

Projects_Auto_Approval_tSNE

June 4, 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

- How to increase the consistency of project vetting across different volunteers to improve t

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
<code>project_id</code>	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 <code>project_id</code> value from the <code>train.csv</code> file. Example: p036502
description	Description 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
<code>project_is_approved</code>	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(20000)
resource_data = resource_data.head(20000)

[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 (20000, 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  897464ce9ddc600bcd1151f324dd63a      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 (20000, 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 counts of approved and non-approved projects.
print(y_value_counts)

print("Number of projects that are approved for funding ", y_value_counts[1], "\n",
      "→", "(", (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()

```

```

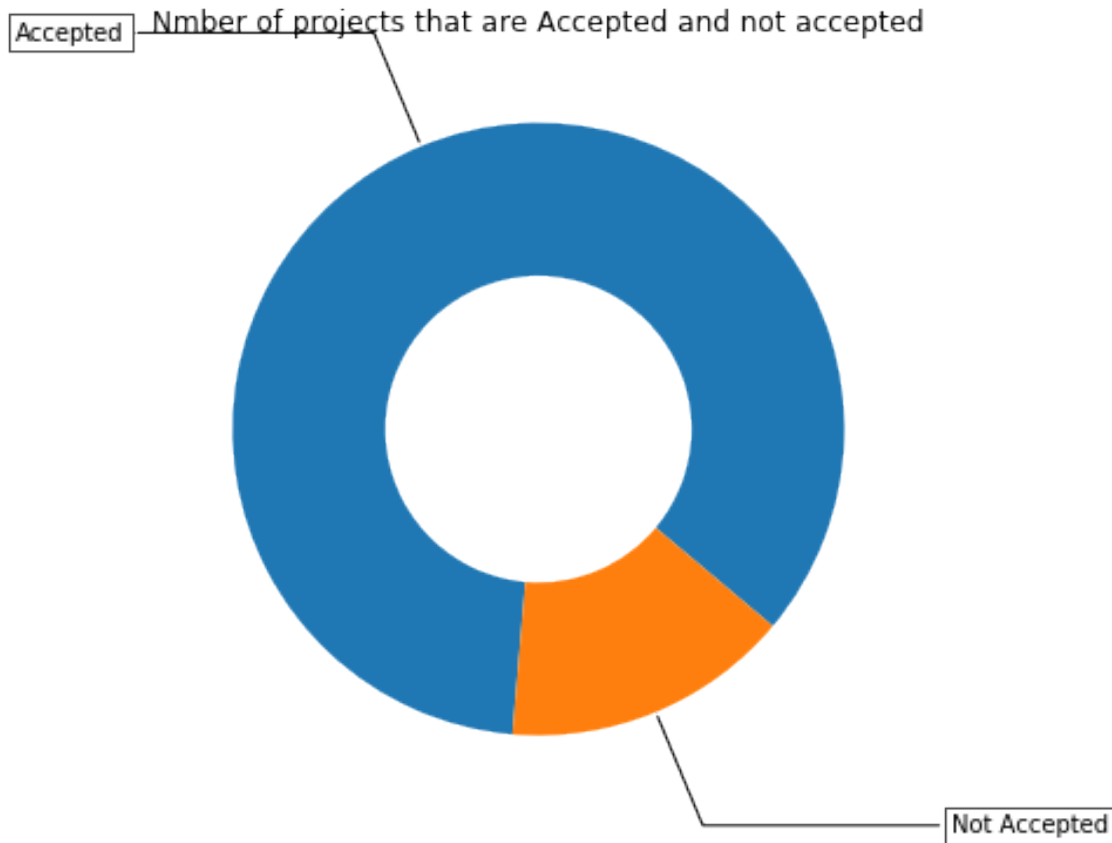
1    16951
0     3049

```

Name: project_is_approved, dtype: int64

Number of projects thar are approved for funding 16951 , (84.755 %)

Number of projects thar are not approved for funding 3049 , (15.245000000000001 %)



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\nndata = [ 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)\n\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
26	MT	0.731707
50	WY	0.760000
7	DC	0.760000
37	OR	0.795082
13	ID	0.798387

States with highest % approvals

	state_code	num_proposals
17	KY	0.883117
16	KS	0.891473
32	NM	0.924731
8	DE	0.927273
28	ND	0.961538

```
[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()

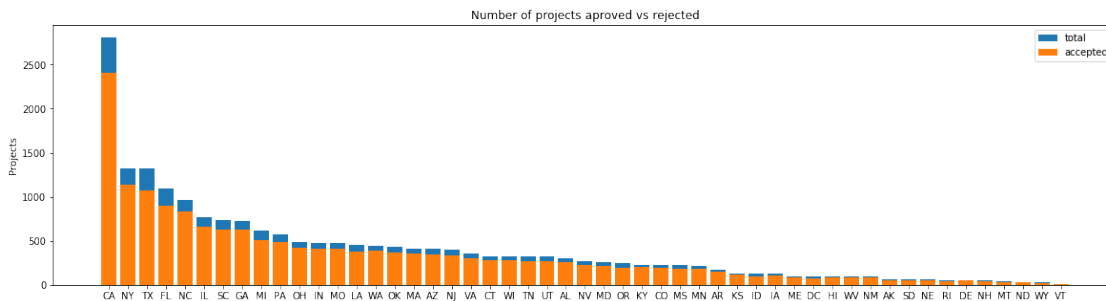
      # Pandas dataframe group by count: https://stackoverflow.com/a/19385591/
      ↳ 4084039
      temp['total'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'total':
      ↳ 'count'})).reset_index()['total']
      temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':
      ↳ 'mean'})).reset_index()['Avg']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

```
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	2401	2808	0.855057
34	NY	1135	1320	0.859848
43	TX	1073	1316	0.815350
9	FL	895	1093	0.818847
27	NC	829	967	0.857291

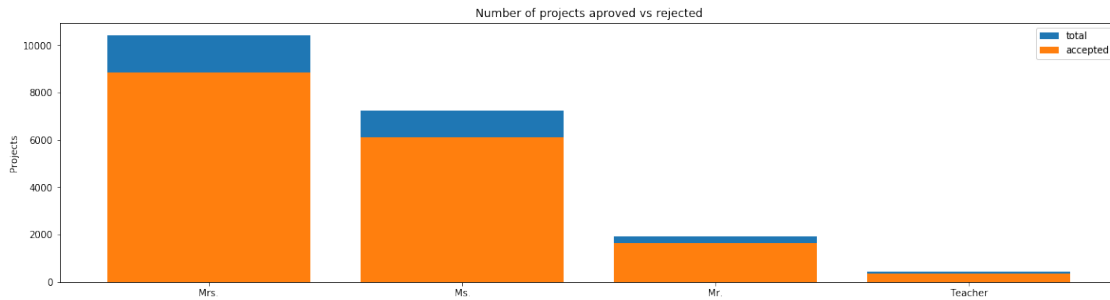
=====

	school_state	project_is_approved	total	Avg
30	NH	45	51	0.882353
26	MT	30	41	0.731707
28	ND	25	26	0.961538
50	WY	19	25	0.760000
46	VT	9	11	0.818182

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.	8858	10420	0.850096
2	Ms.	6111	7222	0.846164
0	Mr.	1626	1908	0.852201
3	Teacher	355	449	0.790646

```
=====
```

	teacher_prefix	project_is_approved	total	Avg
1	Mrs.	8858	10420	0.850096
2	Ms.	6111	7222	0.846164
0	Mr.	1626	1908	0.852201
3	Teacher	355	449	0.790646

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	6901	8171	0.844572
0	Grades 3-5	5819	6785	0.857627
1	Grades 6-8	2585	3079	0.839558
2	Grades 9-12	1646	1965	0.837659

```
=====
```

	project_grade_category	project_is_approved	total	Avg
3	Grades PreK-2	6901	8171	0.844572
0	Grades 3-5	5819	6785	0.857627
1	Grades 6-8	2585	3079	0.839558
2	Grades 9-12	1646	1965	0.837659

2.0.4 1.2.4 Univariate Analysis: project_subject_categories

```
[13]: categories = 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 categories:
    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 category 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 placing 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  897464ce9ddc600bcd1151f324dd63a    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

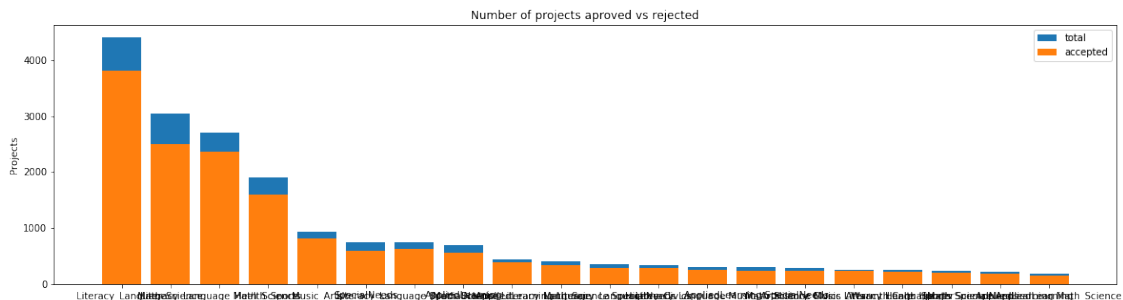
project_essay_4                  project_resource_summary \
0                               NaN  My students need opportunities to practice beg...
1                               NaN  My students need a projector to help with view...

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	3804	4403	0.863956
31	Math_Science	2493	3039	0.820336
27	Literacy_Language Math_Science	2369	2707	0.875139
8	Health_Sports	1605	1896	0.846519
39	Music_Arts	807	929	0.868676

```
=====
```

	clean_categories	project_is_approved	total	Avg
19	History_Civics Literacy_Language	239	257	0.929961
49	Warmth Care_Hunger	227	245	0.926531
14	Health_Sports SpecialNeeds	199	240	0.829167
32	Math_Science AppliedLearning	180	217	0.829493
4	AppliedLearning Math_Science	151	189	0.798942

[16]: *# count of all the words in corpus python: <https://stackoverflow.com/a/22898595/>*
→4084039

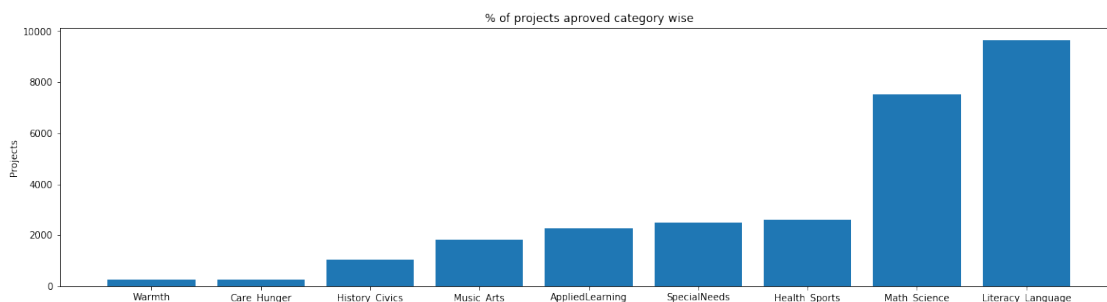
```
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

[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           :      264
Care_Hunger      :      264
History_Civics   :     1057
Music_Arts       :     1827
```

```

AppliedLearning      :      2271
SpecialNeeds         :      2504
Health_Sports        :      2620
Math_Science         :      7534
Literacy_Language    :      9644

```

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  897464ce9ddc600bcd1151f324dd63a      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

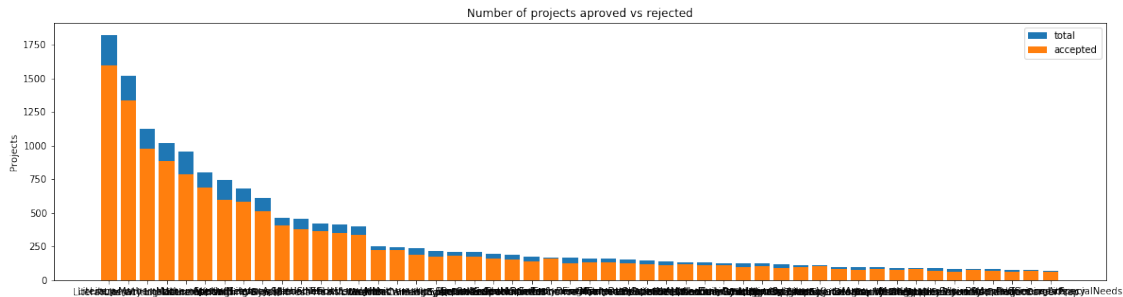
project_essay_4                                project_resource_summary \
0          NaN My students need opportunities to practice beg...
1          NaN My students need a projector to help with view...

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



	clean_subcategories	project_is_approved	total	Avg
265	Literacy	1596	1822	0.875960
267	Literacy Mathematics	1337	1519	0.880184
279	Literature_Writing Mathematics	979	1124	0.870996
266	Literacy Literature_Writing	885	1023	0.865103
289	Mathematics	790	955	0.827225
=====				
	clean_subcategories	project_is_approved	total	Avg
110	ESL	74	85	0.870588

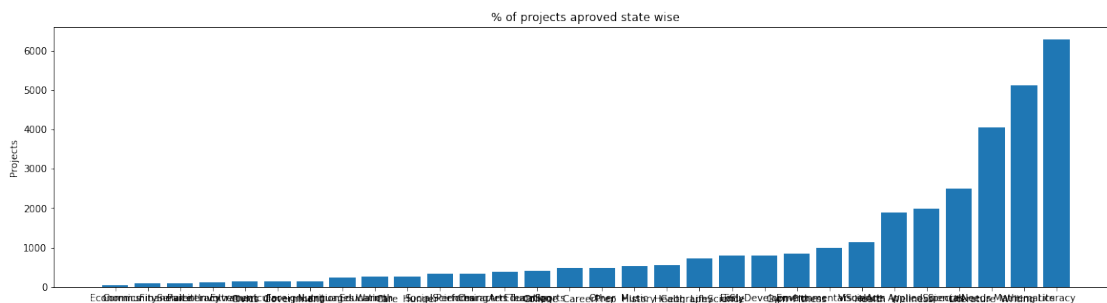
3	AppliedSciences College_CareerPrep	67	84	0.797619
165	EnvironmentalScience Literacy	64	77	0.831169
323	PerformingArts	69	76	0.907895
24	AppliedSciences SpecialNeeds	64	73	0.876712

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

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

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

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



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

Economics	:	58
CommunityService	:	103
FinancialLiteracy	:	106
ParentInvolvement	:	127
Extracurricular	:	137
Civics_Government	:	146
ForeignLanguages	:	150

NutritionEducation	:	246
Warmth	:	264
Care_Hunger	:	264
SocialSciences	:	326
PerformingArts	:	348
CharacterEducation	:	379
TeamSports	:	404
College_CareerPrep	:	475
Other	:	476
Music	:	540
History_Geography	:	566
Health_LifeScience	:	724
ESL	:	787
EarlyDevelopment	:	792
Gym_Fitness	:	840
EnvironmentalScience	:	994
VisualArts	:	1135
Health_Wellness	:	1883
AppliedSciences	:	1988
SpecialNeeds	:	2504
Literature_Writing	:	4056
Mathematics	:	5116
Literacy	:	6280

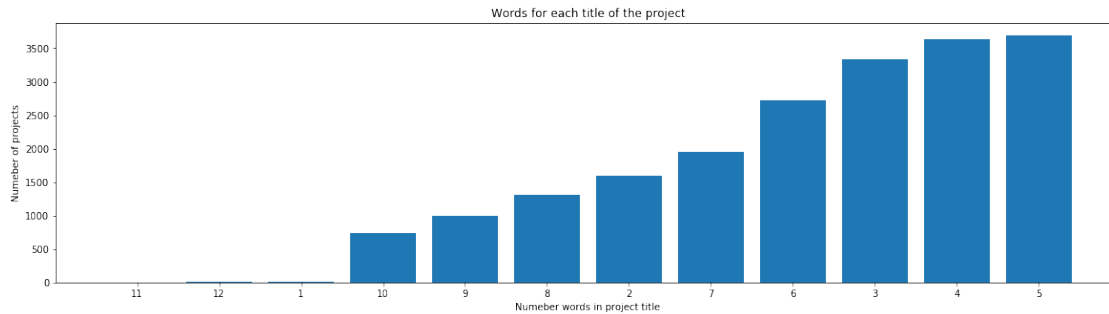
2.0.6 1.2.6 Univariate Analysis: Text features (Title)

[25]: *#How to calculate number of words in a string in DataFrame: <https://stackoverflow.com/a/37483537/4084039>*

```
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))

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

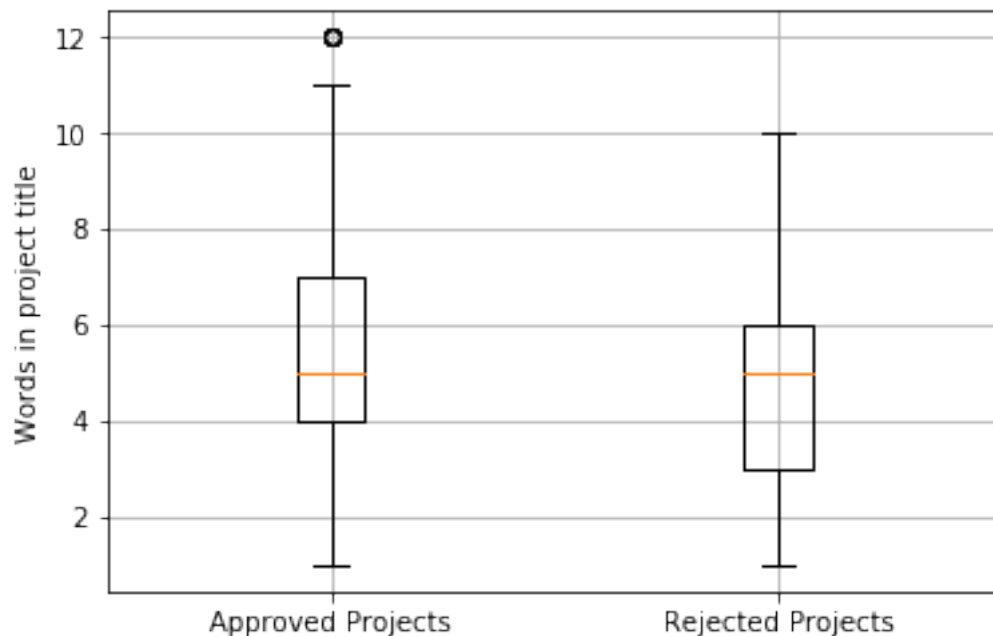
plt.ylabel('Numeber of projects')
plt.xlabel('Numeber words in project title')
plt.title('Words for each title of the project')
plt.xticks(ind, list(word_dict.keys()))
plt.show()
```



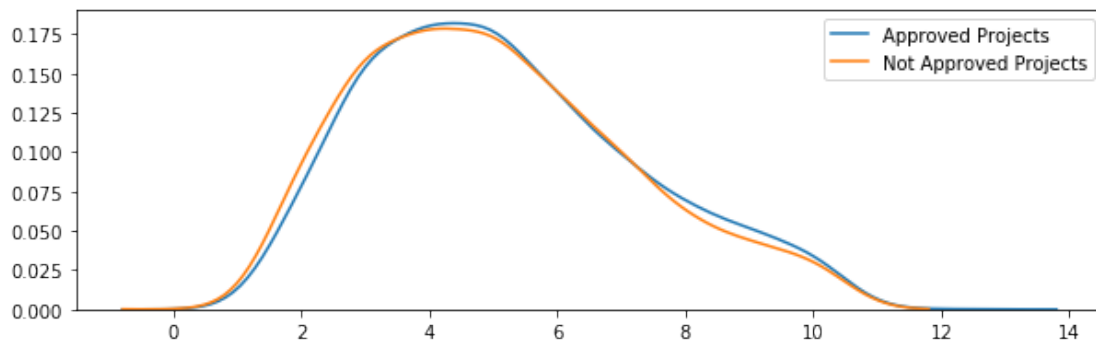
```
[26]: approved_title_word_count =
    →project_data[project_data['project_is_approved']==1]['project_title'].str.
    →split().apply(len)
approved_title_word_count = approved_title_word_count.values

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()
```



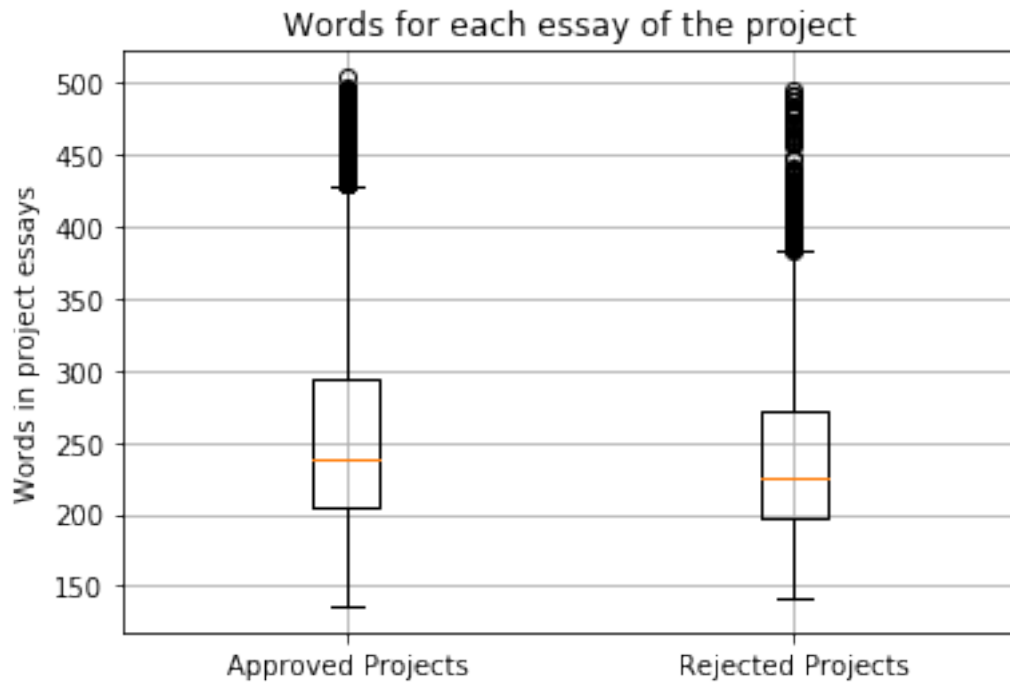
2.0.7 1.2.7 Univariate Analysis: Text features (Project Essay's)

```
[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)

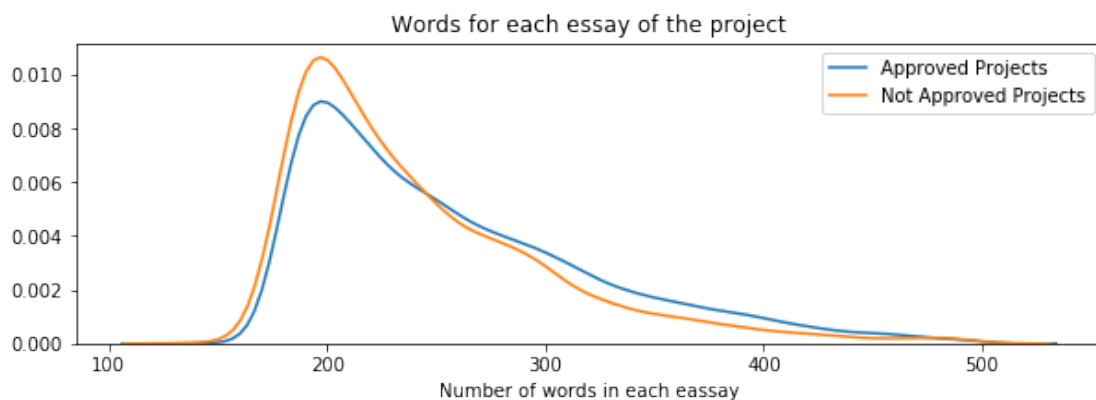
[30]: approved_word_count = \
    →project_data[project_data['project_is_approved']==1]['essay'].str.split().
    →apply(len)
approved_word_count = approved_word_count.values

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)
```

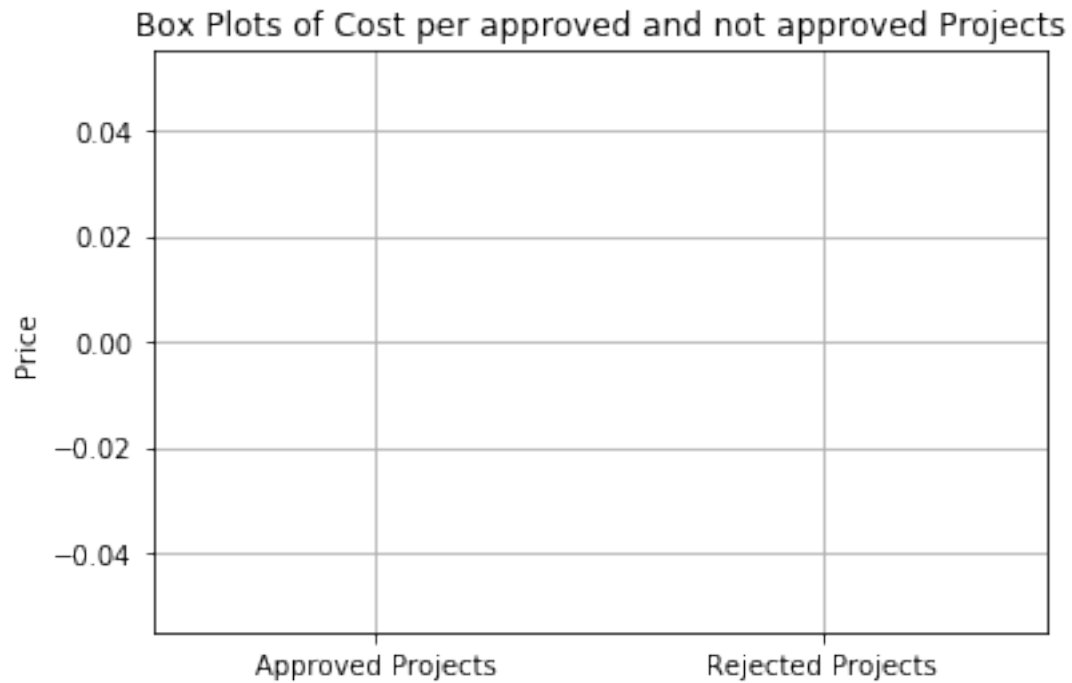
```
[34]:      id      price  quantity
0  p000147    13.13         25
1  p000157   3508.32          9
```

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

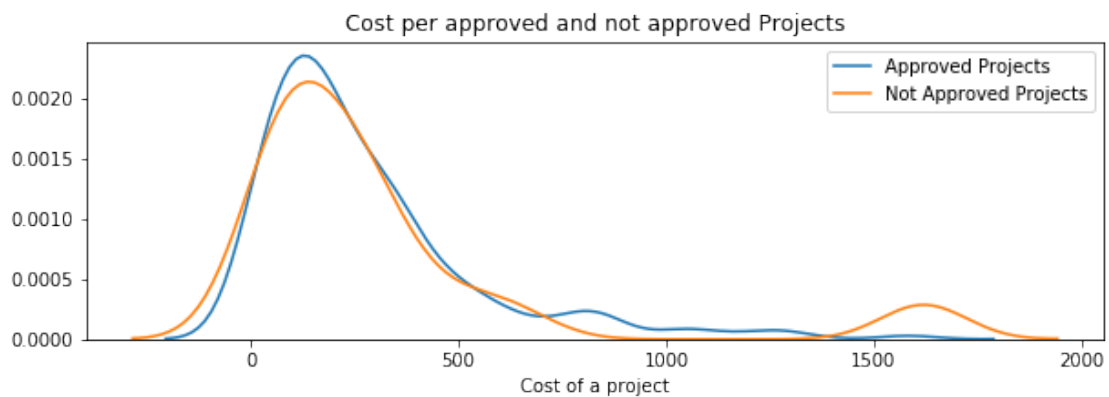
```
[36]: approved_price = project_data[project_data['project_is_approved']==1]['price'].
      ↪values

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

```
[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"]

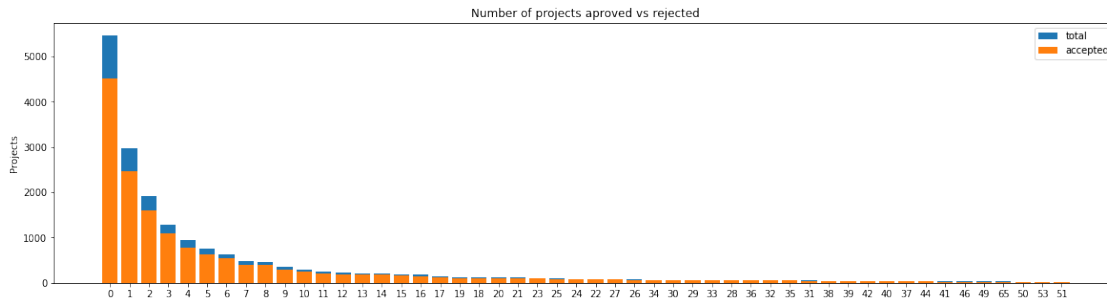
for i in range(0,101,5):
    x.add_row([i,np.round(np.percentile(approved_price,i), 3), np.round(np.
→percentile(rejected_price,i), 3)])
print(x)
```

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	4524	5466
1	1	2469	2971
2	2	1603	1917
3	3	1089	1288
4	4	787	938

Avg

0	0.827662
1	0.831033
2	0.836202
3	0.845497
4	0.839019

```
=====
```

teacher_number_of_previously_posted_projects	project_is_approved	total	\
49	49	24	29
65	65	24	28
50	50	27	27
53	53	24	26
51	51	23	25

Avg

49	0.827586
65	0.857143
50	1.000000
53	0.923077
51	0.920000

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 approval or 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	14401	17175	0.838486
1	True	2550	2825	0.902655

	digit_in_resource_summary	project_is_approved	total	Avg
0	False	14401	17175	0.838486
1	True	2550	2825	0.902655

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  c90749f5d961fff158d4b4d1e7dc665fc  Mrs.
1      140945  p258326  897464ce9ddc600bcd1151f324dd63a    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 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\nnnannan

=====

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[2000])
print(sent)
print("="*50)
```

Describing my students is not an easy task. Many would say that they are

inspirational, creative, and hard-working. They are all unique - unique in their interests, their learning, their abilities, and so much more. What they all have in common is their desire to learn each day, despite difficulties that they encounter. \r\nOur classroom is amazing - because we understand that everyone learns at their own pace. As the teacher, I pride myself in making sure my students are always engaged, motivated, and inspired to create their own learning! \r\nThis project is to help my students choose seating that is more appropriate for them, developmentally. Many students tire of sitting in chairs during lessons, and having different seats available helps to keep them engaged and learning.\r\nFlexible seating is important in our classroom, as many of our students struggle with attention, focus, and engagement. We currently have stability balls for seating, as well as regular chairs, but these stools will help students who have trouble with balance, or find it difficult to sit on a stability ball for a long period of time. We are excited to try these stools as a part of our engaging classroom community!nannan

=====

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

Describing my students is not an easy task. Many would say that they are inspirational, creative, and hard-working. They are all unique - unique in their interests, their learning, their abilities, and so much more. What they all have in common is their desire to learn each day, despite difficulties that they encounter. Our classroom is amazing - because we understand that everyone learns at their own pace. As the teacher, I pride myself in making sure my students are always engaged, motivated, and inspired to create their own learning! This project is to help my students choose seating that is more appropriate for them, developmentally. Many students tire of sitting in chairs during lessons, and having different seats available helps to keep them engaged and learning. Flexible seating is important in our classroom, as many of our students struggle with attention, focus, and engagement. We currently have stability balls for seating, as well as regular chairs, but these stools will help students who have trouble with balance, or find it difficult to sit on a stability ball for a long period of time. We are excited to try these stools as a part of our engaging classroom community!nannan

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

Describing my students is not an easy task Many would say that they are inspirational creative and hard working They are all unique unique in their

interests their learning their abilities and so much more What they all have in common is their desire to learn each day despite difficulties that they encounter Our classroom is amazing because we understand that everyone learns at their own pace As the teacher I pride myself in making sure my students are always engaged motivated and inspired to create their own learning This project is to help my students choose seating that is more appropriate for them developmentally Many students tire of sitting in chairs during lessons and having different seats available helps to keep them engaged and learning Flexible seating is important in our classroom as many of our students struggle with attention focus and engagement We currently have stability balls for seating as well as regular chairs but these stools will help students who have trouble with balance or find it difficult to sit on a stability ball for a long period of time We are excited to try these stools as a part of our engaging classroom community 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]: # Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\r', ' ')
    sent = sent.replace('\n', ' ')
    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%|| 20000/20000 [00:11<00:00, 1769.64it/s]

```
[50]: # after preprocesing
preprocessed_essays[10000]
```

```
[50]: 'my kids caring curious individuals they pretty much clap time science love
figuring world around works they value community overcome economic disadvantages
special needs active learners classroom while rural district one largest area
even small community mindset knowing helping neighbors our class believes
capable effort perseverance great things wi looking set book clubs involve
student family every night give reading 20 minutes assignment i would like
change assignment 10 minutes family member reading 10 minutes student reading
aloud after reading log onto google classroom page submit reflection questions
read i choosing two classic books engaging students adults these books personal
favorites contributed rich diverse discussions previous classes i think giving
family members topics discuss child promote type questioning analysis promote
classroom families able discuss books using google classroom post comment
features topics i post i able measure involvement many posts family makes i
would also like offer coffee shop times kids parents come classroom use
chromebooks not resources home participate i look forward kids parents
interacting great literature nannan'
```

1.3.2 Project title Text

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

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

Bouncing Our Wiggles and Worries Away!

=====

```
[53]: # \r \n \t remove from string python: http://texthandler.com/info/
      ↪ remove-line-breaks-python/
sent = sent.replace('\r', ' ')
sent = sent.replace('\n', ' ')
sent = sent.replace('\t', ' ')
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 sentence in tqdm(project_data['project_title'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\r', ' ')
    sent = sent.replace('\n', ' ')
    sent = sent.replace('\t', ' ')
    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%|| 20000/20000 [00:00<00:00, 38230.46it/s]

```
[56]: # after preprocessing
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 encoding ", 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 (20000, 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', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages',
'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences',
'PerformingArts', 'CharacterEducation', 'TeamSports', 'College_CareerPrep',
'Other', 'Music', 'History_Geography', 'Health_LifeScience', 'ESL',
'EarlyDevelopment', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts',
'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing',
'Mathematics', 'Literacy']
Shape of matrix after one hot encodig (20000, 30)
```

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

2.2.2 1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

```
[61]: # 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 (20000, 8465)

1.4.2.2 Bag of Words on project_title

```
[62]: # you can vectorize the title also
# before you vectorize the title make sure you preprocess it
# data below is preprocessed already.
```

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

```
[64]: # 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(project_data['project_title'])
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

Shape of matrix after one hot encodig (20000, 1116)

```
[65]: # 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)
text_bow = vectorizer.fit_transform(preprocessed_project_title)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

Shape of matrix after one hot encodig (20000, 1108)

1.4.2.3 TFIDF vectorizer

```
[66]: 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 (20000, 8465)

1.4.2.4 TFIDF Vectorizer on project_title

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

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

Shape of matrix after one hot encodig (20000, 1108)

1.4.2.5 Using Pretrained Models: Avg W2V

```
[69]: '''
      # 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)

'''

```

```
[69]: '\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 preprocod_texts:\n
words.extend(i.split(\' \''))\n\nfor i in preprocod_titles:\n
words.extend(i.split(\' \''))\n\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'
```

```
[70]: # 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())
```

```
[71]: # 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%|| 20000/20000 [00:06<00:00, 3060.03it/s]

20000

300

1.4.2.6 Using Pretrained Models: AVG W2V on project_title

[72]: *# Similarly you can vectorize for title also*

```
[73]: # 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%|| 20000/20000 [00:00<00:00, 57480.71it/s]

20000

300

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

```
[74]: # 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())
```

```
[75]: # 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%|| 20000/20000 [00:44<00:00, 448.38it/s]

20000

300

1.4.2.9 Using Pretrained Models: TFIDF weighted W2V on project_title

[76]: # Similarly you can vectorize for title also

[77]: # 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())

[78]: # 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%|| 20000/20000 [00:00<00:00, 74088.67it/s]

20000

300

2.2.3 1.4.3 Vectorizing Numerical features

```
[79]: # check this one: https://www.youtube.com/watch?v=0HQq0cln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/
→sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScaler.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 mean and variance.
price_standardized = price_scalar.transform(project_data['price'].values.
→reshape(-1, 1))
```

Mean : 282.61577092511016, Standard deviation : 286.6269779730071

```
[80]: price_standardized
```

```
[80]: 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 categorical, text, numerical vectors

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

```
(20000, 9)
(20000, 30)
(20000, 1108)
(20000, 1)
```

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

```
[82]: (20000, 1148)
```

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>
```

 Note 1: The TSNE accepts only dense matrices

 Note 2: Consider only 5k to 6k data points to avoid memory issues. If y

```
[83]: # 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 (20000, 9)
```

```
[84]: # 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 (20000, 9)
```

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

iris = datasets.load_iris()

x = iris['data']
y = iris['target']

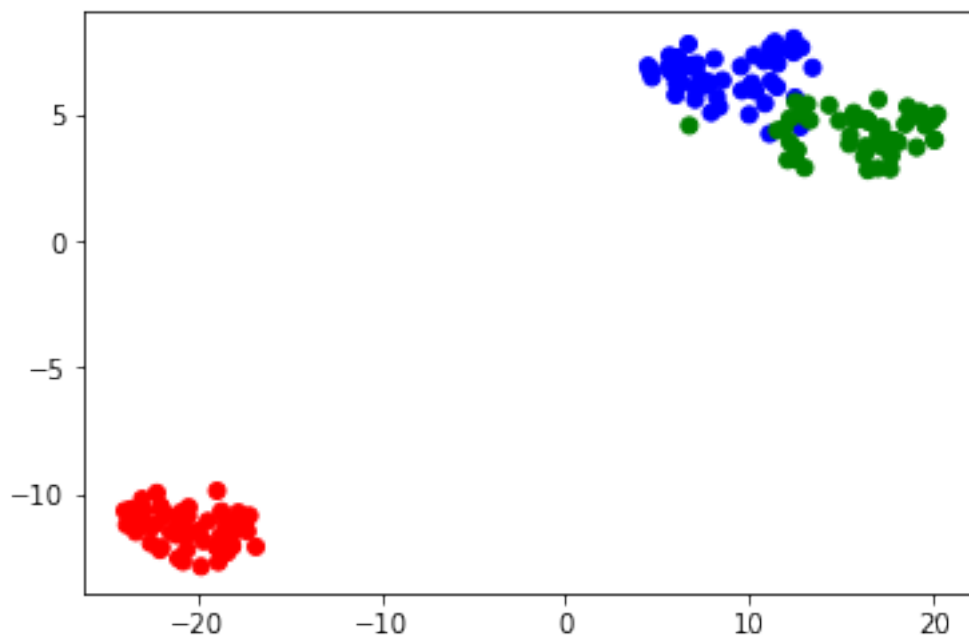
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)
```

```

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

```



```

[86]: 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[0:10000]

#y = price_standardized
#y = text_bow
y = (project_data['project_is_approved'])[0:10000]

tsne = TSNE(n_components=2, perplexity=20, learning_rate=1000, n_iter=1000)

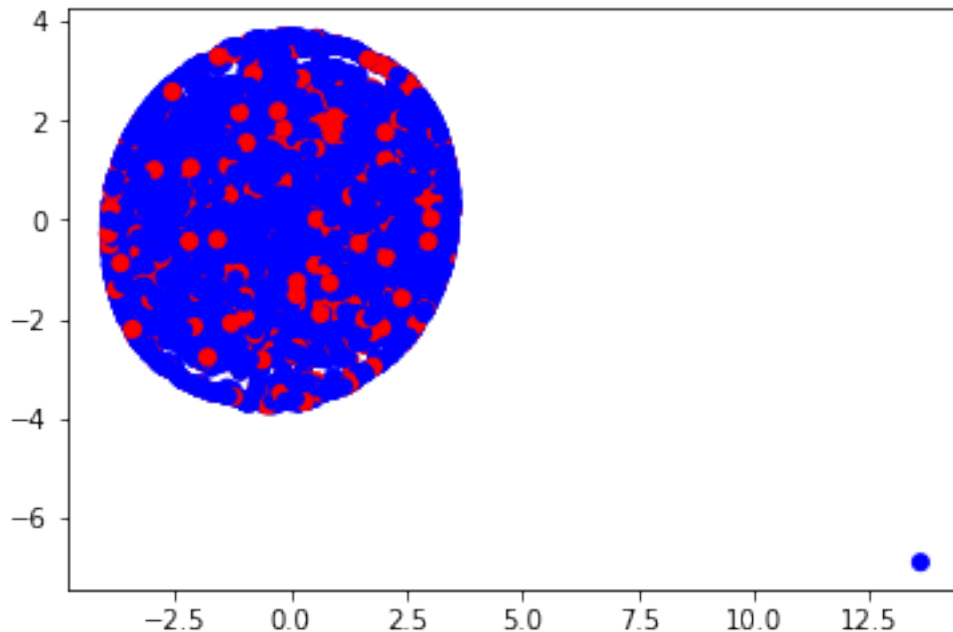
```

```

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

[87]: # please write all of the code with proper documentation and proper titles for
→each subsection

when you plot any graph make sure you use

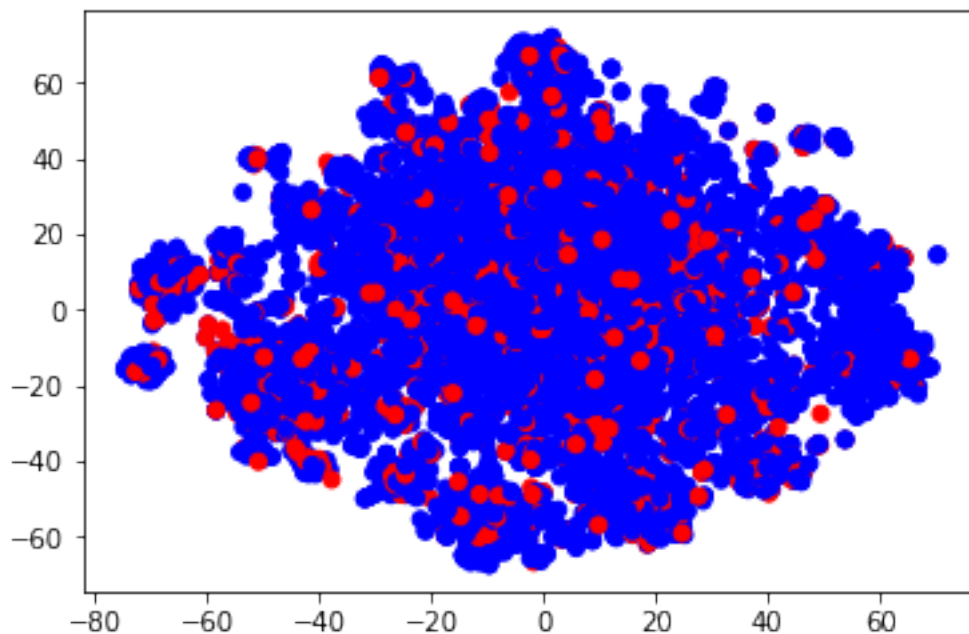
- # a. Title, that describes your plot, this will be very helpful to the
→reader
- # b. Legends if needed
- # c. X-axis label
- # d. Y-axis label

```
[88]: 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 = hstack((categories_one_hot, sub_categories_one_hot, text_bow,
    →price_standardized))
#y = price_standardized
#y = text_bow
x= text_bow[0:10000]
y = (project_data['project_is_approved'])[0:10000]
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200,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()
```



2.2 TSNE with TFIDF encoding of project_title feature

```
[89]: # 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
```

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

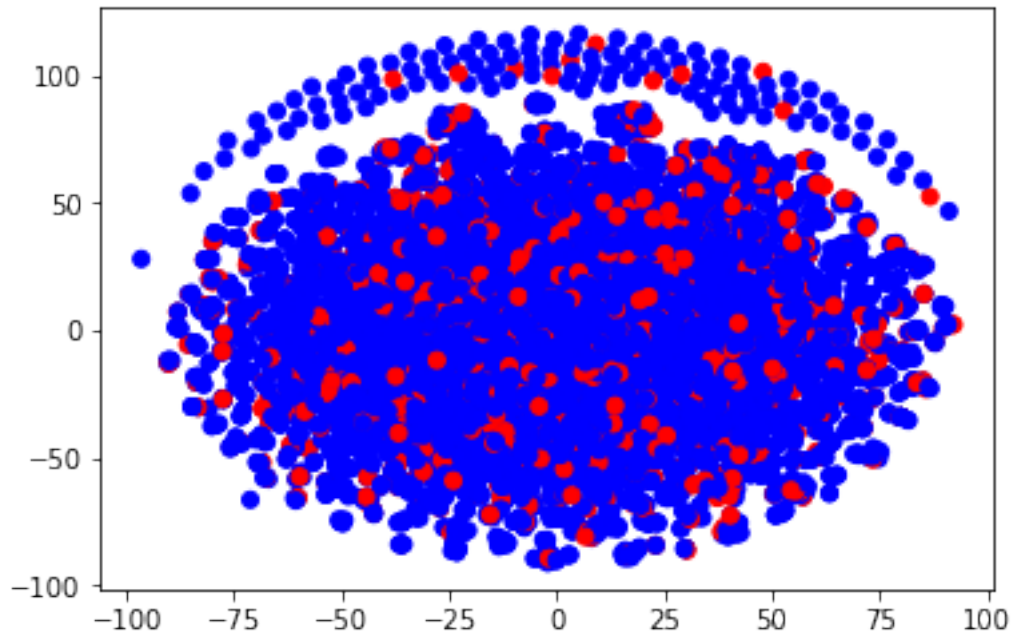
      # please check why to plot on the screen.
      #X = hstack((categories_one_hot, sub_categories_one_hot, text_bow,
      → price_standardized))
      #y = price_standardized
      x = text_tfidf[0:10000]

      y = project_data['project_is_approved'][0:10000]

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

      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

```
[91]: # 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
```

```
[92]: 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 = hstack((categories_one_hot, sub_categories_one_hot, text_bow,
      ↳ price_standardized))
      #y = price_standardized
      x = avg_w2v_vectors[0:10000]
      y = project_data['project_is_approved'][0:10000]

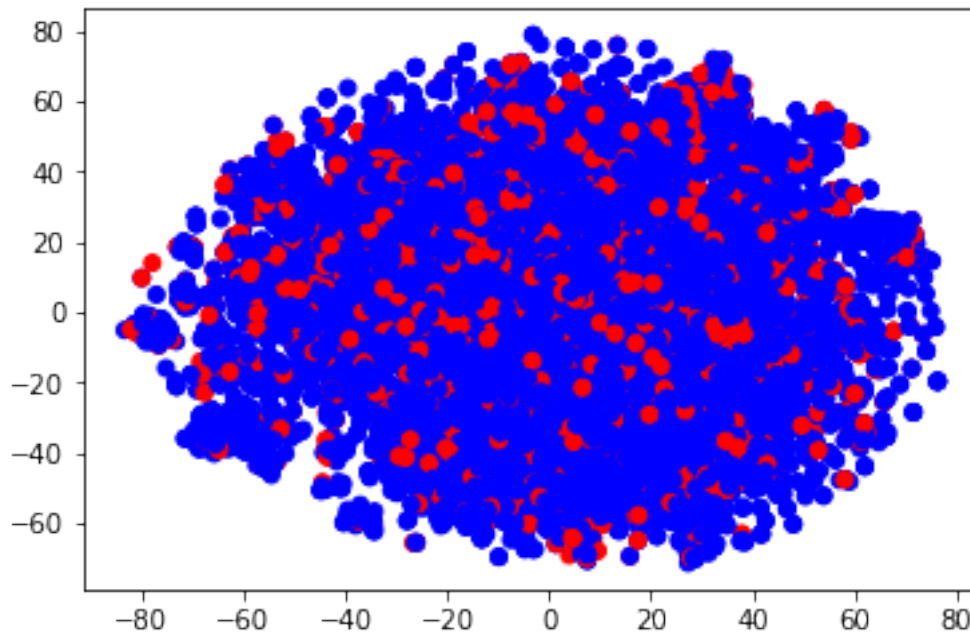
      tsne = TSNE(n_components=2, perplexity=30, learning_rate=200)
```

```

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.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

[93]: # 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

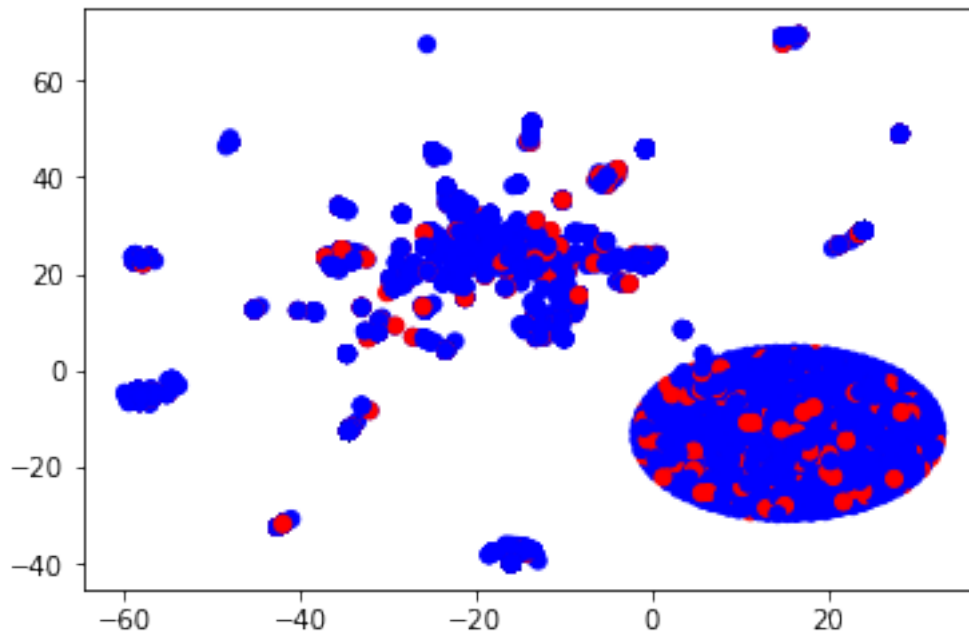
```
[95]: import numpy as np
from sklearn.manifold import TSNE
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

# please check why do plot on the screen.
#X = hstack((categories_one_hot, sub_categories_one_hot, text_bow,
    →price_standardized))
#y = price_standardized
x = tfidf_w2v_vectors[0:10000]
y = project_data['project_is_approved'][0:10000]

tsne = TSNE(n_components=2, perplexity=30, learning_rate=500)

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()
```



2.5 Summary

#tSNE As you can see above, various tSNE plots are drawn. tSNE is way better than PCA, if to segregate dense 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 as 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://distill.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.

[]: