DonorsChoose_EDA_TSNE

October 7, 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.

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

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

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

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

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

        from tqdm import tqdm
        import os

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

1.2 1.1 Reading Data

```
In [2]: project_data = pd.read_csv('../resources/train_data.csv')
        resource_data = pd.read_csv('../resources/resources.csv')
        # resource_data.head()
```

```
In [3]: print("Number of data points in project_data", project_data.shape)
        print('-'*50)
        print("The attributes of data :", project_data.columns.values)
        project_data.head(4)
```

```
Number of data points in project_data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

```
Out[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
```

```
In [4]: print("Number of data points in resource_data", resource_data.shape)
        print(resource_data.columns.values)
        resource_data.head(2)
```

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

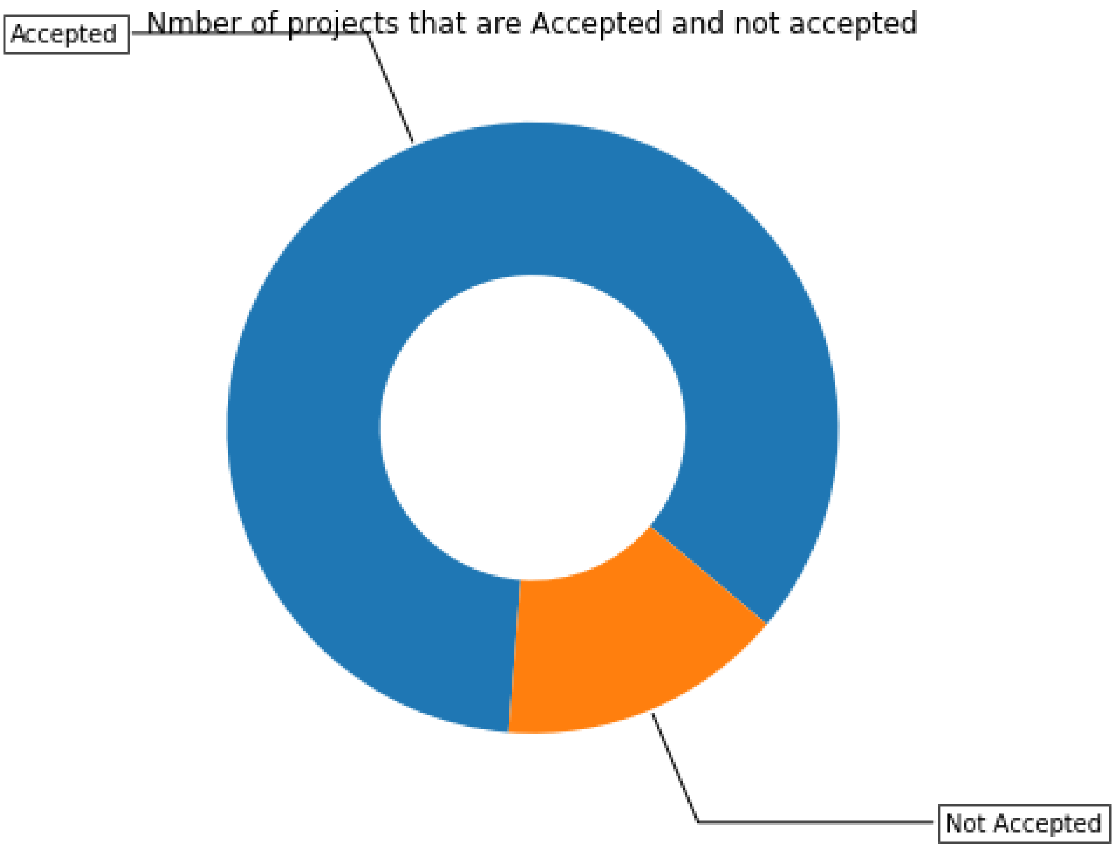
```
Out[4]:         id                                        description  quantity  \
        0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
```

```
1  p069063       Bouncy Bands for Desks (Blue support pipes)        3

    price
0  149.00
1   14.95
```

2 1.2 Data Analysis

In [5]:
```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-g

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

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

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

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

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

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

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

plt.show()
```

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



Observations:

1. The pie chart crearly indicates that the number of projects that are approved is very large as compared to projects that are not approve.
2. But the pie chart doesn't tell us the exact percentage of projects that are approved or not.

2.0.1 1.2.1 Univariate Analysis: School State

```
In [6]: # Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039

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

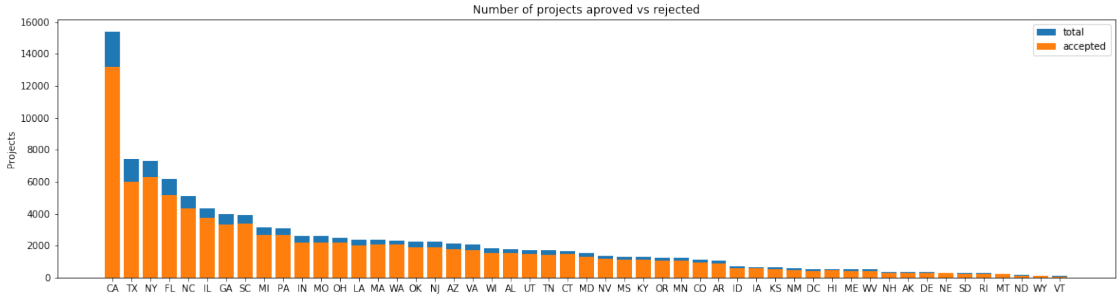
```
In [7]: # https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.
        temp.sort_values(by=['num_proposals'], inplace=True)
        print("States with lowest % approvals")
        print(temp.head(5))
        print('='*50)
        print("States with highest % approvals")
        print(temp.tail(5))
```

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.800000
7          DC       0.802326
43         TX       0.813142
26         MT       0.816327
18         LA       0.831245
==================================================
States with highest % approvals
   state_code  num_proposals
30         NH       0.873563
35         OH       0.875152
47         WA       0.876178
28         ND       0.888112
8          DE       0.897959
```

```
In [8]: #stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/b
        def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
            ind = np.arange(data.shape[0])

            plt.figure(figsize=(20,5))
            p1 = plt.bar(ind, data[col3].values)
            p2 = plt.bar(ind, data[col2].values)

            plt.ylabel('Projects')
            plt.title('Number of projects aproved vs rejected')
            plt.xticks(ind, list(data[xtick].values))
            plt.legend((p1[0], p2[0]), ('total', 'accepted'))
            plt.show()
```

```
In [9]: def univariate_barplots(data, col1, col2='project_is_approved', top=False):
            # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/
            temp = pd.DataFrame(data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).reset_i

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

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

            if top:
                temp = temp[0:top]

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

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



```
  school_state  project_is_approved  total       Avg
4           CA                13205  15388  0.858136
```

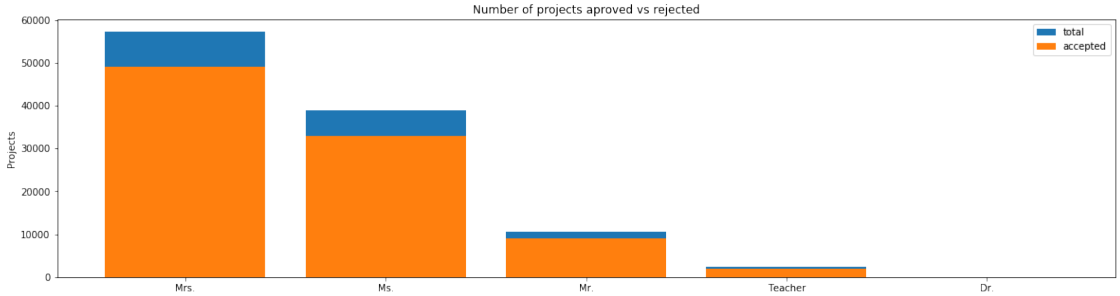
```
43           TX                  6014   7396  0.813142
34           NY                  6291   7318  0.859661
9            FL                  5144   6185  0.831690
27           NC                  4353   5091  0.855038
==================================================
   school_state  project_is_approved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

Observations:

1. Every state has greater than 80% success rate in approval.
2. State with code CA has highest number of approvals.
3. State with code VT has lowest number of approvals this may be because VT has very less project submissions as compared to CA.

2.0.2 1.2.2 Univariate Analysis: teacher_prefix

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



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

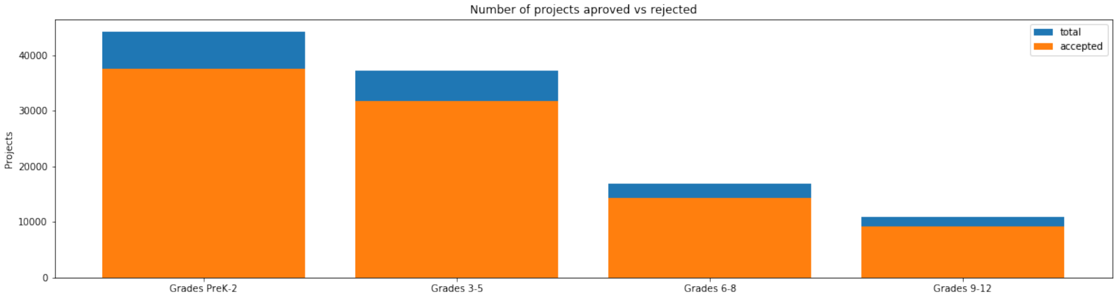
```
0            Dr.                    9     13  0.692308
```

Observations:

1. Teachers with prefix Mrs. and Ms. has highest number of approved projects.
2. Teachers with prefix Dr. has lowest number of approved projects.

2.0.3 1.2.3 Univariate Analysis: project_grade_category

```
In [12]: univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', to
```



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

Observations:

1. Most of the projects submitted belongs to project_grade_category: Grades PreK-2
2. Very less number of projects are submitted for project_grade_category: Grades 9-12
3. Projects with project_grade_category: Grades 3-5 has higher chance of approval as compared to others.

2.0.4 1.2.4 Univariate Analysis: project_subject_categories

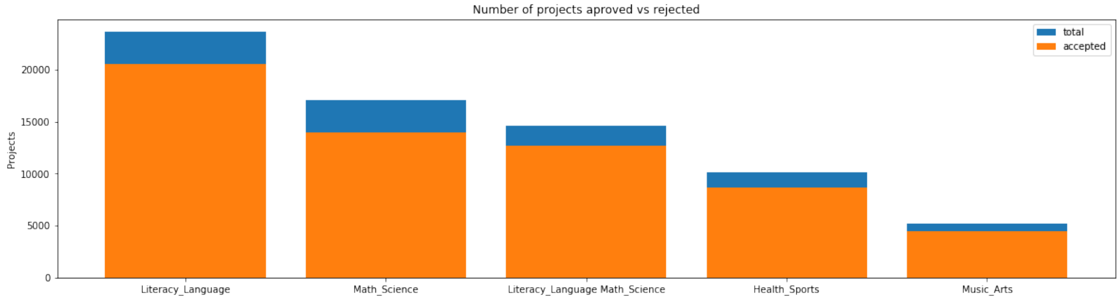
```
In [13]: catogories = list(project_data['project_subject_categories'].values)
         # remove special characters from list of strings python: https://stackoverflow.com/a/
         # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
         # https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-st
         # https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-py
         cat_list = []
         for i in catogories:
             temp = ""
             # consider we have text like this "Math & Science, Warmth, Care & Hunger"
             for j in i.split(','): # it will split it in three parts ["Math & Science", "Warm
                 if 'The' in j.split(): # this will split each of the catogory based on space
                     j=j.replace('The','') # if we have the words "The" we are going to replac
                 j = j.replace(' ','') # we are placing all the ' '(space) with ''(empty) ex:"
                 temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing s
                 temp = temp.replace('&','_') # we are replacing the & value into _
             cat_list.append(temp.strip())
```

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

```
Out[14]:    Unnamed: 0       id                        teacher_id teacher_prefix  \
         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
```

```
In [15]: univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=5)
```



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

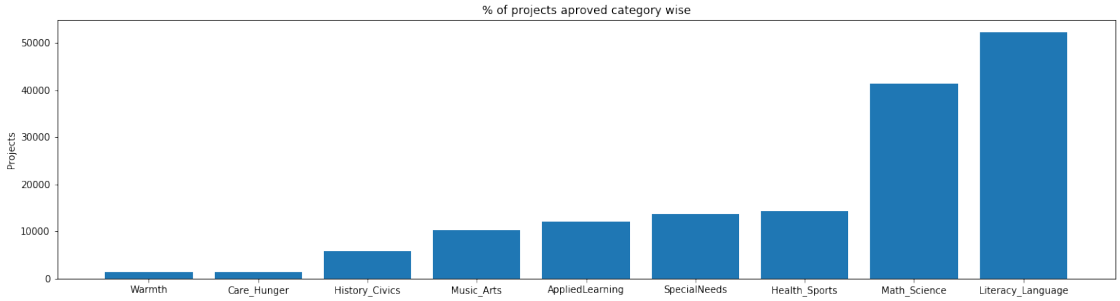
```
                 clean_categories  project_is_approved  total       Avg
24              Literacy_Language                20520  23655  0.867470
32                   Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12725  14636  0.869432
8      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

Observations:

1. Projects with multiple project_categories has highest approval rate.(e.g Literacy_Language Math_Science)

```
In [16]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/40840
         from collections import Counter
         my_counter = Counter()
         for word in project_data['clean_categories'].values:
             my_counter.update(word.split())
```

```
In [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()
```



Observations:

1. projects with project_category Literacy_Language has highest approval rate as compared to others

```
In [18]: for i, j in sorted_cat_dict.items():
             print("{:20} :{:10}".format(i,j))
```

```
Warmth               :      1388
Care_Hunger          :      1388
History_Civics       :      5914
Music_Arts           :     10293
AppliedLearning      :     12135
SpecialNeeds         :     13642
Health_Sports        :     14223
Math_Science         :     41421
Literacy_Language    :     52239
```

2.0.5 1.2.5 Univariate Analysis: project_subject_subcategories

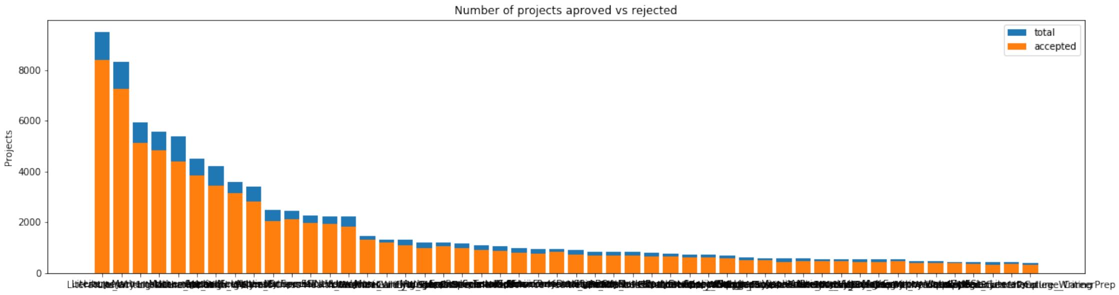
```
In [19]: sub_catogories = list(project_data['project_subject_subcategories'].values)
         # remove special characters from list of strings python: https://stackoverflow.com/a/
         # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
         # https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-st
         # https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-py

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

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

```
Out[20]:    Unnamed: 0       id                        teacher_id teacher_prefix  \
         0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
         1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.

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

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

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

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

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

            teacher_number_of_previously_posted_projects  project_is_approved  \
         0                                             0                    0
         1                                             7                    1
```

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

```
In [21]: univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top=5
```



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

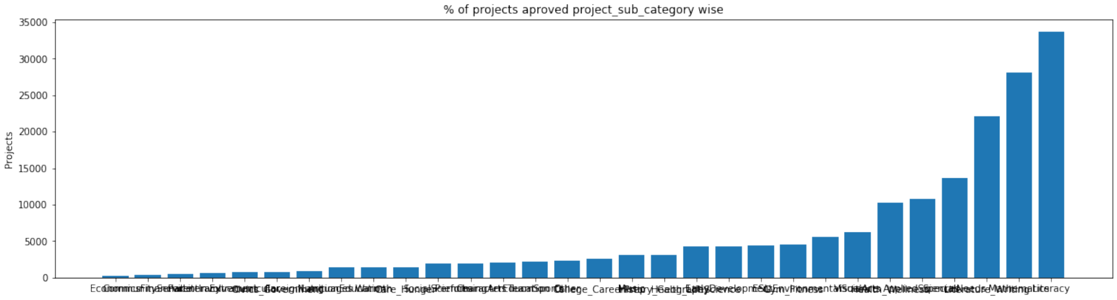
Observations:

1. Most of the posted projects belongs to Literacy and Literacy Mathematics project_sub_category

```
In [22]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/40840
         from collections import Counter
         my_counter = Counter()
         for word in project_data['clean_subcategories'].values:
             my_counter.update(word.split())
```

```
In [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 project_sub_category wise')
plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
plt.show()
```



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

```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
NutritionEducation   :      1355
Warmth               :      1388
Care_Hunger          :      1388
SocialSciences       :      1920
PerformingArts       :      1961
CharacterEducation   :      2065
TeamSports           :      2192
Other                :      2372
College_CareerPrep   :      2568
Music                :      3145
History_Geography    :      3171
Health_LifeScience   :      4235
EarlyDevelopment     :      4254
ESL                  :      4367
Gym_Fitness          :      4509
```

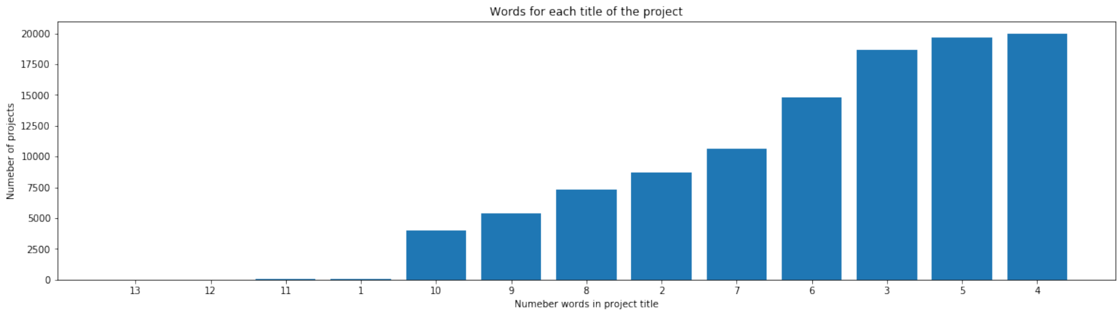
```
EnvironmentalScience :       5591
VisualArts           :       6278
Health_Wellness      :      10234
AppliedSciences      :      10816
SpecialNeeds         :      13642
Literature_Writing   :      22179
Mathematics          :      28074
Literacy             :      33700
```

Observations:

1. Projects belonging to sub_category Literacy and Mathematics has higher approval rates as compared to others.

2.0.6 1.2.6 Univariate Analysis: Text features (Title)

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


         ind = np.arange(len(word_dict))
         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()
```



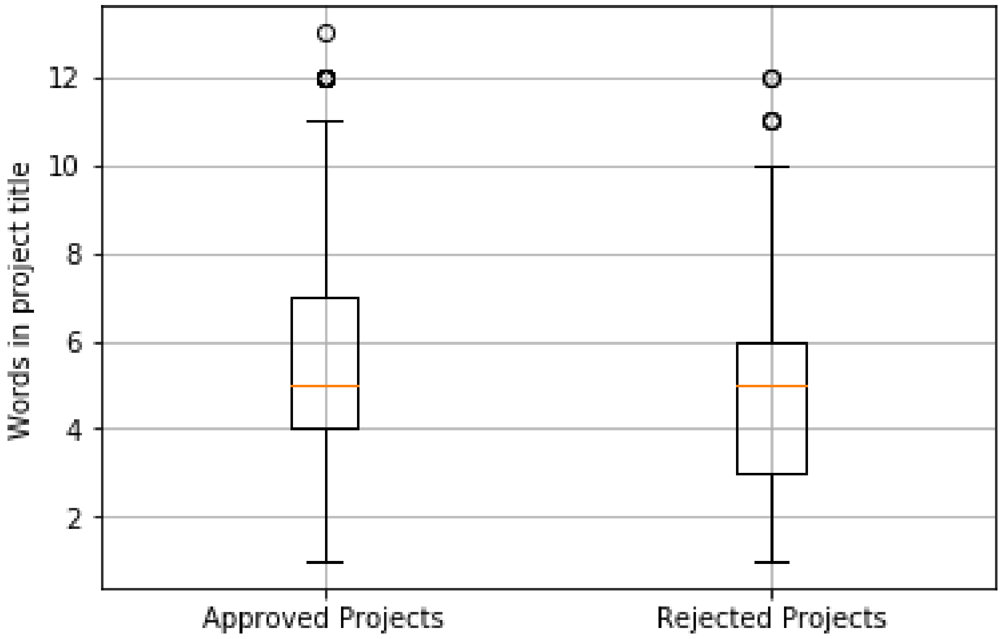
Observations:

1. Projects having shorter titles are having better approval chances.

2. Optimal number of words for a project title is 4

```
In [26]: approved_title_word_count = project_data[project_data['project_is_approved']==1]['pro
         approved_title_word_count = approved_title_word_count.values

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

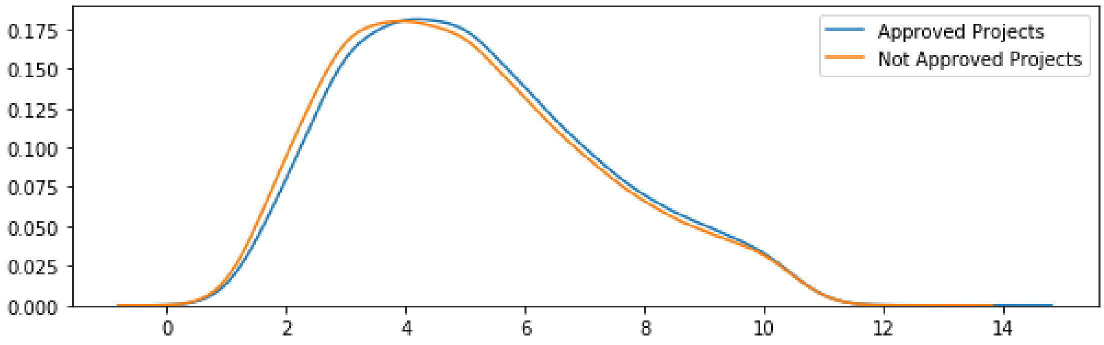
```
In [27]: # https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
         plt.boxplot([approved_title_word_count, rejected_title_word_count])
         plt.xticks([1,2],('Approved Projects','Rejected Projects'))
         plt.ylabel('Words in project title')
         plt.grid()
         plt.show()
```



Observations:

1. About 75% of the rejected projects has <6 words in their title.
2. About 50% of the approved projects has <5 words in their title.

```
In [28]: plt.figure(figsize=(10,3))
         sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)
         sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6)
         plt.legend()
         plt.show()
```



Observations:

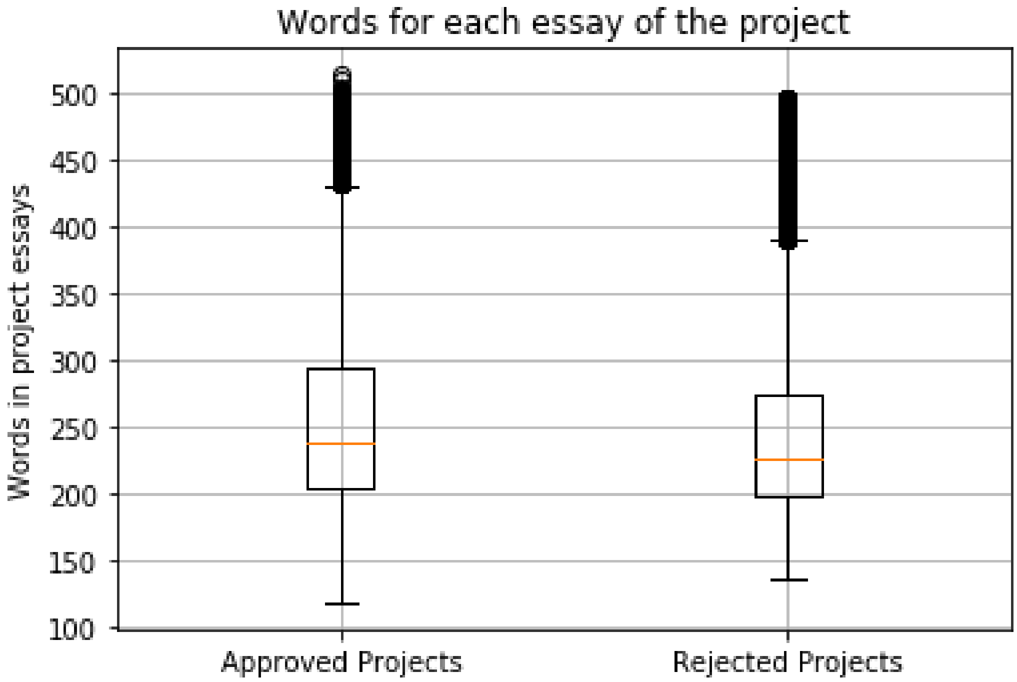
1. Out of the total submitted projects, projects with title length <4 & >11 has higher rejection rate.
2. Projects with 4 to 11 words in title has slightly higher chances of approval than others.

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

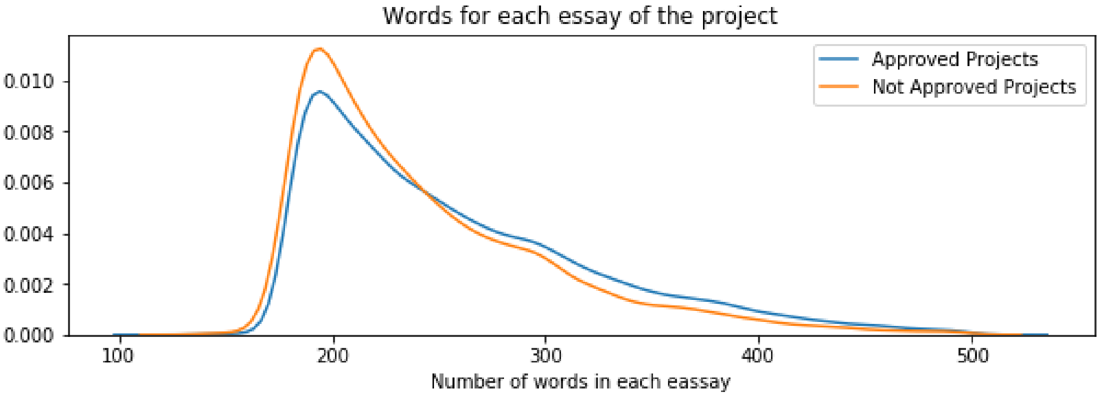
```
In [29]: # merge two column text dataframe:
         project_data["essay"] = project_data["project_essay_1"].map(str) +\
                                 project_data["project_essay_2"].map(str) + \
                                 project_data["project_essay_3"].map(str) + \
                   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

```
In [30]: approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].s
         approved_word_count = approved_word_count.values

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

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



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



Observations:

- Projects with larger description has better approval chances.

2.0.8 1.2.8 Univariate Analysis: Cost per project

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

```
Out[33]:         id                                        description  quantity  \
         0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
         1  p069063        Bouncy Bands for Desks (Blue support pipes)         3

             price
         0  149.00
         1   14.95
```

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

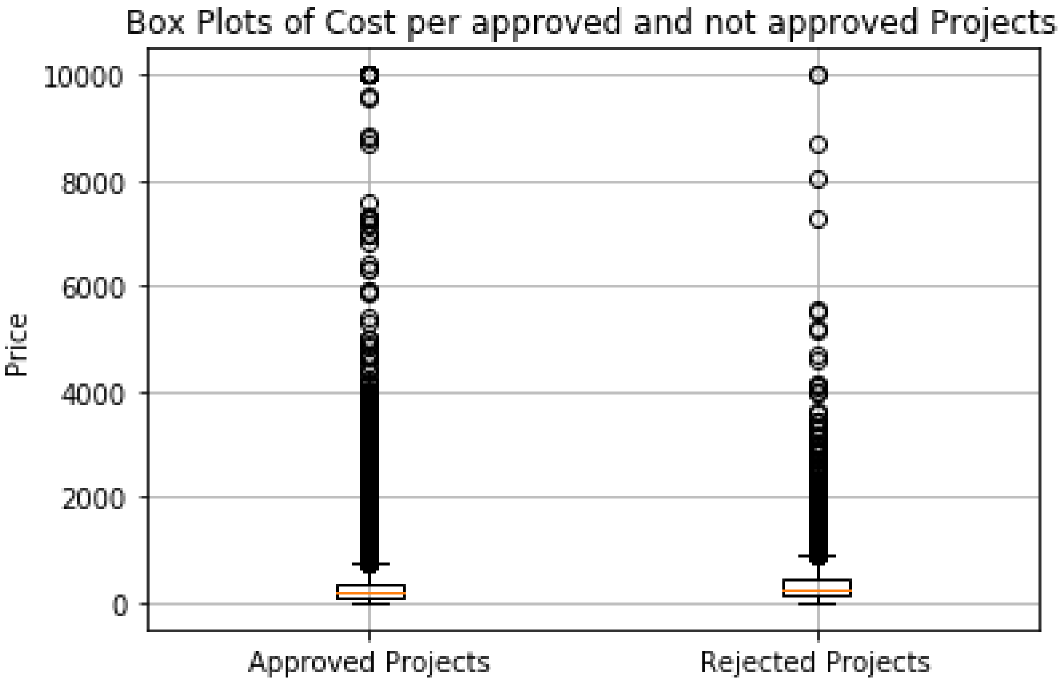
```
Out[34]:         id   price  quantity
         0  p000001  459.56         7
         1  p000002  515.89        21
```

```
In [35]: # join two dataframes in python:
         project_data = pd.merge(project_data, price_data, on='id', how='left')
         project_data.head(5)
         print(project_data.shape)
```

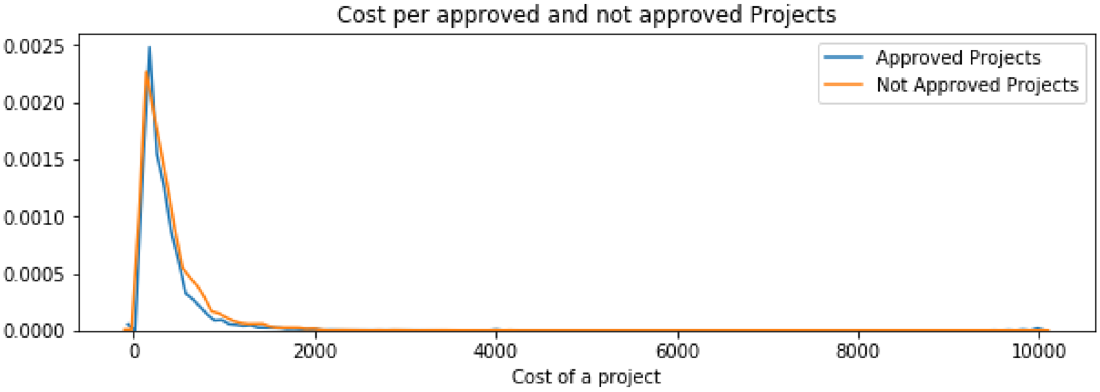
```
(109248, 20)
```

```
In [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
```

```
In [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()
```



In [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()
```



Observations:

1. Likelihood of project being approved and rejected is not dependent on project cost.
2. Most of the rejected projects has cost <4k dollars

```
In [39]: # http://zetcode.com/python/prettytable/
         from prettytable import PrettyTable

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

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

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

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

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

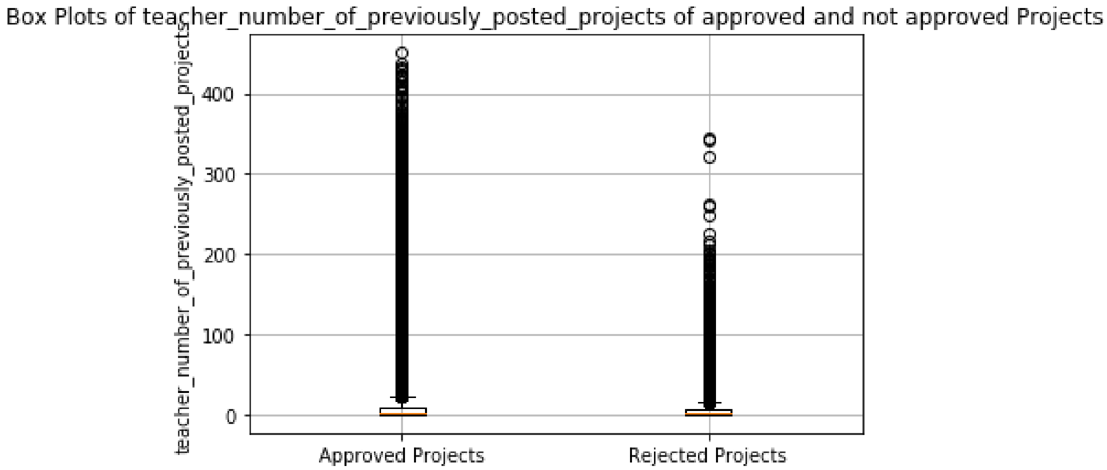
```
In [40]: # we get the teacher_number_of_previously_posted_projects using train_data.csv file
         project_data.head(2)
```

```
Out[40]:    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

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

In [41]:
```
approved = project_data[project_data['project_is_approved']==1]['teacher_number_of_pro
rejected = project_data[project_data['project_is_approved']==0]['teacher_number_of_pro
```

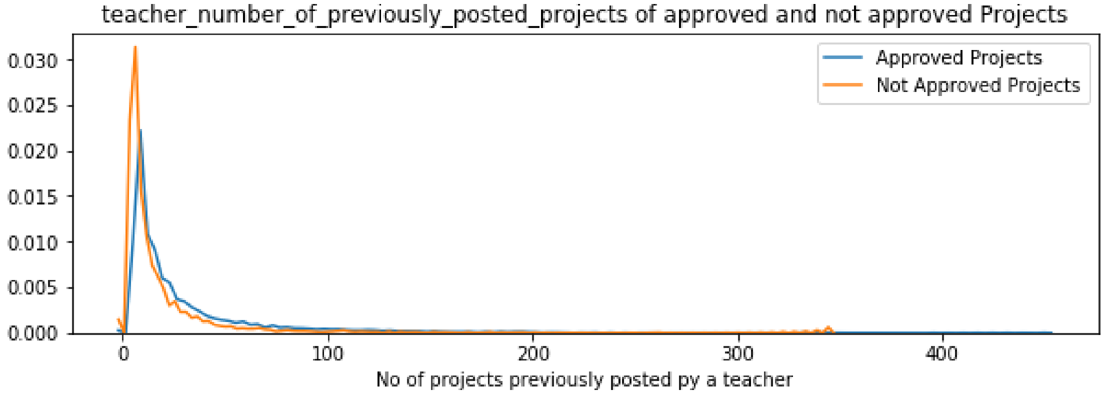
In [42]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved, rejected])
plt.title('Box Plots of teacher_number_of_previously_posted_projects of approved and r
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('teacher_number_of_previously_posted_projects')
plt.grid()
plt.show()
```



Observations:

1. Projects whose teachers have number_of_previously_posted_projects >200 has better chances of approval.

```
In [43]: plt.figure(figsize=(10,3))
         sns.distplot(approved, hist=False, label="Approved Projects")
         sns.distplot(rejected, hist=False, label="Not Approved Projects")
         plt.title('teacher_number_of_previously_posted_projects of approved and not approved P
         plt.xlabel('No of projects previously posted py a teacher')
         plt.legend()
         plt.show()
```



Observations:

1. Projects whose teachers have number_of_previously_posted_projects >350 has nearly 100% chances of approval.
2. Projects whose teachers have number_of_previously_posted_projects >20 and <100 has slightly higher approval chances than rejection.
3. Projects whose teachers have number_of_previously_posted_projects <5 has greater rejection rates.

```
In [44]: # http://zetcode.com/python/prettytable/
         from prettytable import PrettyTable

         #If you get a ModuleNotFoundError error , install prettytable using: pip3 install pre
         #or
         #https://anaconda.org/synthicity/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,i), 3), np.round(np.percentile(rejec
         print(x)
```

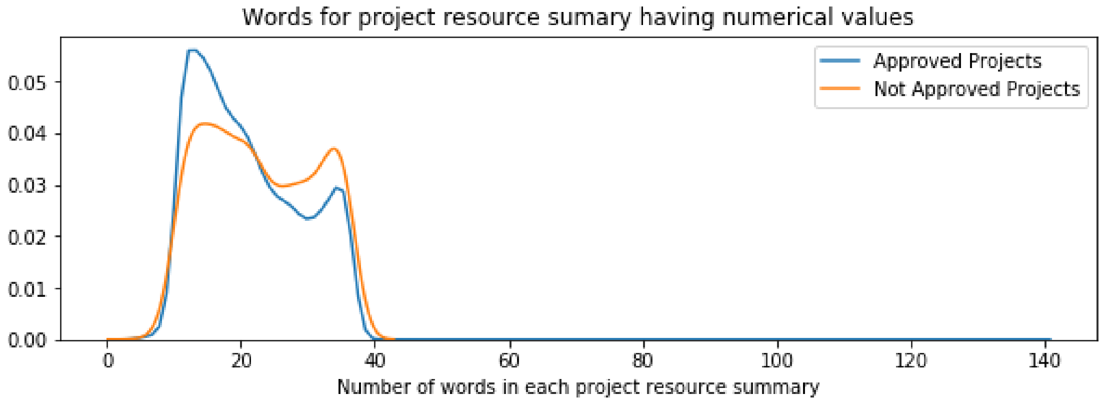
```
+------------+-------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+-------------------+-----------------------+
|     0      |        0.0        |          0.0          |
|     5      |        0.0        |          0.0          |
|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

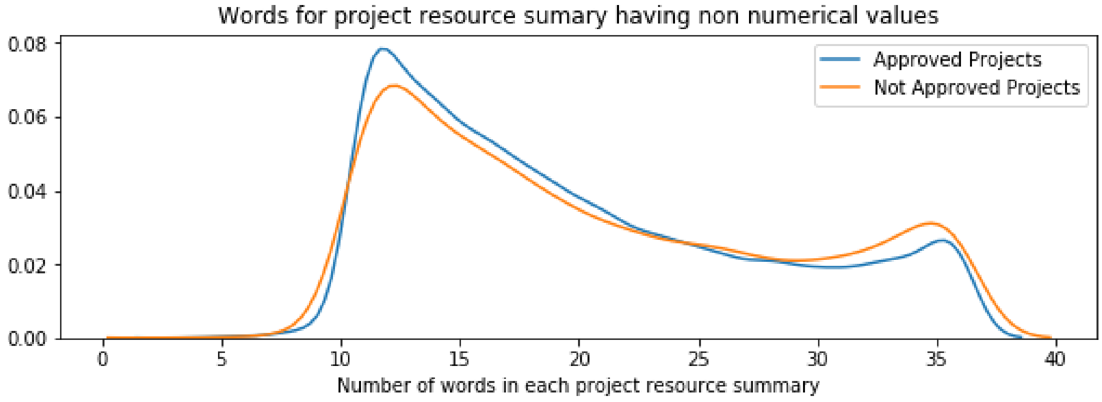
1.2.10 Univariate Analysis: project_resource_summary

In [45]:
```
# https://stackoverflow.com/questions/44140489/get-non-numerical-rows-in-a-column-pan
#https://stackoverflow.com/questions/19859282/check-if-a-string-contains-a-number
import re

numbers_string_df = project_data[project_data['project_resource_summary'].apply(lambda
string_df = project_data[project_data['project_resource_summary'].apply(lambda x: not

print(numbers_string_df.shape)
print(string_df.shape)


approved_numbers_string = numbers_string_df[numbers_string_df['project_is_approved']==
approved_string = string_df[string_df['project_is_approved']==1]['project_resource_su

approved_numbers_string = approved_numbers_string.values
approved_string = approved_string.values


rejected_numbers_string = numbers_string_df[numbers_string_df['project_is_approved']==
rejected_string = string_df[string_df['project_is_approved']==0]['project_resource_su

rejected_numbers_string = rejected_numbers_string.values
rejected_string = rejected_string.values
```

```
(15756, 20)
(93492, 20)
```

In [46]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_numbers_string, hist=False, label="Approved Projects")
sns.distplot(rejected_numbers_string, hist=False, label="Not Approved Projects")
plt.title('Words for project resource sumary having numerical values')
plt.xlabel('Number of words in each project resource summary')
plt.legend()
plt.show()

plt.figure(figsize=(10,3))
sns.distplot(approved_string, hist=False, label="Approved Projects")
sns.distplot(rejected_string, hist=False, label="Not Approved Projects")
plt.title('Words for project resource sumary having non numerical values')
plt.xlabel('Number of words in each project resource summary')
plt.legend()
plt.show()
```





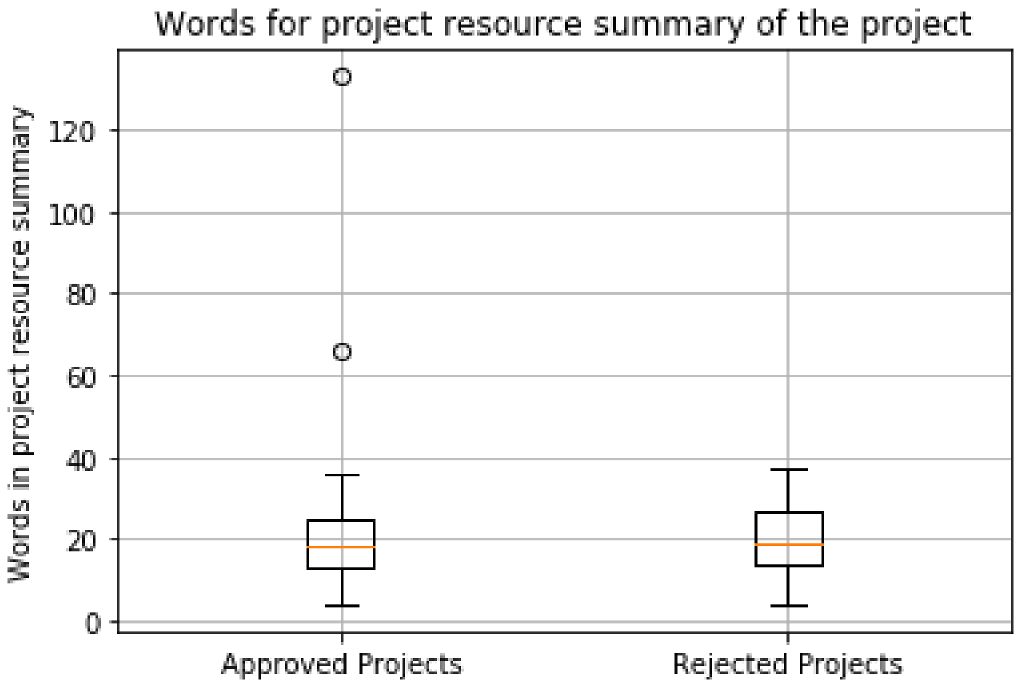
Observations:

1. By looking at pdf of both string and string with numerical values of project_resource_summary, we find that pdf of both the plots is similar.
2. The only difference is in the spread which is because of the number of points. Since the number of points in project resource summary having only string is more so, spread is more.

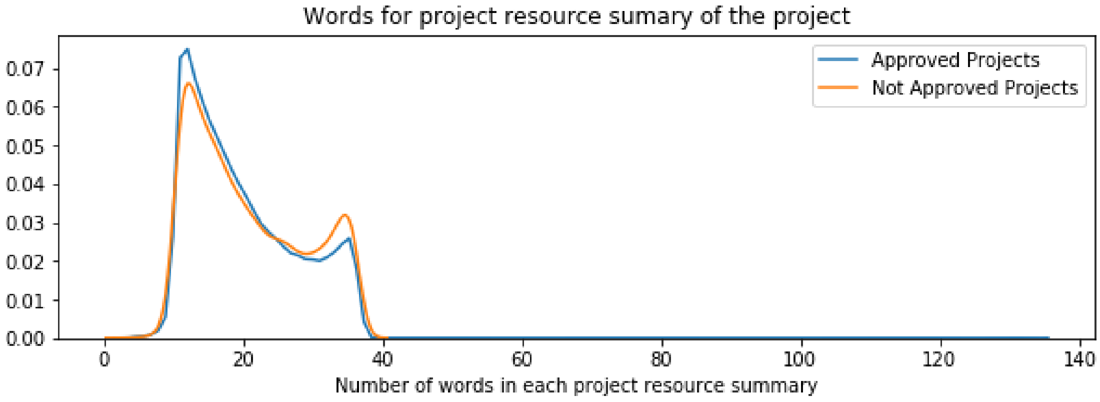
```
In [47]: project_data['project_resource_summary'] = project_data['project_resource_summary'].ap
```

```
In [48]: approved_project_resource_summary = project_data[project_data['project_is_approved']==
         approved_project_resource_summary = approved_project_resource_summary.values

         rejected_project_resource_summary = project_data[project_data['project_is_approved']==
         rejected_project_resource_summary = rejected_project_resource_summary.values
```

```
In [49]: # https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
         plt.boxplot([approved_project_resource_summary, rejected_project_resource_summary])
         plt.title('Words for project resource summary of the project')
         plt.xticks([1,2],('Approved Projects','Rejected Projects'))
         plt.ylabel('Words in project resource summary')
         plt.grid()
         plt.show()
```



```
In [50]: plt.figure(figsize=(10,3))
         sns.distplot(approved_project_resource_summary, hist=False, label="Approved Projects"
         sns.distplot(rejected_project_resource_summary, hist=False, label="Not Approved Proje
         plt.title('Words for project resource sumary of the project')
         plt.xlabel('Number of words in each project resource summary')
         plt.legend()
         plt.show()
```



Observations:

1. Most of the projects has <40 words resource summary
2. Only few projects has resource summary words >40 and there approval chance is very high.

2.1 1.3 Text preprocessing

2.1.1 1.3.1 Essay Text

- I will perform t-SNE on 4000 rows only due to memory constraint

```
In [51]: final_4000 = project_data[:4000]

         #https://stackoverflow.com/questions/12850345/how-to-combine-two-data-frames-in-pytho

         projects_approved = project_data[project_data["project_is_approved"] == 1].sample(n =
         projects_rejected = project_data[project_data["project_is_approved"] == 0].sample(n =
         final_4000 = pd.concat([projects_approved, projects_rejected])
         final_4000.head(2)
```

```
Out[51]:        Unnamed: 0       id                        teacher_id teacher_prefix  \
         56268      104617  p098697  8131749e34b7ef3fa0890b5d840deb2a            Ms.
         97545       26847  p098801  d0e1e45a5d6d7be3143d80c6604b47fd           Mrs.

               school_state project_submitted_datetime project_grade_category  \
         56268           NC        2016-06-20 13:27:03             Grades 3-5
         97545           MD        2017-04-07 13:16:36             Grades 3-5

                                                     project_title  \
         56268  Filling Our Science Equipment Gaps for Outstan...
         97545                                Comfy Cozy Reading

                                                   project_essay_1  \
         56268  \"We love science in your class!\" CJ exclaime...
         97545  I teach fifty fourth grade students Reading an...

                                                   project_essay_2 project_essay_3  \
         56268  My students love to experiment as we learn sci...             NaN
         97545  My students are in need of places to read and ...             NaN

               project_essay_4                            project_resource_summary  \
         56268             NaN  My students need hands-on materials to improve...
         97545             NaN  My students need seating that helps them learn...

                teacher_number_of_previously_posted_projects  project_is_approved  \
         56268                                             1                    1
```

```
97545                                              0                    1

        clean_categories                       clean_subcategories  \
56268       Math_Science  EnvironmentalScience Health_LifeScience
97545  Literacy_Language                                  Literacy

                                                 essay   price  quantity
56268  \"We love science in your class!\" CJ exclaime...  149.86         3
97545  I teach fifty fourth grade students Reading an...  489.92        12
```

```
In [52]: # printing some random essays.
         print(final_4000['essay'].values[0])
         print("="*50)
         print(final_4000['essay'].values[150])
         print("="*50)
         print(final_4000['essay'].values[1000])
         print("="*50)
         print(final_4000['essay'].values[2000])
         print("="*50)
         print(final_4000['essay'].values[3999])
         print("="*50)
```

```
\"We love science in your class!\" CJ exclaimed as he came in the door this past year as a fou
==================================================
The best way we can improve physical fitness is by increasing movement among students at school
==================================================
Roughly 90% of the students qualify for free/reduced lunch. Nearly 50% have an Individualized I
==================================================
I work at a Title One School where 95% of the students receive free or reduced lunch. Majority
==================================================
My students are economically disadvantage in the Hispanic demographics. It is really hard for t
==================================================
```

```
In [53]: # https://stackoverflow.com/a/47091490/4084039
         import re

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

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

```
            phrase = re.sub(r"\'t", " not", phrase)
            phrase = re.sub(r"\'ve", " have", phrase)
            phrase = re.sub(r"\'m", " am", phrase)
            return phrase
```

```
In [54]: sent = decontracted(final_4000['essay'].values[2000])
         print(sent)
         print("="*50)
```

```
I work at a Title One School where 95% of the students receive free or reduced lunch. Majority
==================================================
```

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

```
I work at a Title One School where 95% of the students receive free or reduced lunch. Majority
```

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

```
I work at a Title One School where 95 of the students receive free or reduced lunch Majority o
```

```
In [57]: # https://gist.github.com/sebleier/554280
         # we are removing the words from the stop words list: 'no', 'nor', 'not'
         stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'r
                     "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him'
                     'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
                     'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "t
                     'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'h
                     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
                     'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
                     'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'o
                     'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
                     'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
                     's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'n
                     've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't"
                     "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mig
                     "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
                     'won', "won't", 'wouldn', "wouldn't"]
```

```
In [58]: # Combining all the above statemennts
         from tqdm import tqdm
```

```
        preprocessed_essays = []
        # tqdm is for printing the status bar
        for sentance in tqdm(final_4000['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%|| 4000/4000 [00:02<00:00, 1756.23it/s]
```

```
In [59]: # after preprocesing
         preprocessed_essays[2000]
```

```
Out[59]: 'i work title one school 95 students receive free reduced lunch majority students come
```

1.3.2 Project title Text

```
In [60]: # printing some random essays.
         print(final_4000['project_title'].values[0])
         print("="*50)
         print(final_4000['project_title'].values[100])
         print("="*50)
         print(final_4000['project_title'].values[1000])
         print("="*50)
         print(final_4000['project_title'].values[2000])
         print("="*50)
         print(final_4000['project_title'].values[3999])
         print("="*50)
```

```
Filling Our Science Equipment Gaps for Outstanding Learning
==================================================
Kinder Farm to Kinder Table
==================================================
Multi-Cultural Percussion
==================================================
Science Lab Supplies
==================================================
Active bodies, building healthier clever minds!!
==================================================
```

```
In [61]: # similarly you can preprocess the titles also
         preprocessed_titles = []
         # tqdm is for printing the status bar
         for sentance in tqdm(final_4000['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)
             sent = ' '.join(e for e in sent.split() if e not in stopwords)
             preprocessed_titles.append(sent.lower().strip())
```

```
100%|| 4000/4000 [00:00<00:00, 36140.50it/s]
```

```
In [62]: print(preprocessed_titles[100])
         print("-"*50)
         print(preprocessed_titles[2000])
```

```
kinder farm kinder table
--------------------------------------------------
science lab supplies
```

2.2 1. 4 Preparing data for models

```
In [63]: final_4000.columns
```

```
Out[63]: 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'],
               dtype='object')
```

we are going to consider

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

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

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

2.2.1 1.4.1 Vectorizing Categorical data

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

```
In [64]: # 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
         vectorizer.fit(final_4000['clean_categories'].values)
         print(vectorizer.get_feature_names())


         categories_one_hot = vectorizer.transform(final_4000['clean_categories'].values)
         print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'I
Shape of matrix after one hot encodig  (4000, 9)
```

```
In [65]: # we use count vectorizer to convert the values into one hot encoded features
         vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fa
         vectorizer.fit(final_4000['clean_subcategories'].values)
         print(vectorizer.get_feature_names())


         sub_categories_one_hot = vectorizer.transform(final_4000['clean_subcategories'].values
         print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
Shape of matrix after one hot encodig  (4000, 30)
```

```
In [66]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/40840
         my_counter = Counter()
         for word in final_4000['school_state'].values:
             my_counter.update(word.split())

         state_dict = dict(my_counter)
         sorted_state_dict = dict(sorted(state_dict.items(), key=lambda kv: kv[1]))

         # we use count vectorizer to convert the values into one hot encoded features
         vectorizer = CountVectorizer(vocabulary=list(sorted_state_dict.keys()), lowercase=Fal
         vectorizer.fit(final_4000['school_state'].values)
         print(vectorizer.get_feature_names())


         school_state_one_hot = vectorizer.transform(final_4000['school_state'].values)
         print("Shape of matrix after one hot encodig ",school_state_one_hot.shape)
```

```
['MT', 'ND', 'WY', 'VT', 'SD', 'NH', 'AK', 'NE', 'DE', 'RI', 'ME', 'NM', 'WV', 'DC', 'HI', 'IA
Shape of matrix after one hot encodig  (4000, 51)
```

```
In [67]: #https://stackoverflow.com/questions/48090658/sklearn-how-to-incorporate-missing-data
         my_counter = Counter()
         project_data['teacher_prefix'] = final_4000['teacher_prefix'].replace({np.nan:'None'}


         for word in final_4000['teacher_prefix'].values:
             my_counter.update(word.split())

         teacher_prefix_dict = dict(my_counter)
         sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv: k

         # we use count vectorizer to convert the values into one hot encoded features
         vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), lower
         vectorizer.fit(final_4000['teacher_prefix'].values)
         print(vectorizer.get_feature_names())


         teacher_prefix_one_hot = vectorizer.transform(final_4000['teacher_prefix'].values)
         print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
```

```
['Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encodig  (4000, 4)
```

```
In [68]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/40840
         my_counter = Counter()
         for word in final_4000['project_grade_category'].values:
             my_counter.update(word.split())

         print(my_counter)
         project_grade_dict = dict(my_counter)
         sorted_project_grade_dict = dict(sorted(project_grade_dict.items(), key=lambda kv: kv

         # we use count vectorizer to convert the values into one hot encoded features
         vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_dict.keys()), lowerc
         vectorizer.fit(final_4000['project_grade_category'].values)
         print(vectorizer.get_feature_names())


         project_grade_one_hot = vectorizer.transform(final_4000['project_grade_category'].valu
         print("Shape of matrix after one hot encodig ",project_grade_one_hot.shape)
```

```
Counter({'Grades': 4000, 'PreK-2': 1607, '3-5': 1355, '6-8': 613, '9-12': 425})
['9-12', '6-8', '3-5', 'PreK-2', 'Grades']
Shape of matrix after one hot encodig  (4000, 5)
```

2.2.2 1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

```
In [69]: # We are considering only the words which appeared in at least 10 documents(rows or p
         vectorizer = CountVectorizer(min_df=10)
         essays_bow = vectorizer.fit_transform(preprocessed_essays)
         print("Shape of matrix after vectorizing essays ",essays_bow.shape)
```

```
Shape of matrix after vectorizing essays  (4000, 3824)
```

1.4.2.2 Bag of Words on project_title

```
In [70]: # Similarly you can vectorize for title also
         vectorizer = CountVectorizer(min_df=10)
         title_bow = vectorizer.fit_transform(preprocessed_titles)
         print("Shape of matrix after vectorizing title ",title_bow.shape)
```

```
Shape of matrix after vectorizing title  (4000, 318)
```

1.4.2.3 TFIDF vectorizer

```
In [71]: from sklearn.feature_extraction.text import TfidfVectorizer
         vectorizer = TfidfVectorizer(min_df=10)
         essay_tfidf = vectorizer.fit_transform(preprocessed_essays)
         print("Shape of matrix after one hot encodig ",essay_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (4000, 3824)
```

1.4.2.4 TFIDF Vectorizer on project_title

```
In [72]: # Similarly you can vectorize for title also
         vectorizer = TfidfVectorizer(min_df=10)
         title_tfidf = vectorizer.fit_transform(preprocessed_titles)
         print("Shape of matrix after vectorizing title ",title_tfidf.shape)
```

```
Shape of matrix after vectorizing title  (4000, 318)
```

1.4.2.5 Using Pretrained Models: Avg W2V

```
In [74]: # 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('../resources/glove.42B.300d.txt')

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

         #Loading Glove Model
         #1917495it [02:42, 11830.67it/s]
         #Done. 1917495  words loaded!

         # ============================
         '''

Out[74]: '\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gl

In [75]: '''
         words = []
         for i in preprocessed_essays:
             words.extend(i.split(' '))

         for i in preprocessed_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

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

         '''

Out[75]: '\nwords = []\nfor i in preprocessed_essays:\n    words.extend(i.split(\' \'))\n\nfor

In [76]: # storing variables into pickle files python: http://www.jessicayung.com/how-to-use-p
         # make sure you have the glove_vectors file
         with open('glove_vectors', 'rb') as f:
             model = pickle.load(f)
             glove_words =  set(model.keys())

In [77]: # 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%|| 4000/4000 [00:01<00:00, 3409.31it/s]

4000
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

```
In [78]: avg_w2v_title_vector = [];
         for sentence in tqdm(preprocessed_titles): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             cnt_words =0; # num of words with a valid vector in the sentence/review
             for word in sentence.split(): # for each word in a review/sentence
                 if word in glove_words:
                     vector += model[word]
                     cnt_words += 1
             if cnt_words != 0:
                 vector /= cnt_words
             avg_w2v_title_vector.append(vector)

         print(len(avg_w2v_title_vector))
         print(len(avg_w2v_title_vector[0]))
```

```
100%|| 4000/4000 [00:00<00:00, 66518.18it/s]
```

```
4000
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

```
In [79]: tfidf_model = TfidfVectorizer()
         tfidf_model.fit(preprocessed_essays)
         # we are converting a dictionary with word as a key, and the idf as a value
         dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
         tfidf_words = set(tfidf_model.get_feature_names())
```

```
In [80]: tfidf_w2v_vectors = [];
         for sentence in tqdm(preprocessed_essays):
             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
             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((s
                     tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
                     vector += (vec * tf_idf) # calculating tfidf weighted w2v
                     tf_idf_weight += tf_idf
             if tf_idf_weight != 0:
                 vector /= tf_idf_weight
             tfidf_w2v_vectors.append(vector)

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

```
100%|| 4000/4000 [00:08<00:00, 485.54it/s]
```

```
4000
300
```

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

```
In [81]: # Similarly you can vectorize for title also
         tfidf_model = TfidfVectorizer()
         tfidf_model.fit(preprocessed_titles)
         # we are converting a dictionary with word as a key, and the idf as a value
         dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
         tfidf_words = set(tfidf_model.get_feature_names())
```

```
In [82]: tfidf_w2v_title_vectors = [];
         for sentence in tqdm(preprocessed_titles):
             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
             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((s
                     tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # 
                     vector += (vec * tf_idf) # calculating tfidf weighted w2v
                     tf_idf_weight += tf_idf
             if tf_idf_weight != 0:
                 vector /= tf_idf_weight
             tfidf_w2v_title_vectors.append(vector)

         print(len(tfidf_w2v_title_vectors))
         print(len(tfidf_w2v_title_vectors[0]))
```

```
100%|| 4000/4000 [00:00<00:00, 29036.67it/s]
```

```
4000
300
```

2.2.3 1.4.3 Vectorizing Numerical features

```
In [83]: # check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
         # standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.
         from sklearn.preprocessing import StandardScaler

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

         price_scalar = StandardScaler()
         price_scalar.fit(final_4000['price'].values.reshape(-1,1)) # finding the mean and sta
         print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.va

         # Now standardize the data with above maen and variance.
         price_standardized = price_scalar.transform(final_4000['price'].values.reshape(-1, 1)

Mean : 326.33975749999996, Standard deviation : 354.2030430407342
```

```
In [84]: price_standardized

Out[84]: array([[-0.49824461],
                [ 0.46182619],
                [-0.12004346],
                ...,
                [-0.44864029],
                [ 0.52814409],
                [-0.11617562]])
```

```
In [85]: # standardizing teacher_number_of_previously_posted_projects

         standard_scalar = StandardScaler()
         standard_scalar.fit(final_4000['teacher_number_of_previously_posted_projects'].values
         print(f"Mean : {standard_scalar.mean_[0]}, Standard deviation : {np.sqrt(standard_sca

         # Now standardize the data with above maen and variance.
         teacher_previous_projects_standardized = standard_scalar.transform(final_4000['teache

Mean : 9.51025, Standard deviation : 24.70468366398364
```

```
In [86]: teacher_previous_projects_standardized

Out[86]: array([[-0.34447921],
                [-0.38495737],
                [ 0.62699649],
                ...,
                [-0.1825666 ],
                [ 0.38412757],
                [-0.38495737]])
```

2.2.4 1.4.4 Merging all the above features

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

```
In [87]: print(categories_one_hot.shape)
         print(sub_categories_one_hot.shape)
         print(essays_bow.shape)
         print(price_standardized.shape)
```

```
(4000, 9)
(4000, 30)
(4000, 3824)
(4000, 1)
```

```
In [88]: # 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 mati
         X = hstack((categories_one_hot, sub_categories_one_hot, essays_bow, price_standardize
         X.shape
```

```
Out[88]: (4000, 3864)
```

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_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)</li>
        <li>price : numerical</li>
        <li>teacher_number_of_previously_posted_projects : numerical</li>
     </ul>
</li>
<li> Now, plot FOUR t-SNE plots with each of these feature sets.
    <ol>
        <li>categorical, numerical features + project_title(BOW)</li>
        <li>categorical, numerical features + project_title(TFIDF)</li>
        <li>categorical, numerical features + project_title(AVG W2V)</li>
        <li>categorical, numerical features + project_title(TFIDF W2V)</li>
    </ol>
</li>
<li> Concatenate all the features and Apply TNSE on the final data matrix </li>
<li> <font color='blue'>Note 1: The TSNE accepts only dense matrices</font></li>
<li> <font color='blue'>Note 2: Consider only 5k to 6k data points to avoid memory issues. If y
```

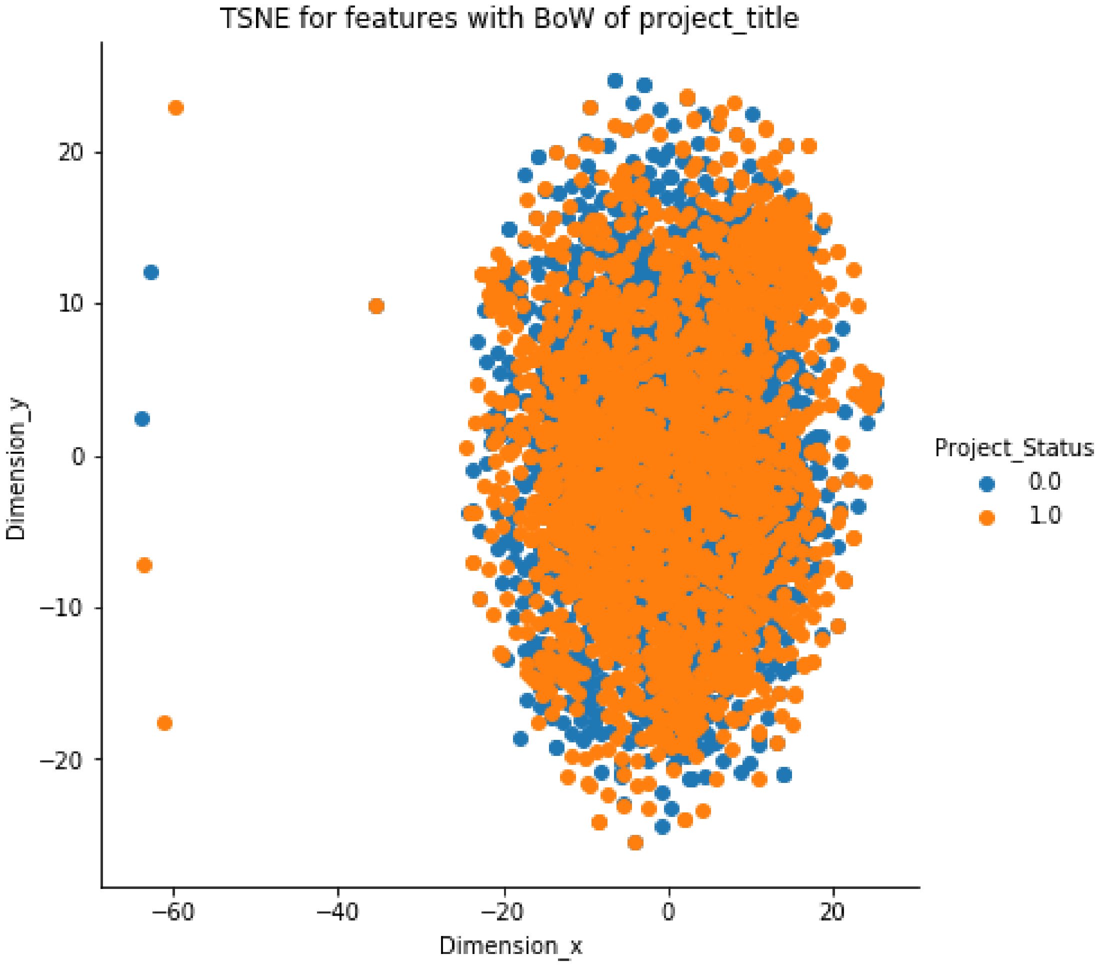
2.1 TSNE with `BOW` encoding of `project_title` feature

```
In [89]: #https://stackoverflow.com/questions/394809/does-python-have-a-ternary-conditional-op
         from sklearn.manifold import TSNE
         from sklearn import datasets
```

```
def plot_tsne(vectors,y_labels, isSparce, title):
    tsne = TSNE(n_components=2,random_state=0, perplexity=50, n_iter=1000)
    # since vectors is a sparse matrix we need to pass it as X_embedding = tsne.fit_t
    X_embedding = tsne.fit_transform(vectors.toarray()) if isSparce else tsne.fit_tran
    for_tsne = np.vstack((X_embedding.T, y_labels)).T
    tsne_df = pd.DataFrame(data=for_tsne, columns=['Dimension_x','Dimension_y','Projec
    # Ploting the result of tsne
    sns.FacetGrid(tsne_df, hue="Project_Status", size=6).map(plt.scatter, 'Dimension_x
    plt.title(title)
    plt.show()
```

In [90]:

```
y_labels = final_4000['project_is_approved'] # we are only considering 4000 rows

# we have stacked our categorical, numerical features with bag of word of project_tit
X = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,
            teacher_prefix_one_hot, essays_bow, price_standardized, teacher_previous_p

plot_tsne(X ,y_labels, True, 'TSNE for features with BoW of project_title')
```

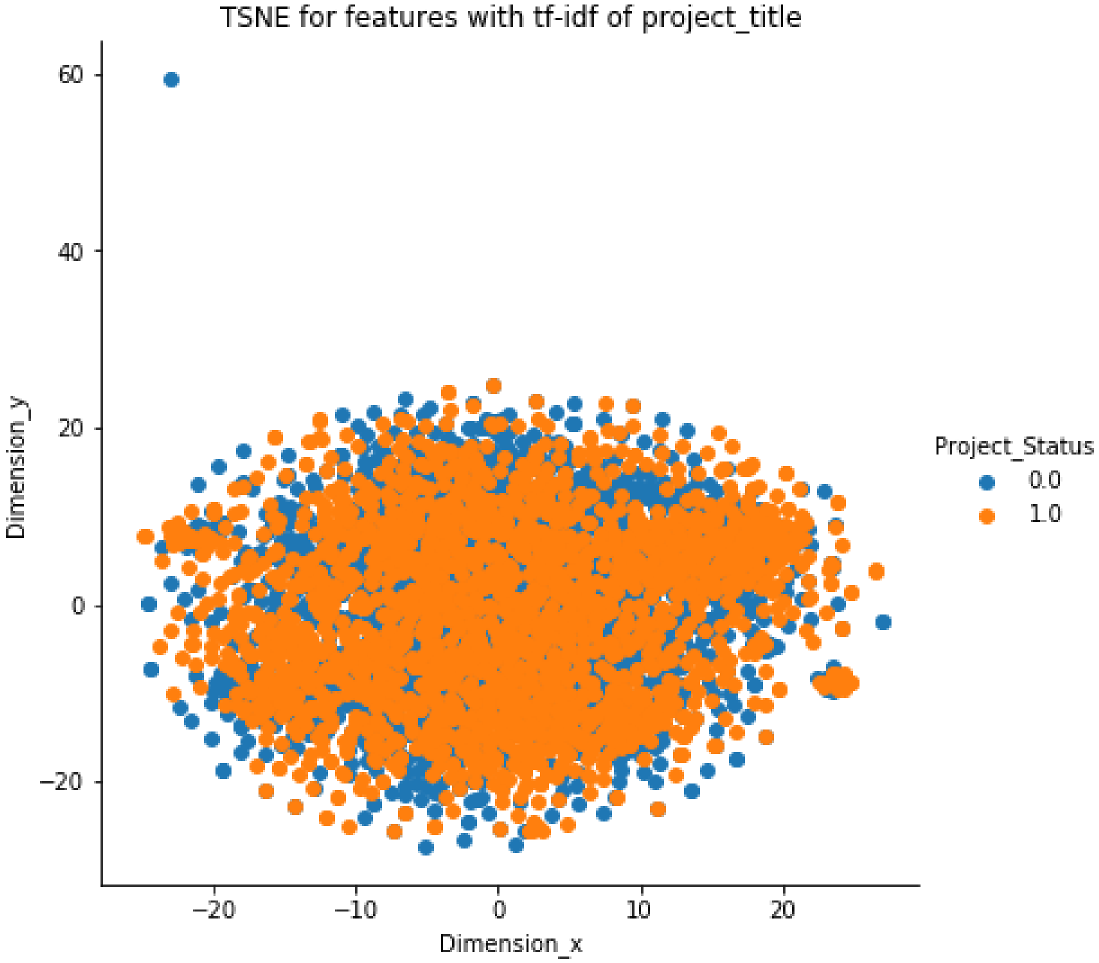


Observations:

- We can see that the points representing project status approved and rejected are highly overlapped.
- So, we are unable to draw a plane to separate project status based on these feature set.

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

```
In [91]: # we have stacked our categorical, numerical features with tf-idf of project_title
         X = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,
                     teacher_prefix_one_hot, essays_bow, price_standardized, teacher_previous_p
         plot_tsne(X,y_labels, True, 'TSNE for features with tf-idf of project_title')
```

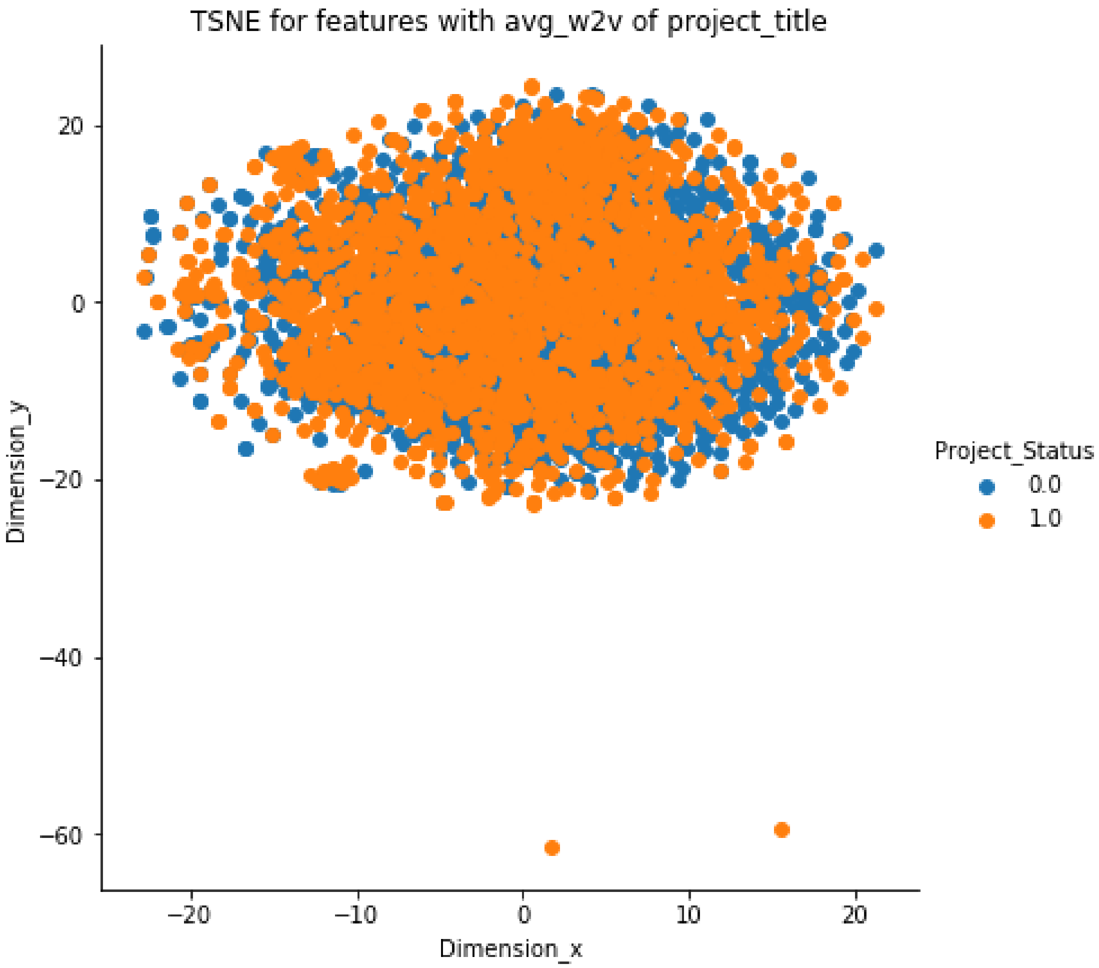


Observations:

- Similar to t-SNE for Bag of Words this plot also reflects huge overlapping of both categories.
- So, we are unable to draw a plane to separate project status based on these feature set.

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

```
In [92]: # we have stacked our categorical, numerical features with average W2V of project_tit

         X = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,
                     teacher_prefix_one_hot, essays_bow, price_standardized, teacher_previous_p
         plot_tsne(X,y_labels, True, 'TSNE for features with avg_w2v of project_title')
```

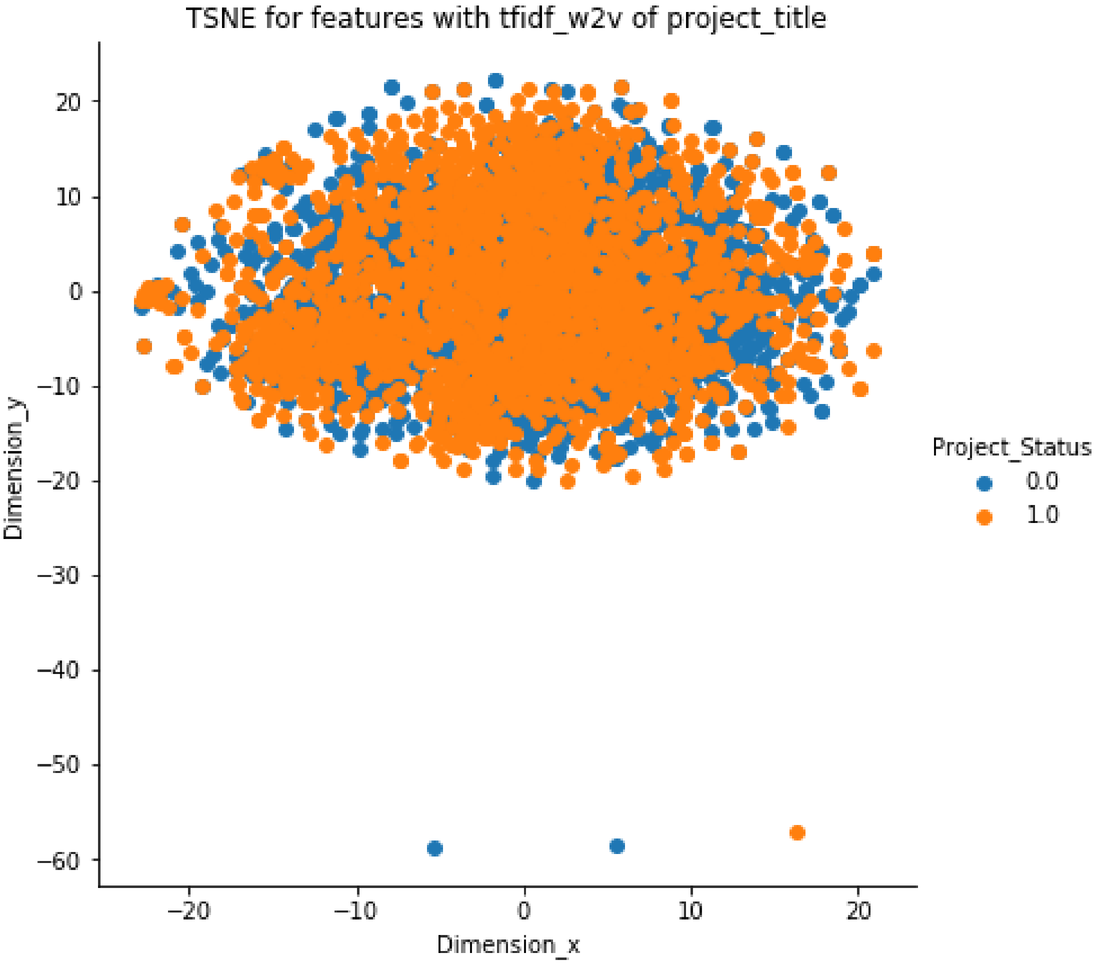


Observations:

- Similar to t-SNE for Bag of Words and tf-idf this plot also reflects huge overlapping of both categories.
- So, we are unable to draw a plane to separate project status based on these feature set.

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

```
In [93]: # we have stacked our categorical, numerical features with tf-idf weighted W2V of pro

         X = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,
                     teacher_prefix_one_hot, essays_bow, price_standardized, teacher_previous_
         plot_tsne(X,y_labels, True, 'TSNE for features with tfidf_w2v of project_title')
```

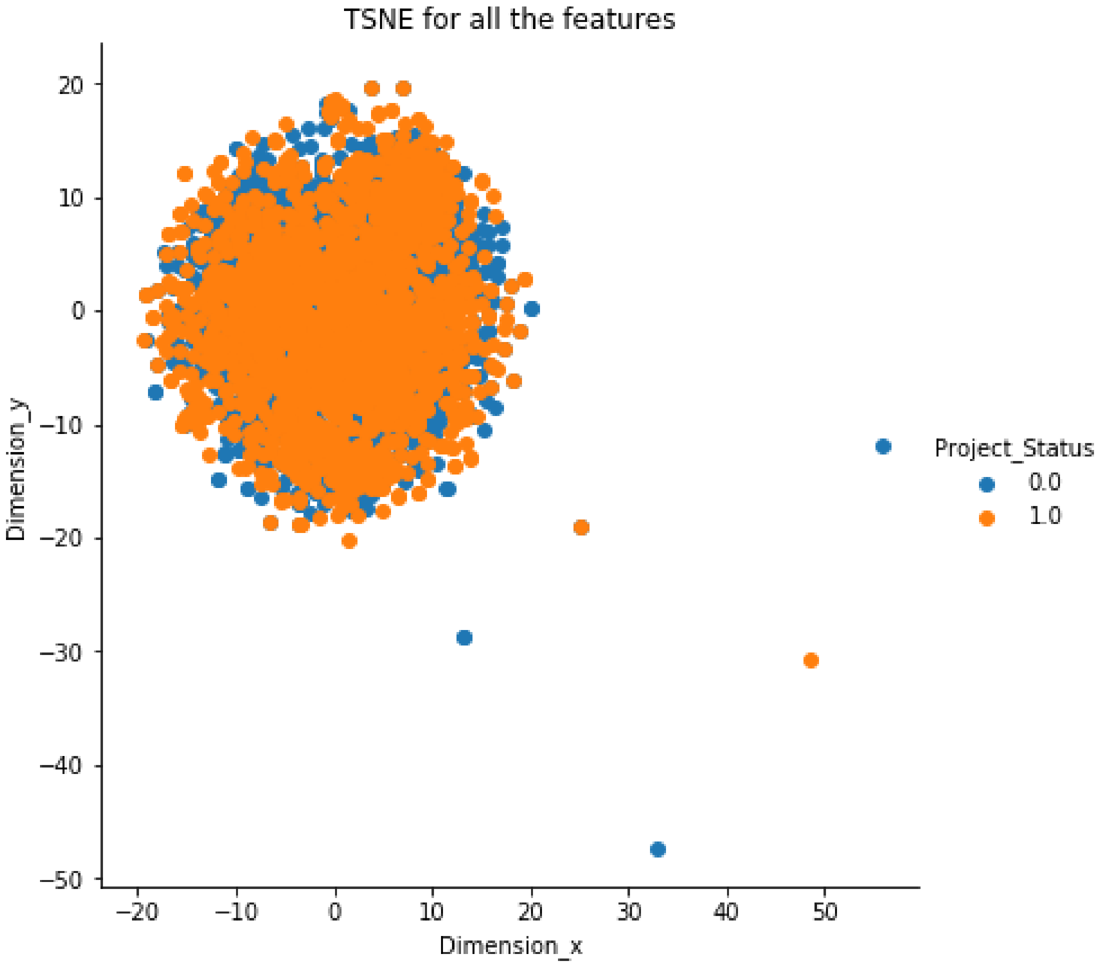


Observations:

- This plot also shows huge overlap between both categories.
- So, we are unable to draw a plane to separate project status based on these feature set.

2.5 Concatenating all the features and Apply TNSE on the final data matrix

```
In [94]: # we have stacked our categorical, numerical features with above 4 kind of features f

         X = hstack((school_state_one_hot, categories_one_hot, sub_categories_one_hot,
                     teacher_prefix_one_hot, essays_bow, price_standardized,
                     teacher_previous_projects_standardized,title_bow,title_tfidf,
                     avg_w2v_title_vector,tfidf_w2v_title_vectors))

         plot_tsne(X,y_labels, True, 'TSNE for all the features')
```



Observations:

- Even after concatenating all the features the tsne plot shows huge overlap.
- So, we are unable to draw a plane to separate project status based on these feature set.

2.6 Summary

- donorschoose dataset is imbalanced dataset as number of approved projects is very large as compared to rejected projects.
- Projects with multiple project_categories has higher approval rate.(e.g Literacy_Language Math_Science)

- From above TSNE plots none of the plot gives us clear separation of both categories.
- So, based on above features we can not draw a plane to separate both the accepted and rejected projects.
- We might have to consider some other approach that will fit well in this situation.