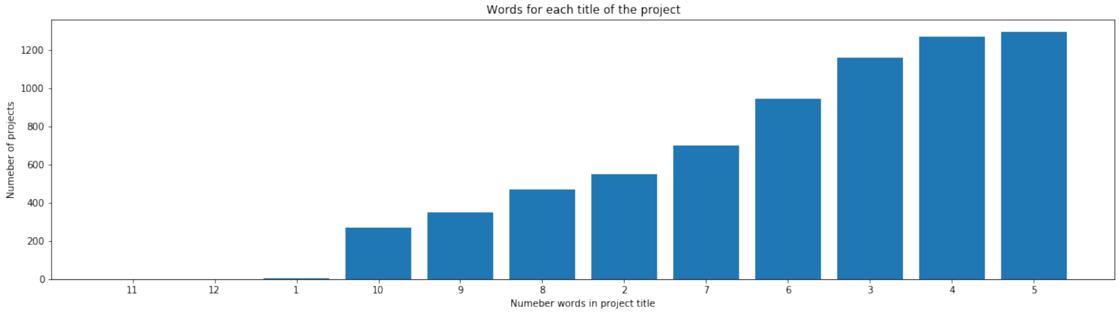
[24]:
```python
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :         21
FinancialLiteracy    :         31
ForeignLanguages     :         39
CommunityService     :         41
Civics_Government    :         43
Extracurricular      :         46
ParentInvolvement    :         47
Warmth               :         83
Care_Hunger          :         83
NutritionEducation   :         83
SocialSciences       :        119
CharacterEducation   :        125
PerformingArts       :        133
TeamSports           :        160
Other                :        162
College_CareerPrep   :        169
History_Geography    :        188
Music                :        191
ESL                  :        265
EarlyDevelopment     :        275
Health_LifeScience   :        277
Gym_Fitness          :        324
EnvironmentalScience :        369
VisualArts           :        400
Health_Wellness      :        666
AppliedSciences      :        697
SpecialNeeds         :        868
Literature_Writing   :       1422
Mathematics          :       1800
Literacy             :       2204
```

2.0.6 1.2.6 Univariate Analysis: Text features (Title)

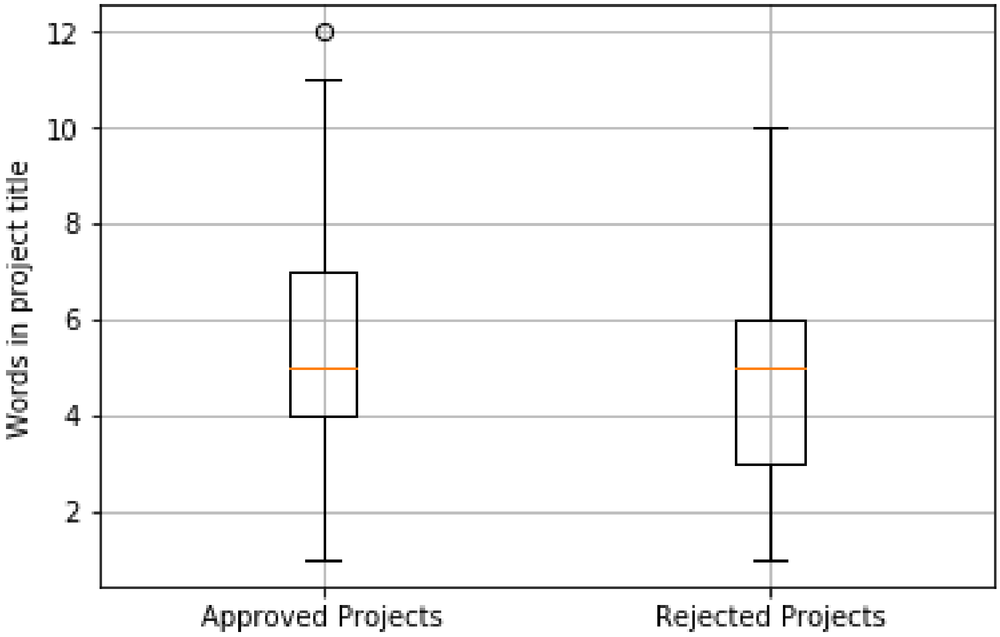
[25]:

```python
#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()
```

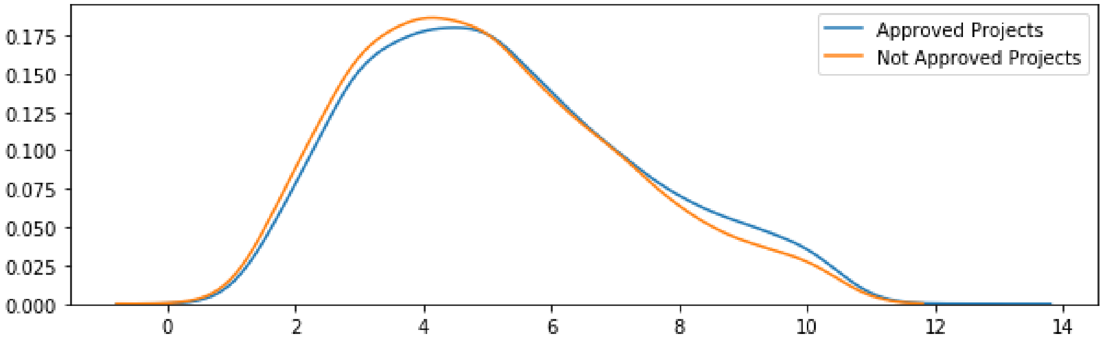


[26]:
```
approved_title_word_count =␣
 ↪project_data[project_data['project_is_approved']==1]['project_title'].str.
 ↪split().apply(len)
approved_title_word_count = approved_title_word_count.values

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

[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()
```



[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()
```

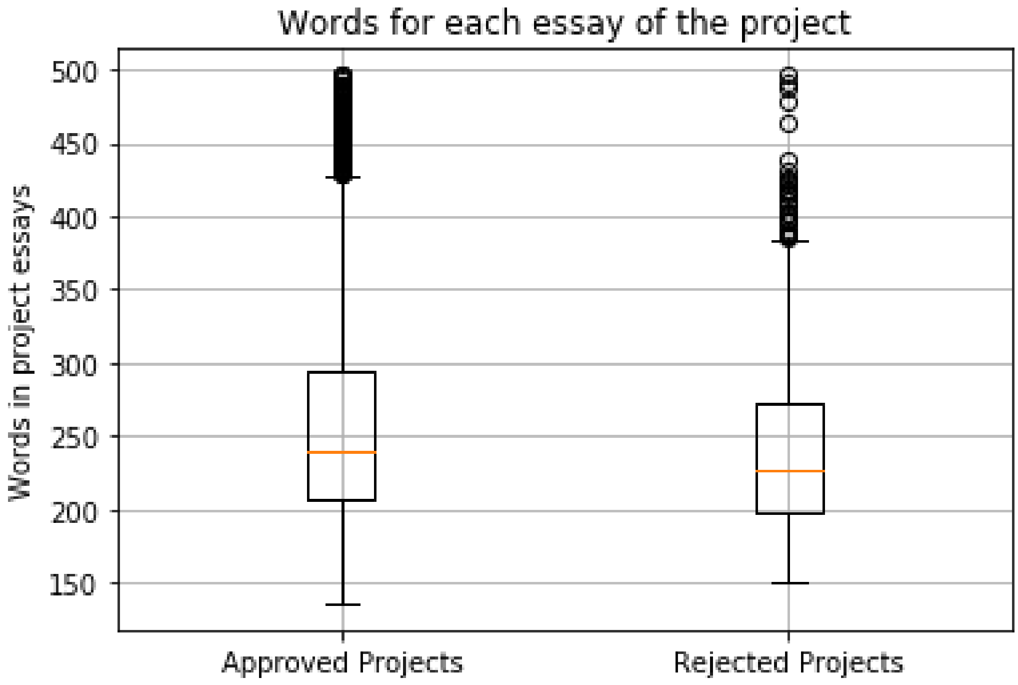


2.0.7 1.2.7 Univariate Analysis: Text features (Project Essay's)

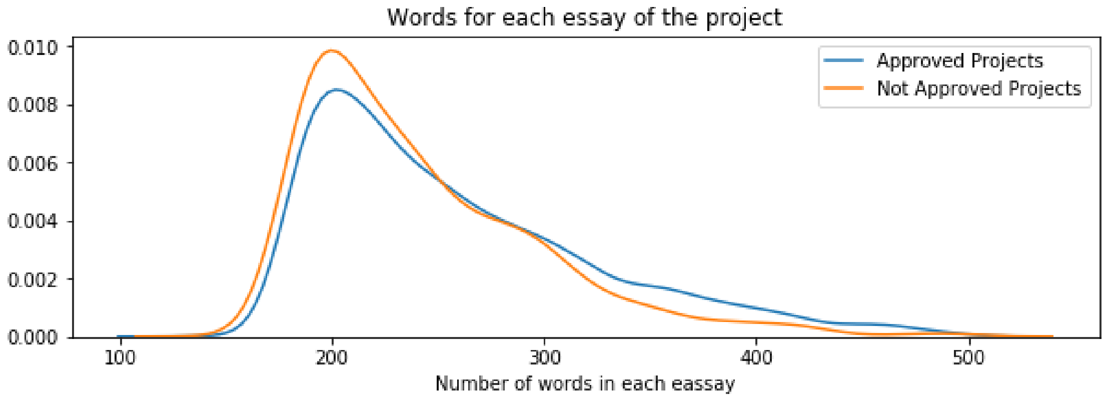
[29]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                            project_data["project_essay_4"].map(str)
```

[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
```

[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()
```



[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()
```



2.0.8 1.2.8 Univariate Analysis: Cost per project

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

[33]:
```
        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
```

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

[34]:
```
        id    price  quantity
0  p000341  1295.23        12
1  p000477   443.49         6
```

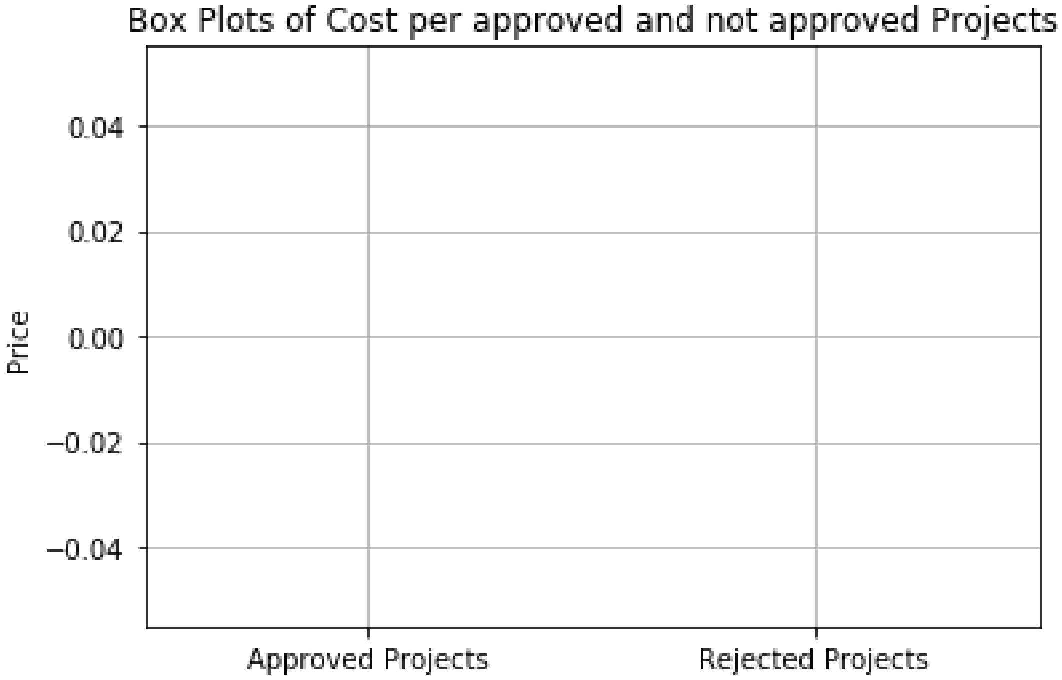
[35]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

[36]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].
 ↪values
print (approved_price)

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

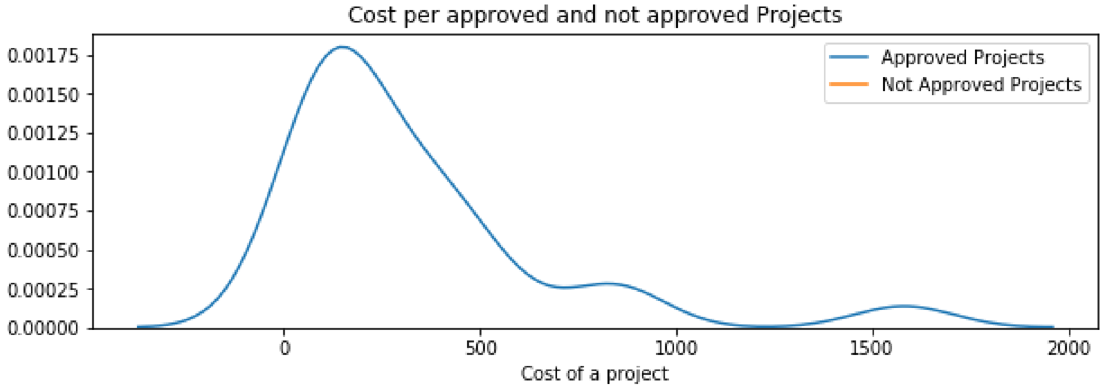
```
print (rejected_price)
```

```
[nan nan nan ... nan nan nan]
[nan nan nan ... nan nan nan]
```

[37]:
```
# 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()
```



[38]:
```
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()
```



[39]:

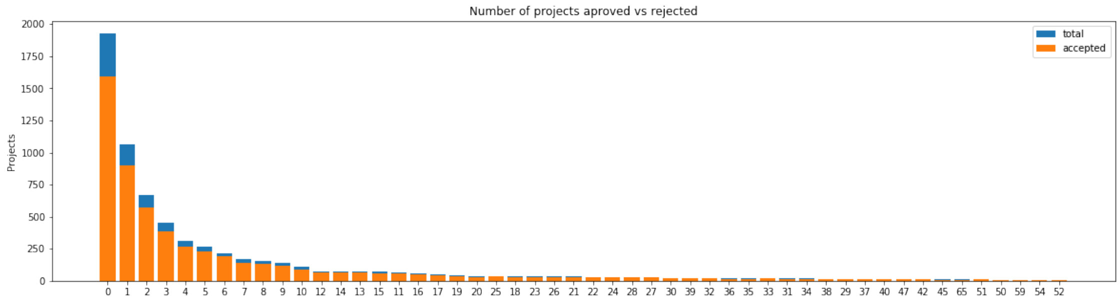
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

#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), 3)])
print(x)
```

```
+------------+-------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+-------------------+-----------------------+
|     0      |        nan        |          nan          |
|     5      |        nan        |          nan          |
|     10     |        nan        |          nan          |
|     15     |        nan        |          nan          |
|     20     |        nan        |          nan          |
|     25     |        nan        |          nan          |
|     30     |        nan        |          nan          |
|     35     |        nan        |          nan          |
|     40     |        nan        |          nan          |
|     45     |        nan        |          nan          |
|     50     |        nan        |          nan          |
|     55     |        nan        |          nan          |
|     60     |        nan        |          nan          |
|     65     |        nan        |          nan          |
|     70     |        nan        |          nan          |
|     75     |        nan        |          nan          |
|     80     |        nan        |          nan          |
|     85     |        nan        |          nan          |
|     90     |        nan        |          nan          |
|     95     |        nan        |          nan          |
|    100     |        nan        |          nan          |
+------------+-------------------+-----------------------+
```

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

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

[40]:
```
univariate_barplots(project_data,␣
↪'teacher_number_of_previously_posted_projects', 'project_is_approved',␣
↪top=50)
```



```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                 1591   1928
1                                             1                  900   1060
2                                             2                  571    669
3                                             3                  388    454
4                                             4                  263    310

        Avg
0  0.825207
1  0.849057
2  0.853513
3  0.854626
4  0.848387
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved  total  \
51                                            51                   10     10
50                                            50                    9      9
59                                            59                    9      9
54                                            54                    9      9
52                                            52                    7      9

         Avg
```

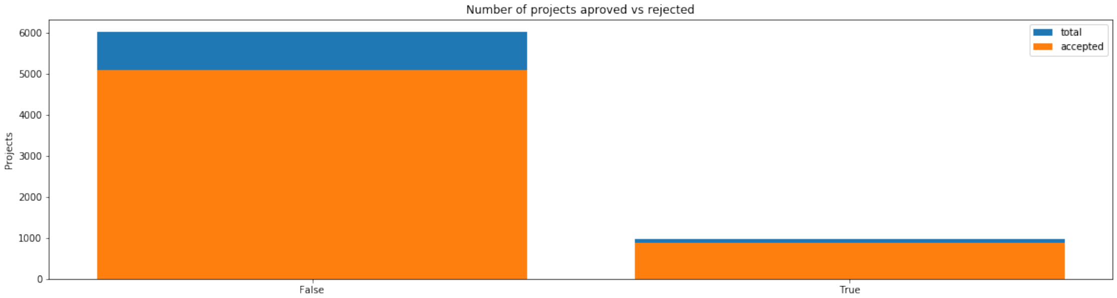
```
51   1.000000
50   1.000000
59   1.000000
54   1.000000
52   0.777778
```

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.

[41]:
```
# add the logic to filter if project summary has any digit in it or now. we␣
 ↪want to see
#if any digit will make any impact on project approvalor not.dgt = []

dgt = []
for values in project_data['project_resource_summary']:
    dgt.append(bool(re.search(r'[1-9]', values)))
project_data['digit_in_resource_summary'] = dgt

univariate_barplots(project_data, 'digit_in_resource_summary',␣
 ↪'project_is_approved', top=50)
```



```
   digit_in_resource_summary  project_is_approved  total       Avg
0                      False                 5096   6022  0.846230
1                       True                  876    978  0.895706
==================================================
   digit_in_resource_summary  project_is_approved  total       Avg
0                      False                 5096   6022  0.846230
1                       True                  876    978  0.895706
```

2.1 1.3 Text preprocessing

2.1.1 1.3.1 Essay Text

[42]:
```
project_data.head(2)
```

[42]:
```
   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_4  \
0  \"The limits of your language are the limits o...  ...             NaN
1  The projector we need for our school is very c...  ...             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...   NaN       NaN
1  Our students arrive to our school eager to lea...   NaN       NaN

   digit_in_resource_summary
0                      False
1                      False

[2 rows x 21 columns]
```

[43]:

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

```
My students are English learners that are working on English as their second or
third languages. We are a melting pot of refugees, immigrants, and native-born
Americans bringing the gift of language to our school. \r\n\r\n We have over 24
languages represented in our English Learner program with students at every
level of mastery.  We also have over 40 countries represented with the families
within our school.  Each student brings a wealth of knowledge and experiences to
us that open our eyes to new cultures, beliefs, and respect.\"The limits of your
language are the limits of your world.\"-Ludwig Wittgenstein  Our English
learner's have a strong support system at home that begs for more resources.
Many times our parents are learning to read and speak English along side of
their children.  Sometimes this creates barriers for parents to be able to help
their child learn phonetics, letter recognition, and other reading
skills.\r\n\r\nBy providing these dvd's and players, students are able to
continue their mastery of the English language even if no one at home is able to
assist.  All families with students within the Level 1 proficiency status, will
be a offered to be a part of this program.  These educational videos will be
specially chosen by the English Learner Teacher and will be sent home regularly
to watch.  The videos are to help the child develop early reading
skills.\r\n\r\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 most of the time. At our school, 97.3% of the students
receive free or reduced price lunch. Of the 560 students, 97.3% are minority
students. \r\nThe school has a vibrant community that loves to get together and
celebrate. Around Halloween there is a whole school parade to show off the
beautiful costumes that students wear. On Cinco de Mayo we put on a big festival
with crafts made by the students, dances, and games. At the end of the year the
school hosts a carnival to celebrate the hard work put in during the school
year, with a dunk tank being the most popular activity.My students will use
these five brightly colored Hokki stools in place of regular, 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 occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the
day they will be used by the students who need the highest amount of movement in
their life in order to stay focused on school.\r\n\r\nWhenever asked what the
classroom is missing, my students always say more Hokki Stools. They can't get
their fill of the 5 stools we already have. When the students are sitting 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 taken. There are always students who head over to the
kidney table to get one of the stools who are disappointed as there are not
enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The
Hokki stools will be a compromise that allow my 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 barrier
that exists in schools for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with
plain walls, rows of desks, and a teacher in front of the room? A typical day in
our room is nothing like that. I work hard to create a warm inviting themed room
for my students look forward to coming to each day.\r\n\r\nMy class is made up
of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey
attend a Title I school, which means there is a high enough percentage of free
and reduced-price lunch to qualify. Our school is an \"open classroom\" concept,
which is very unique as there are no walls separating the classrooms. These 9
and 10 year-old students are very eager learners; they are like sponges,
absorbing all the information and experiences and keep on wanting more.With
these resources such as the comfy red throw pillows and the whimsical nautical
hanging decor and the blue fish nets, I will be able to help create the mood in
our classroom setting to be one of a themed nautical environment. Creating a
classroom environment is very important in the success in each and every child's
education. The nautical photo props will be used with each child as they step
foot into our classroom for the first time on Meet the Teacher evening. I'll
take pictures of each child with them, have them developed, and then hung in our
classroom ready for their first day of 4th grade.  This kind gesture will set
the tone before even the first day of school! The nautical thank you cards will
be used throughout the year by the students as they create thank you cards to
their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs
lost of money out of my own pocket on resources to get our classroom ready.
Please consider helping with this project to make our new school year a very
successful one. Thank you!nannan
```

[44]:

```
# 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
```

[45]:

```
sent = decontracted(project_data['essay'].values[1000])
print(sent)
print("="*50)
```

```
How do you remember your days of school? Was it in a sterile environment with
plain walls, rows of desks, and a teacher in front of the room? A typical day in
our room is nothing like that. I work hard to create a warm inviting themed room
for my students look forward to coming to each day.\r\n\r\nMy class is made up
of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey
attend a Title I school, which means there is a high enough percentage of free
and reduced-price lunch to qualify. Our school is an \"open classroom\" concept,
which is very unique as there are no walls separating the classrooms. These 9
and 10 year-old students are very eager learners; they are like sponges,
absorbing all the information and experiences and keep on wanting more.With
these resources such as the comfy red throw pillows and the whimsical nautical
hanging decor and the blue fish nets, I will be able to help create the mood in
our classroom setting to be one of a themed nautical environment. Creating a
classroom environment is very important in the success in each and every child
is education. The nautical photo props will be used with each child as they step
foot into our classroom for the first time on Meet the Teacher evening. I will
take pictures of each child with them, have them developed, and then hung in our
classroom ready for their first day of 4th grade.  This kind gesture will set
the tone before even the first day of school! The nautical thank you cards will
be used throughout the year by the students as they create thank you cards to
their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs
lost of money out of my own pocket on resources to get our classroom ready.
Please consider helping with this project to make our new school year a very
successful one. Thank you!nannan
==================================================
```

[46]:

```
# \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)
```

```
How do you remember your days of school? Was it in a sterile environment with
plain walls, rows of desks, and a teacher in front of the room? A typical day in
our room is nothing like that. I work hard to create a warm inviting themed room
for my students look forward to coming to each day.    My class is made up of 28
wonderfully unique boys and girls of mixed races in Arkansas.  They attend a
Title I school, which means there is a high enough percentage of free and
reduced-price lunch to qualify. Our school is an  open classroom  concept, which
is very unique as there are no walls separating the classrooms. These 9 and 10
year-old students are very eager learners; they are like sponges, absorbing all
the information and experiences and keep on wanting more.With these resources
such as the comfy red throw pillows and the whimsical nautical hanging decor and
the blue fish nets, I will be able to help create the mood in our classroom
setting to be one of a themed nautical environment. Creating a classroom
environment is very important in the success in each and every child is
education. The nautical photo props will be used with each child as they step
foot into our classroom for the first time on Meet the Teacher evening. I will
take pictures of each child with them, have them developed, and then hung in our
classroom ready for their first day of 4th grade.  This kind gesture will set
the tone before even the first day of school! The nautical thank you cards will
be used throughout the year by the students as they create thank you cards to
their team groups.    Your generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.    It costs lost
of money out of my own pocket on resources to get our classroom ready. Please
consider helping with this project to make our new school year a very successful
one. Thank you!nannan
```

[47]:

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

```
How do you remember your days of school Was it in a sterile environment with
plain walls rows of desks and a teacher in front of the room A typical day in
our room is nothing like that I work hard to create a warm inviting themed room
for my students look forward to coming to each day My class is made up of 28
wonderfully unique boys and girls of mixed races in Arkansas They attend a Title
I school which means there is a high enough percentage of free and reduced price
lunch to qualify Our school is an open classroom concept which is very unique as
there are no walls separating the classrooms These 9 and 10 year old students
are very eager learners they are like sponges absorbing all the information and
experiences and keep on wanting more With these resources such as the comfy red
```

```
throw pillows and the whimsical nautical hanging decor and the blue fish nets I
will be able to help create the mood in our classroom setting to be one of a
themed nautical environment Creating a classroom environment is very important
in the success in each and every child is education The nautical photo props
will be used with each child as they step foot into our classroom for the first
time on Meet the Teacher evening I will take pictures of each child with them
have them developed and then hung in our classroom ready for their first day of
4th grade This kind gesture will set the tone before even the first day of
school The nautical thank you cards will be used throughout the year by the
students as they create thank you cards to their team groups Your generous
donations will help me to help make our classroom a fun inviting learning
environment from day one It costs lost of money out of my own pocket on
resources to get our classroom ready Please consider helping with this project
to make our new school year a very successful one Thank you nannan
```

[48]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',␣
 ↪"you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',␣
 ↪'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',␣
 ↪'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',␣
 ↪'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',␣
 ↪'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',␣
 ↪'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',␣
 ↪'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',␣
 ↪'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',␣
 ↪'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so',␣
 ↪'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should',␣
 ↪"should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',␣
 ↪"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",␣
 ↪'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',␣
 ↪"shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

[49]:
```python
# 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%|| 7000/7000 [00:03<00:00, 1766.13it/s]
```

[50]:
```python
# after preprocesing
preprocessed_essays[2000]
```

[50]:
```
'describing students not easy task many would say inspirational creative hard
working they unique unique interests learning abilities much what common desire
learn day despite difficulties encounter our classroom amazing understand
everyone learns pace as teacher i pride making sure students always engaged
motivated inspired create learning this project help students choose seating
appropriate developmentally many students tire sitting chairs lessons different
seats available helps keep engaged learning flexible seating important classroom
many students struggle attention focus engagement we currently stability balls
seating well regular chairs stools help students trouble balance find difficult
sit stability ball long period time we excited try stools part engaging
classroom community nannan'
```

1.3.2 Project title Text

[51]:
```python
# similarly you can preprocess the titles also
```

[52]:
```python
# logic added for prject title.
sent = decontracted(project_data['project_title'].values[5000])
print(sent)
print("="*50)
```

```
Bouncing Our Wiggles and Worries Away!
==================================================
```

[53]:
```python
# \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)
```

```
Bouncing Our Wiggles and Worries Away!
```

[54]:

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

```
Bouncing Our Wiggles and Worries Away
```

[55]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    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%|| 7000/7000 [00:00<00:00, 38600.92it/s]
```

[56]:

```
# after preprocesing
preprocessed_essays[5000]
```

[56]:

```
'my class made students various grade levels we work hard filling learning gaps
students reach grade level my students dealing emotional issues make hard handle
frustration tasks need lot individual attention by learning work independently
students chance mainstream classrooms peer groups our biggest goal students
learn not control emotions learn students many spent large amount time absent
school different reasons need get routine class task day modeling good classroom
routines task important master move back general education classrooms being
apart title 1 school means resources students need massive lot supplies shared
parents make sure homework completed bouncy bands give students way get rid
anxiety tension energy staying desk working independently students use bands
either desk whole group table chair avoid get asked stop moving movement key
keeping students adhd disabilities focused finishing assignments staying task
teacher teaching my goal help students learn helpful strategies allow join peers
general education setting by learning maintain focus getting wiggles extra
energy grow academically nannan'
```

2.2 1. 4 Preparing data for models

[57]:

```
project_data.columns
```

[57]:

```
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',
       'digit_in_resource_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/

[58]:

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


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (7000, 9)
```

[59]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()),␣
 ↪lowercase=False, binary=True)
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


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

```
['Economics', 'FinancialLiteracy', 'ForeignLanguages', 'CommunityService',
'Civics_Government', 'Extracurricular', 'ParentInvolvement', 'Warmth',
'Care_Hunger', 'NutritionEducation', 'SocialSciences', 'CharacterEducation',
'PerformingArts', 'TeamSports', 'Other', 'College_CareerPrep',
'History_Geography', 'Music', 'ESL', 'EarlyDevelopment', 'Health_LifeScience',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness',
'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics',
'Literacy']
Shape of matrix after one hot encodig  (7000, 30)
```

[60]:
```
# Please do the similar feature encoding with state, teacher_prefix and␣
 ↪project_grade_category also
```

[61]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()),␣
 ↪lowercase=False, binary=True)
vectorizer.fit(project_data['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)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (7000, 9)
```

[62]:
```
# 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)
teacher_prefix_data = project_data['teacher_prefix']
teacher_prefix_notnull = teacher_prefix_data[pd.notnull(teacher_prefix_data)]
vectorizer.fit(teacher_prefix_notnull.values)
print(vectorizer.get_feature_names())
```

```
teacher_prefix_one_hot = vectorizer.transform(teacher_prefix_notnull.values)
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (7000, 9)
```

[63]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()),␣
 ↪lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())


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

```
['Economics', 'FinancialLiteracy', 'ForeignLanguages', 'CommunityService',
'Civics_Government', 'Extracurricular', 'ParentInvolvement', 'Warmth',
'Care_Hunger', 'NutritionEducation', 'SocialSciences', 'CharacterEducation',
'PerformingArts', 'TeamSports', 'Other', 'College_CareerPrep',
'History_Geography', 'Music', 'ESL', 'EarlyDevelopment', 'Health_LifeScience',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness',
'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics',
'Literacy']
Shape of matrix after one hot encodig  (7000, 30)
```

2.2.2 1.4.3 Vectorizing Numerical features

[64]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/
 ↪sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

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

```
Mean : 332.3216666666667, Standard deviation : 340.2684336748791
```

2.2.3 1.4.2 Vectorizing Text data

1.4.2.1 Bag of words 1.4.2.1.1 Bag of Words on preprocessed_essays

[65]:
```
# TEXT BOW
# We are considering only the words which appeared in at least 10␣
 ↪documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (7000, 5222)
```

1.4.2.1.3 Bag of Words on project_title

[66]:
```
# PROJECT_TITLE BOW
# We are considering only the words which appeared in at least 10␣
 ↪documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
project_title_bow = vectorizer.fit_transform(project_data['project_title'])
print("Shape of matrix after one hot encodig ",project_title_bow.shape)
```

```
Shape of matrix after one hot encodig  (7000, 513)
```

1.4.2.1.4 Bag of Words on project_grade_category

[67]:
```
# We are considering only the words which appeared in at least 10␣
 ↪documents(rows or projects).
#project_grade_category after cleanup. see above logic for preprocessed_essays1.
vectorizer = CountVectorizer(min_df=10)
project_grade_category_bow = vectorizer.
 ↪fit_transform(project_data['project_grade_category'])
print("Shape of matrix after one hot encodig ",project_grade_category_bow.shape)
```

```
Shape of matrix after one hot encodig  (7000, 3)
```

1.4.2.1.5 Bag of Words on project_subject_categories -clean_subcategories

[68]:
```
# We are considering only the words which appeared in at least 10␣
 ↪documents(rows or projects).
#project_subject_categories after cleanup. see above logic for␣
 ↪preprocessed_essays1.
```

```
vectorizer = CountVectorizer(min_df=10)
project_subject_categories_bow = vectorizer.
 ↪fit_transform(project_data['clean_subcategories'])
print("Shape of matrix after one hot encodig ",project_subject_categories_bow.
 ↪shape)
```

```
Shape of matrix after one hot encodig  (7000, 30)
```

1.4.2.1.6 Bag of Words on teacher_prefix

[69]:

```
# We are considering only the words which appeared in at least 10␣
 ↪documents(rows or projects).
#teacher_prefix after cleanup. see above logic for preprocessed_essays1.
vectorizer = CountVectorizer(min_df=10)
teacher_prefix_bow = vectorizer.fit_transform(project_data['teacher_prefix'].
 ↪values.astype(str))
print("Shape of matrix after one hot encodig ",teacher_prefix_bow.shape)
```

```
Shape of matrix after one hot encodig  (7000, 4)
```

1.4.2.1.7 Bag of Words on school_state

[70]:

```
# We are considering only the words which appeared in at least 10␣
 ↪documents(rows or projects).
#school_state after cleanup. see above logic for preprocessed_essays1.
vectorizer = CountVectorizer(min_df=10)
school_state_bow = vectorizer.fit_transform(project_data['school_state'])
print("Shape of matrix after one hot encodig ",school_state_bow.shape)
```

```
Shape of matrix after one hot encodig  (7000, 50)
```

1.4.2.1.8 Bag of Words on Price

[71]:

```
# We are considering only the words which appeared in at least 10␣
 ↪documents(rows or projects).
#project_title after cleanup. see above logic for preprocessed_essays1.
vectorizer = CountVectorizer(min_df=10)
price_bow = vectorizer.fit_transform(project_data['price'].values.astype(str))
print("Shape of matrix after one hot encodig ",price_bow.shape)
```

```
Shape of matrix after one hot encodig  (7000, 1)
```

1.4.2.3 TFIDF vectorizer

[72]:

```
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  (7000, 5222)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

[73]:

```
# Similarly you can vectorize for title also
```

[74]:

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

```
Shape of matrix after one hot encodig  (7000, 508)
```

1.4.2.4 TFIDF Vectorizer on project_grade_category

[75]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
project_grade_category_tfidf = vectorizer.
 ↪fit_transform(project_data['project_grade_category'])
print("Shape of matrix after one hot encodig ",project_grade_category_tfidf.
 ↪shape)
```

```
Shape of matrix after one hot encodig  (7000, 3)
```

1.4.2.4 TFIDF Vectorizer on teacher_prefix

[76]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
teacher_prefix_tfidf = vectorizer.fit_transform(project_data['teacher_prefix'].
 ↪values.astype(str))
print("Shape of matrix after one hot encodig ",teacher_prefix_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (7000, 4)
```

1.4.2.4 TFIDF Vectorizer on school_state

[77]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
school_state_tfidf = vectorizer.fit_transform(project_data['school_state'])
print("Shape of matrix after one hot encodig ",school_state_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (7000, 50)
```

1.4.2.1.5 TDIDF on project_subject_categories -clean_subcategories

[78]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
project_subject_categories_tfidf = vectorizer.
 ↪fit_transform(project_data['clean_subcategories'])
print("Shape of matrix after one hot encodig ",project_subject_categories_tfidf.
 ↪shape)
```

```
Shape of matrix after one hot encodig  (7000, 30)
```

1.4.2.1.5 TDIDF on price

[79]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
price_tfidf = vectorizer.fit_transform(project_data['price'].values.astype(str))
print("Shape of matrix after one hot encodig ",price_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (7000, 1)
```

1.4.2.5 Using Pretrained Models: Avg W2V

[80]:

```
'''
# 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-load-variables-in-python/

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


'''
```

[80]:

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

[81]:

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

[82]:

```
# 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%|| 7000/7000 [00:02<00:00, 3008.05it/s]

7000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

[83]:

```
# Similarly you can vectorize for title also
```

[84]:

```
# 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_project_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

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

```
100%|| 7000/7000 [00:00<00:00, 67865.86it/s]

7000
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

[85]:

```
# 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())
```

[86]:

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

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

```
100%|| 7000/7000 [00:16<00:00, 431.79it/s]
```

```
7000
300
```

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

[87]:

```
# Similarly you can vectorize for title also
```

[88]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data['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_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

[89]:

```
# 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(project_data['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf␣
 ↪value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.
 ↪split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

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

```
100%|| 7000/7000 [00:00<00:00, 77121.63it/s]

7000
300
```

[90]:
```
price_standardized
```

[90]:
```
array([[nan],
       [nan],
       [nan],
       ...,
       [nan],
       [nan],
       [nan]])
```

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

[91]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(7000, 9)
(7000, 30)
```

```
(7000, 5222)
(7000, 1)
```

[92]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a␣
 ↪dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow,␣
 ↪price_standardized))
X.shape
```

[92]:
```
(7000, 5262)
```

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.

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.

EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects

Build the data matrix using these features

school_state : categorical data (one hot encoding)

```
        <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
```

[93]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()),␣
 ↪lowercase=False, binary=True)
vectorizer.fit(project_data['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)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (7000, 9)
```

[94]:

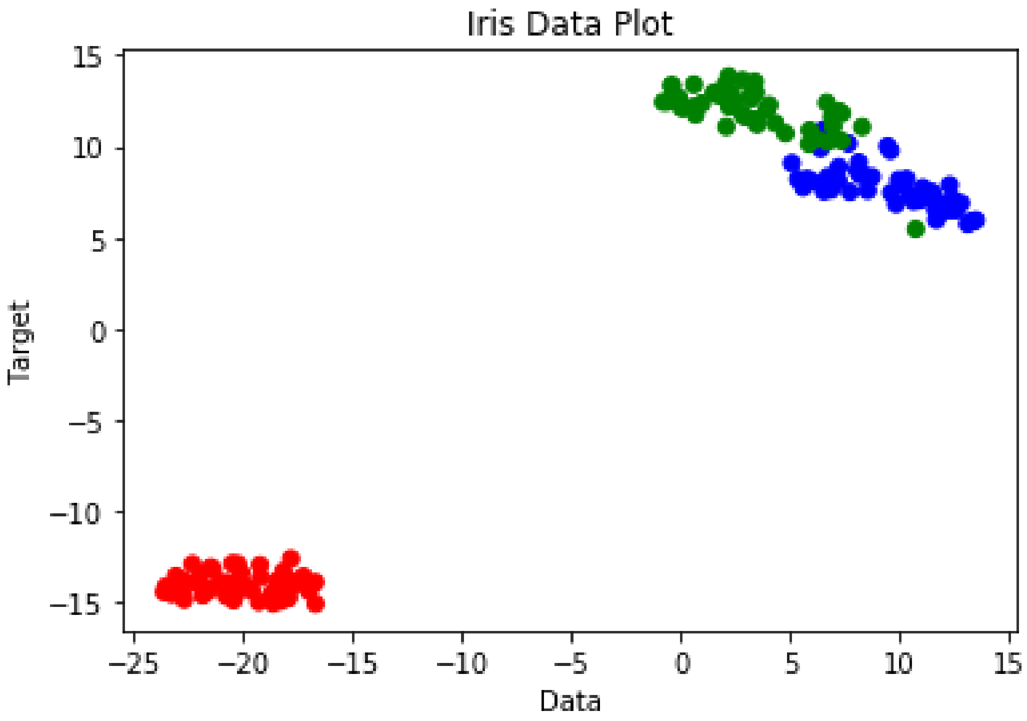
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()),␣
 ↪lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)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

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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (7000, 9)
```

[95]:

```
# 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']
plt.title('Iris Data Plot')
plt.xlabel('Data')
plt.ylabel('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() will convert the sparse matrix into␣
 ↪dense matrix
```

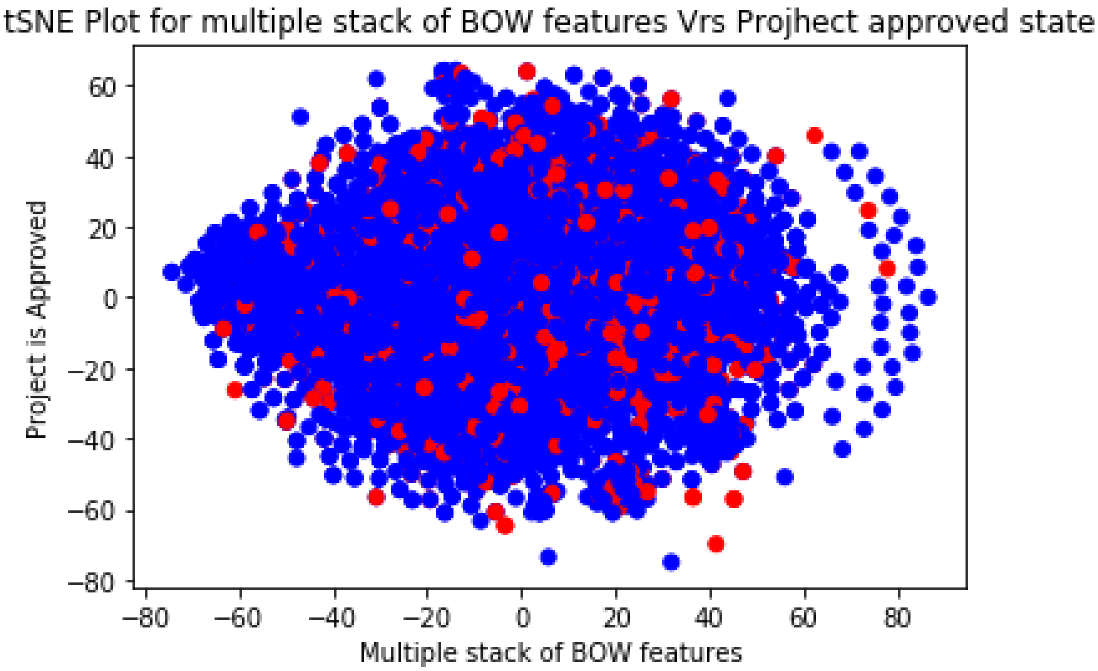
```
for_tsne = np.hstack((X_embedding, y.reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne,␣
 ↪columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'],␣
 ↪c=for_tsne_df['Score'].apply(lambda x: colors[x]))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



Apply TSNE for BOW Multiple Features

[96]:
```
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

# please check why to plot on the screen.
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow,␣
 ↪price_standardized))
#y = price_standardized
x = hstack((text_bow, project_title_bow, project_grade_category_bow, \
            project_subject_categories_bow, school_state_bow,price_bow,␣
 ↪teacher_prefix_bow))
y = project_data['project_is_approved']
```

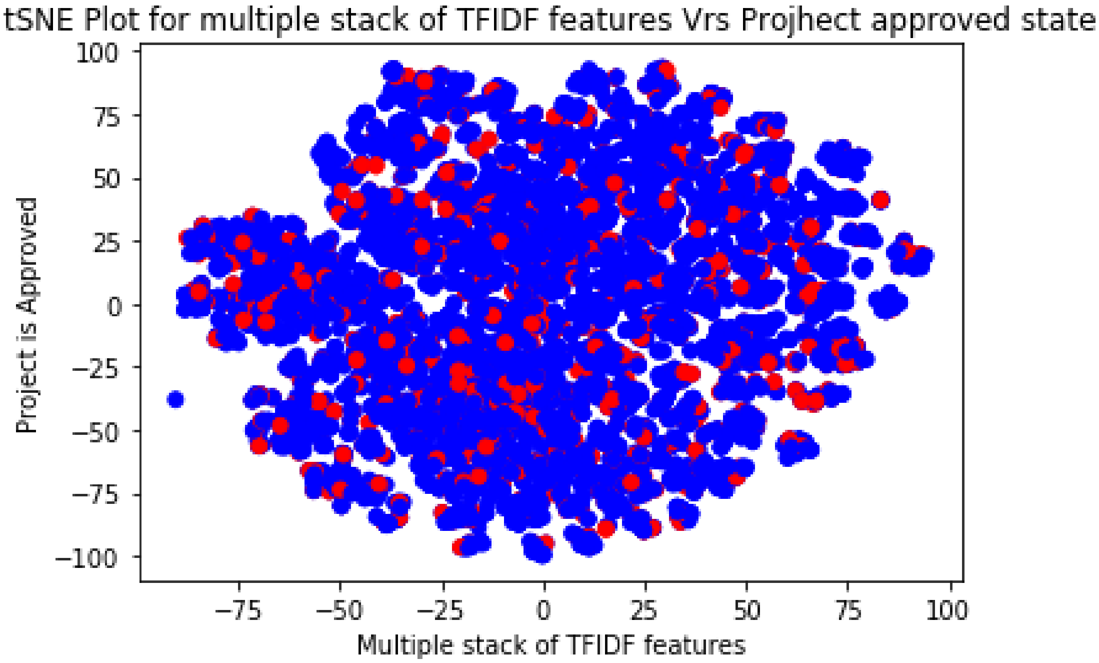
```
plt.title('tSNE Plot for multiple stack of BOW features Vrs Projhect approved␣
 ↪state')
plt.xlabel('Multiple stack of BOW features')
plt.ylabel('Project is Approved')

tsne = TSNE(n_components=2, perplexity=10, learning_rate=100)

X_embedding = tsne.fit_transform(x.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.
 ↪fit_transform(x.toarray()) ,
# toarray() will convert the sparse matrix into dense matrix

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



Apply TSNE for TFIDF Multiple Features

[97]:
```
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
```

```python
import pandas as pd
import matplotlib.pyplot as plt

# please check why to plot on the screen.
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
#y = price_standardized
x = hstack((text_tfidf, project_title_tfidf, project_grade_category_tfidf, teacher_prefix_tfidf, \
            school_state_tfidf,project_subject_categories_tfidf,price_tfidf))
            #,price_standardized))
y = project_data['project_is_approved']

plt.title('tSNE Plot for multiple stack of TFIDF features Vrs Projhect approved state')
plt.xlabel('Multiple stack of TFIDF features')
plt.ylabel('Project is Approved')

tsne = TSNE(n_components=2, perplexity=10, learning_rate=100)

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

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

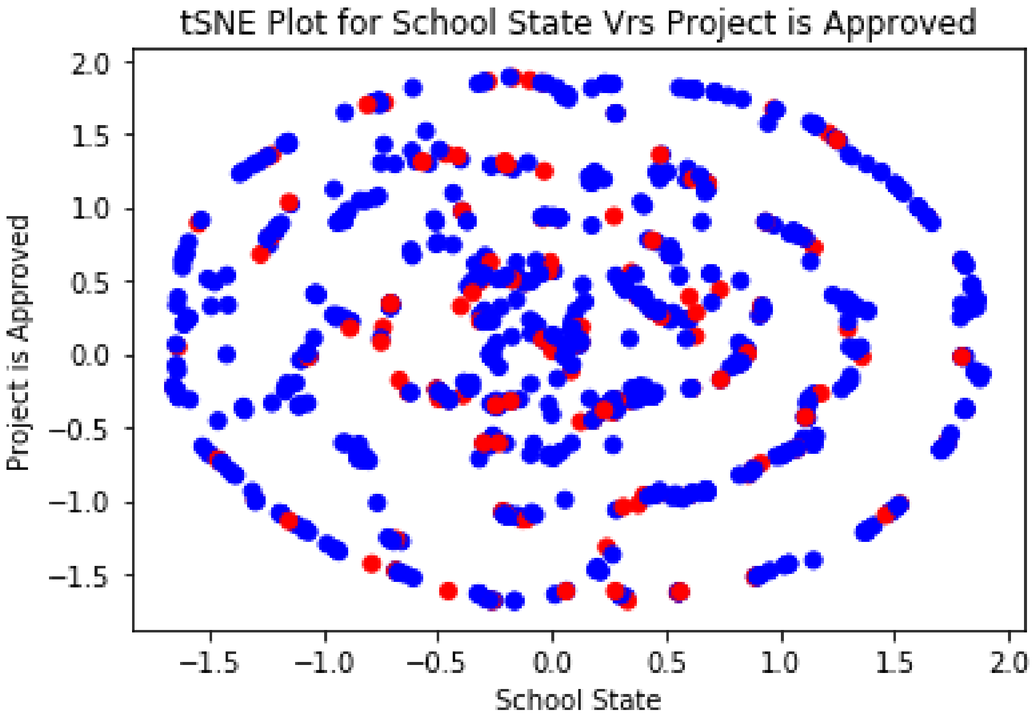


Apply TSNE for School State Vrs Project Approved State

[98]:

```
import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
# please check why do plot on the screen.
x = school_state_one_hot[0:1000]
y = (project_data['project_is_approved'])[0:1000]

plt.title('tSNE Plot for School State Vrs Project is Approved')
plt.xlabel('School State')
plt.ylabel('Project is Approved')
tsne = TSNE(n_components=2, perplexity=20, learning_rate=100,n_iter=1000)

X_embedding = tsne.fit_transform(x.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.
 ↪fit_transform(x.toarray()) , .toarray() will convert the sparse matrix into␣
 ↪dense matrix

for_tsne = np.hstack((X_embedding,  y.to_numpy().reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne,␣
 ↪columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue', 2:'green'}
```

```
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'],␣
 ↪c=for_tsne_df['Score'].apply(lambda x: colors[x]))
plt.show()
```



2.1 TSNE with BOW encoding of project_title feature

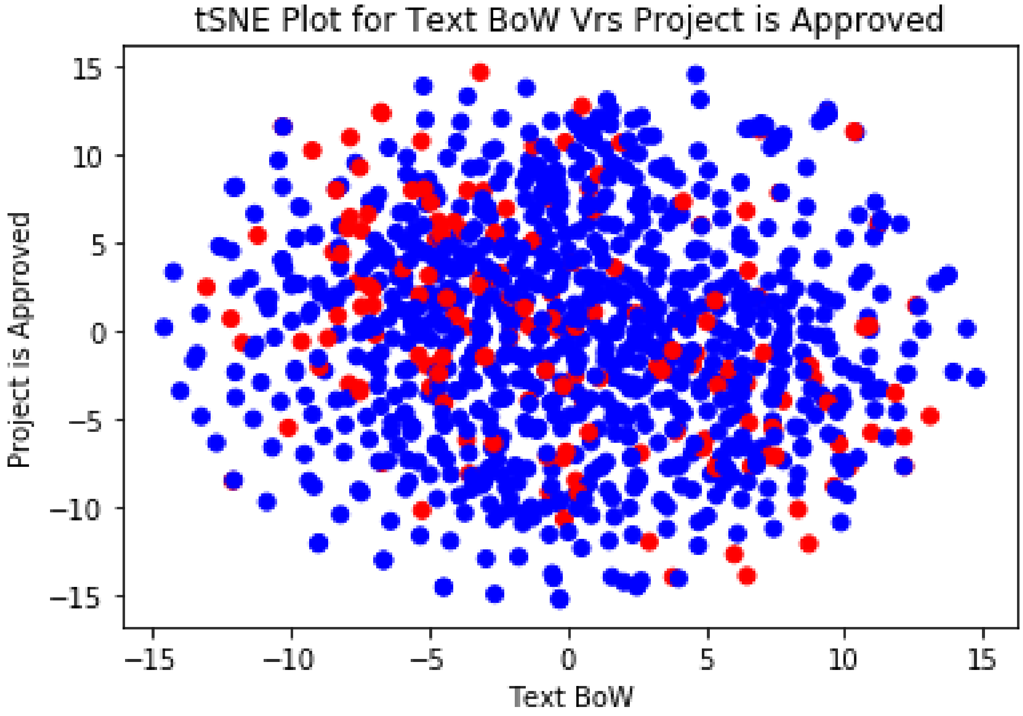
[99]:

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

[100]:

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

# please check why do plot on the screen.
x= text_bow[0:1000]
y = (project_data['project_is_approved'])[0:1000]
```

```
tsne = TSNE(n_components=2, perplexity=30, learning_rate=100,n_iter=1000)

X_embedding = tsne.fit_transform(x.toarray())


plt.title('tSNE Plot for Text BoW Vrs Project is Approved')
plt.xlabel('Text BoW')
plt.ylabel('Project is Approved')

# if x is a sparse matrix you need to pass it as X_embedding = tsne.
 ↪fit_transform(x.toarray()) , .toarray() will convert the sparse matrix into␣
 ↪dense matrix
for_tsne = np.hstack((X_embedding, y.to_numpy().reshape(-1,1)))
for_tsne_df = pd.DataFrame(data=for_tsne,␣
 ↪columns=['Dimension_x','Dimension_y','Score'])
colors = {0:'red', 1:'blue', 2:'green'}
plt.scatter(for_tsne_df['Dimension_x'], for_tsne_df['Dimension_y'],␣
 ↪c=for_tsne_df['Score'].apply(lambda x: colors[x]))
plt.show()
```



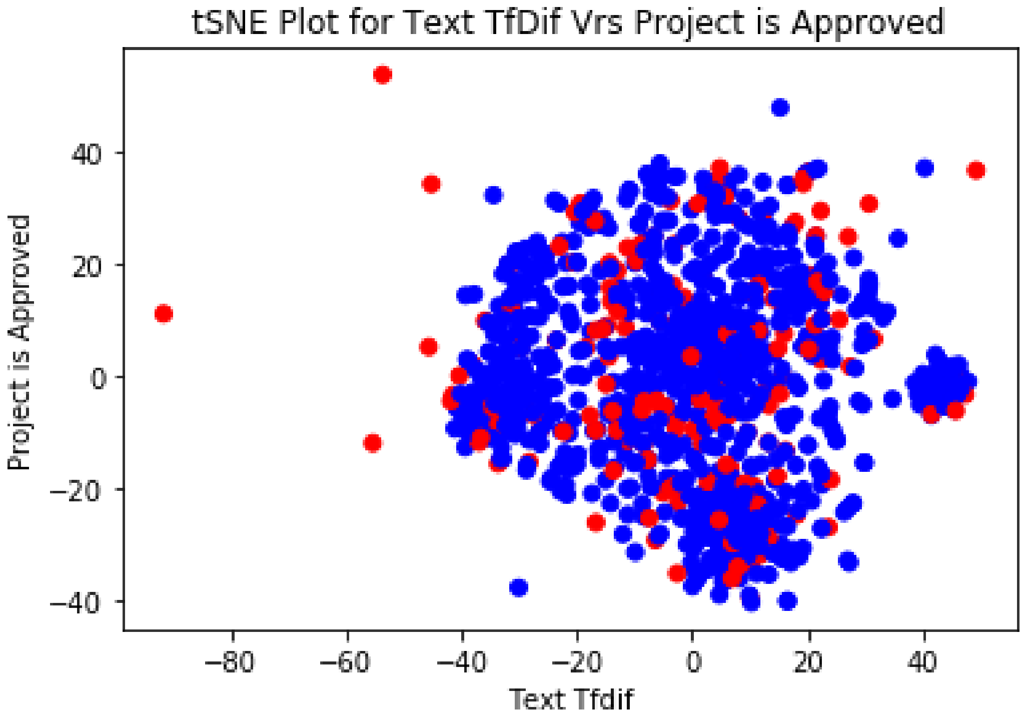
2.2 TSNE with TFIDF encoding of project_title feature

[101]:
```
# 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
```

[102]:

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

# please check why to plot on the screen.
x = text_tfidf[0:1000]
y = (project_data['project_is_approved'])[0:1000]

plt.title('tSNE Plot for Text TfDif Vrs Project is Approved')
plt.xlabel('Text Tfdif')
plt.ylabel('Project is Approved')

tsne = TSNE(n_components=2, perplexity=30, learning_rate=100,n_iter=1000)

X_embedding = tsne.fit_transform(x.toarray())
# if x is a sparse matrix you need to pass it as X_embedding = tsne.
 ↪fit_transform(x.toarray()) ,
# toarray() will convert the sparse matrix into dense matrix

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



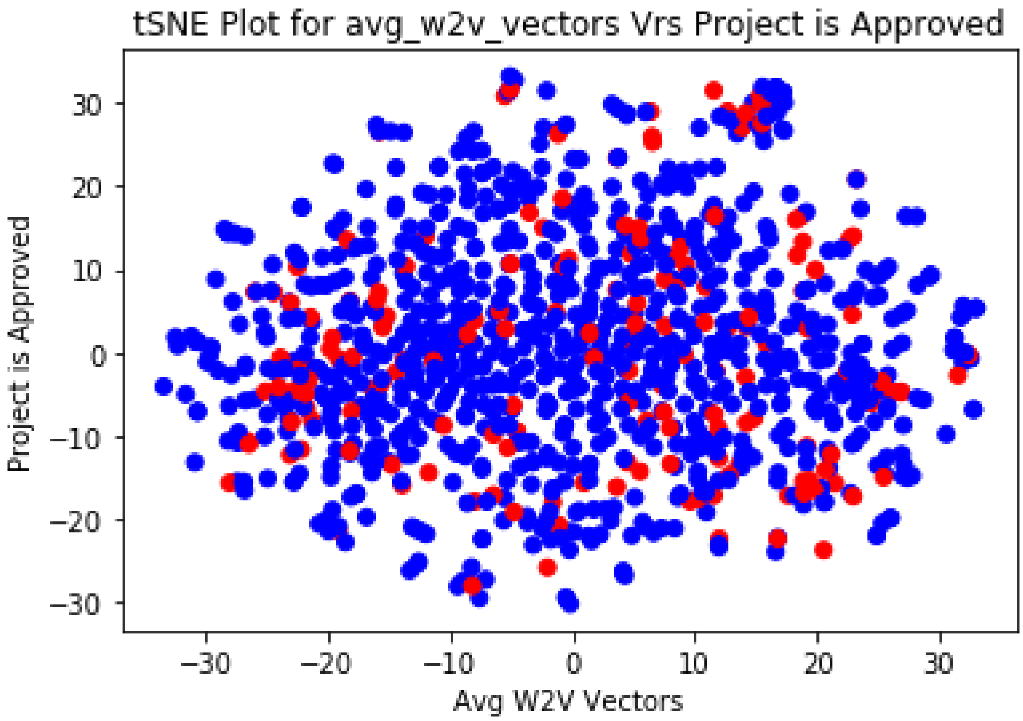
2.3 TSNE with `AVG W2V` encoding of `project_title` feature

[103]:

```
# 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
```

[104]:

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

# please check why do plot on the screen.
x = avg_w2v_vectors[0:1000]
y = (project_data['project_is_approved'])[0:1000]


plt.title('tSNE Plot for avg_w2v_vectors Vrs Project is Approved')
plt.xlabel('Avg W2V Vectors')
plt.ylabel('Project is Approved')
tsne = TSNE(n_components=2, perplexity=30, learning_rate=100 ,n_iter=1000)

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() will convert the sparse matrix into␣
 ↪dense matrix

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



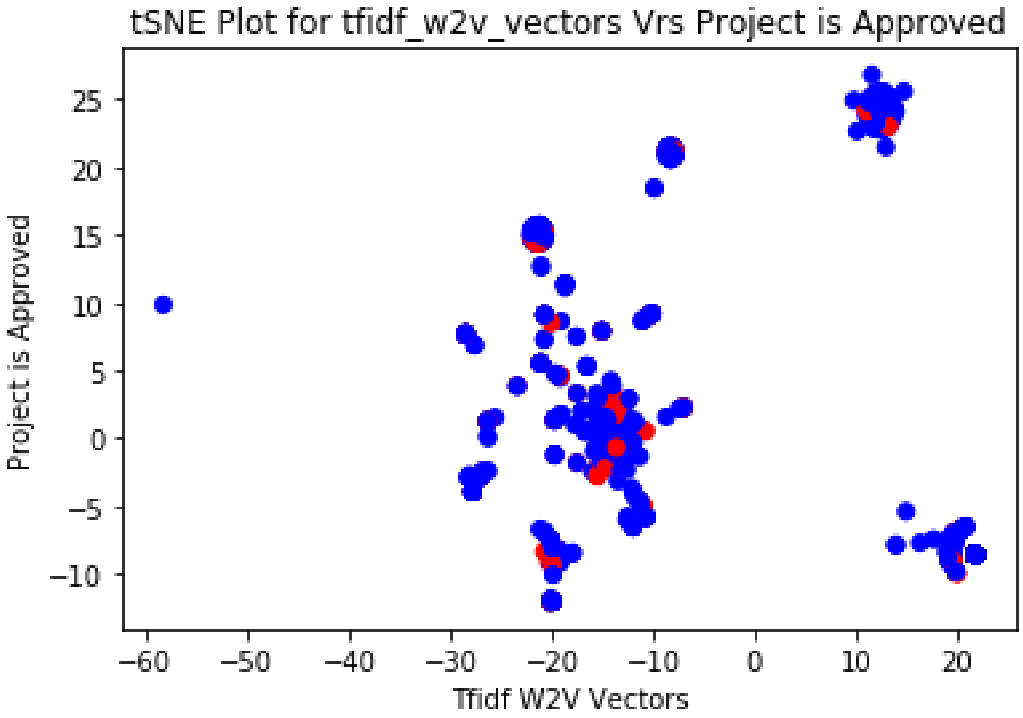
2.4 TSNE with TFIDF Weighted W2V encoding of project_title feature

[105]:

```
# 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
```

[106]:

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

# please check why do plot on the screen.
x = tfidf_w2v_vectors[0:1000]
y = (project_data['project_is_approved'])[0:1000]

plt.title('tSNE Plot for tfidf_w2v_vectors Vrs Project is Approved')
plt.xlabel('Tfidf W2V Vectors')
plt.ylabel('Project is Approved')

tsne = TSNE(n_components=2, perplexity=30, learning_rate=100,n_iter=1000)

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() will convert the sparse matrix into␣
 ↪dense matrix

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



2.2.5 tSNE for all features

2.5 Summary

#tSNE A you can see above, various tSNE plots are drawn. tSNE is way better than PCA, if to segregate densed data into various clusters. tSNE are plotted for target data which is project_is_approved in above example.

In above graphs, data is still not clustered properly ad I had to reduce the data to 1000 rows, learning rate to 200 and perplexity to 30. Please try regenerating the graph with full population, keep increasing the learning rate and perplexity till you see the graph stable.

2.3 My understanding about TSNE:

2.3.1 TSNE

Embedding: - Picking a point from high dim space and mapping it into a low dim space. tSNE preserves distance of the local points/features. points which are farther away, tSNE will not guarantee to preserve the distance. but tSNE will surely preserve nearer distance among points. Crowding Problem: We cannot perfectly embed all the nearer points as we earlier mentioned =, tSNE should preserve distance of local points in 1 D space. Sometime issue comes when we map all nearer points to local space from multiple dim to one dim space. T-Distribution is used to resolve crowding problem, but not always to solve crowding problem. Refer https://distll.pub or tSNE, tSNE is iterative algorithm. at each state, it tries to find the space to put points in space. Which is called embedding. so, we use step size - number of iteration.at each iteration it tries to find better place for points. We should keep iterating until shape is stable. Perplexity: number of neighbors

- if I want to preserve distances of my 5 points, so my perplexity will be 5. so perplexity says how many data points do I need to preserve. perplexity very low and very high might not work. make sure you keep trying changing to see if your data is stable.so always run tSNE with different parameters to see if your shape is stable. running same data might give you diff result since tSNE behaves randomly. tSNE expands dense cultures and shrinks sparse clusters. Drawback of tSNE. tSNE does not preserve distances among clusters as well.

[ ]: