2_DonorsChoose_EDA_TSNE

February 5, 2019

1 DonorsChoose

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

```
Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result

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

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

1.1 About the DonorsChoose Data Set

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

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

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

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

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

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

Literacy
Literature & Writing, Social Sciences

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

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

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

nan
Dr.
Mr.
Mrs.
Ms.
Teacher.

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

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

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

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`

Feature	Description
`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 TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.2 1.1 Reading Data

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

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
```

```
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

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

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

Out[4]:

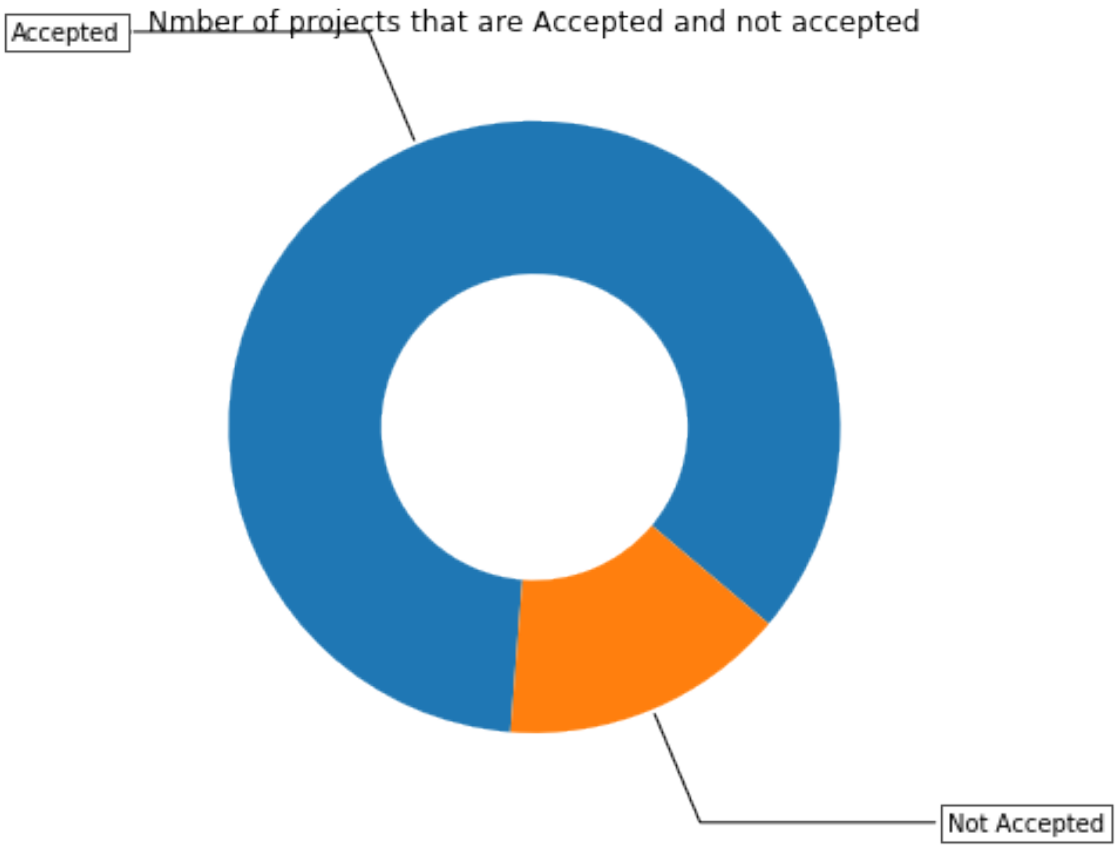
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

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 %)
```



2.0.1 1.2.1 Univariate Analysis: School State

In [7]:
```
# 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')
        '''
```

Out[7]: '# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n\nscl =

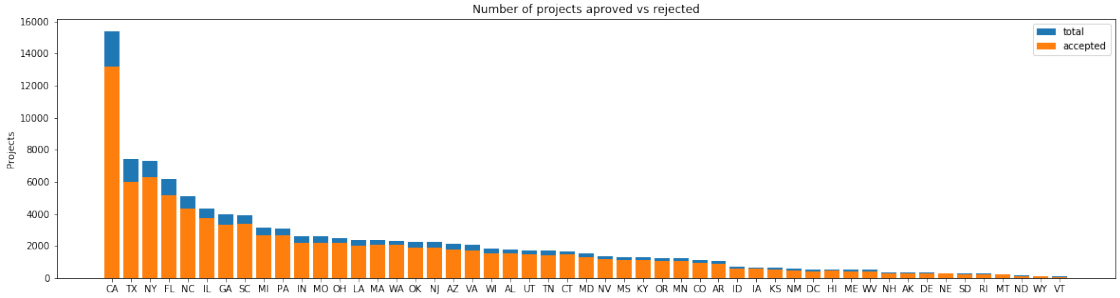
```
In [8]: # https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.
        temp.sort_values(by=['num_proposals'], inplace=True)
        print("States with lowest % approvals")
        print(temp.head(5))
        print('='*50)
        print("States with highest % approvals")
        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
```

As we can see %approval is different for different states hence state code is an important feature

```
In [9]: #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 [10]: def univariate_barplots(data, col1, col2='project_is_approved', top=False):
             # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521
             temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum()))

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

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



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

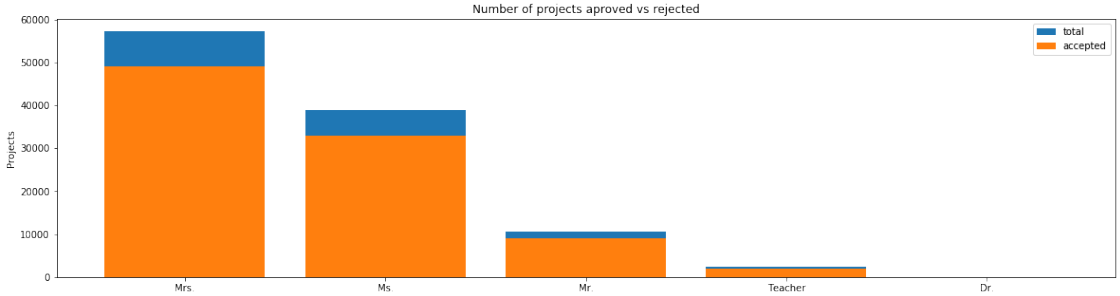
==========

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

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

2.0.2 1.2.2 Univariate Analysis: teacher_prefix

```
In [12]: univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False
```



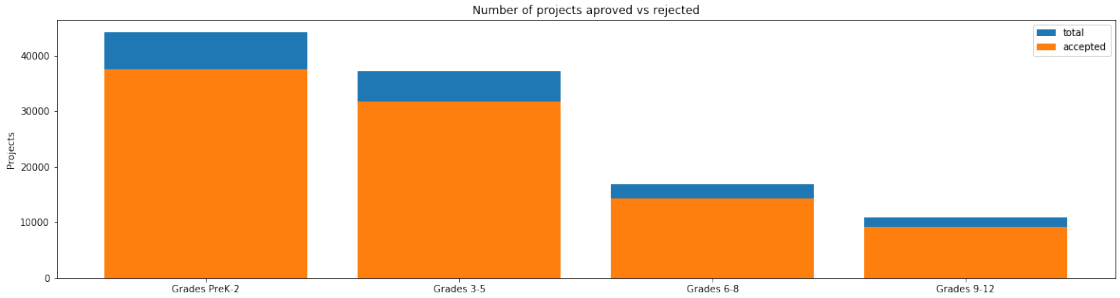
	teacher_prefix	project_is_approved	total	Avg
2	Mrs.	48997	57269	0.855559
3	Ms.	32860	38955	0.843537

```
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
```

If Teacher is either Mrs., Ms. or Mr. then chances of project approval is high

2.0.3 1.2.3 Univariate Analysis: project_grade_category

```
In [13]: 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
```

projects are posted mostly for Grades PreK-2 and Grades 3-5

2.0.4 1.2.4 Univariate Analysis: project_subject_categories

```
In [15]: catogories = list(project_data['project_subject_categories'].values)
         # remove special characters from list of strings python: https://stackoverflow.com/a/
```

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

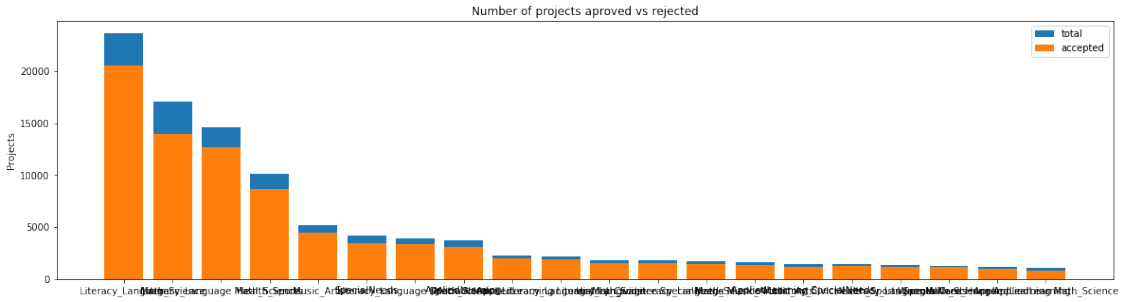
In [16]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[16]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_subject_subcategories	project_title	project_essay_1	project_essay_2	project_essay_3	project_essay_4	project_resource_summary
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	ESL, Literacy	Educational Support for English Learners at Home	My students are English learners that are work...	\"The limits of your language are the limits o...	NaN	NaN	My students need opportunities to practice beg...
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Civics & Government, Team Sports	Wanted: Projector for Hungry Learners	Our students arrive to our school eager to lea...	The projector we need for our school is very c...	NaN	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 [18]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=20)
```



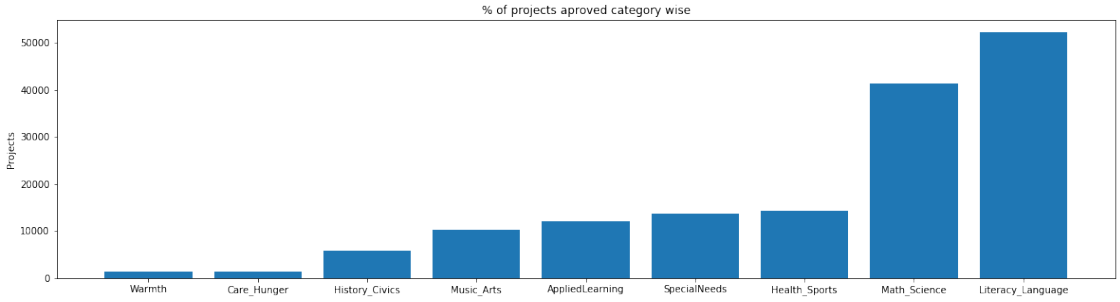

```
                clean_categories  project_is_approved  total       Avg
24              Literacy_Language                20520  23655  0.867470
32                   Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12725  14636  0.869432
8                   Health_Sports                 8640  10177  0.848973
40                     Music_Arts                 4429   5180  0.855019
==================================================
                   clean_categories  project_is_approved  total       Avg
19  History_Civics Literacy_Language                 1271   1421  0.894441
14        Health_Sports SpecialNeeds                 1215   1391  0.873472
50               Warmth Care_Hunger                  1212   1309  0.925898
33      Math_Science AppliedLearning                 1019   1220  0.835246
4       AppliedLearning Math_Science                  855   1052  0.812738
```

There is lot of variability in %project approval per clean categories. If the project fall under Warm care_hunger then will be high chances of project approval

In [19]:
```
# 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 [20]:
```
# 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()
```



Lot of Projects which approved are mainly from math science and Literacy language category

```
In [22]: 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 [23]: 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
```

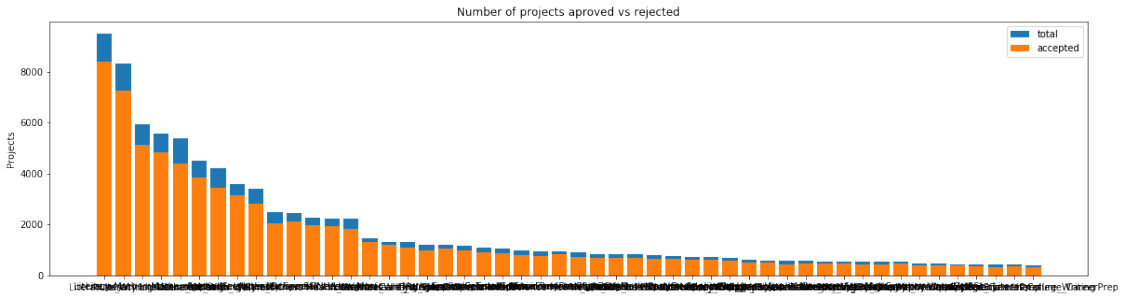
```
sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warm
        if 'The' in j.split(): # this will split each of the catogory based 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 [24]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[24]:

	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
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educational Support for English Learners at Home	My students are English learners that are work...	\"The limits of your language are the limits o...	NaN	NaN	My students need opportunities to practice beg...	0	0	Literacy_Language	ESL Literacy
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Wanted: Projector for Hungry Learners	Our students arrive to our school eager to lea...	The projector we need for our school is very c...	NaN	NaN	My students need a projector to help with view...	7	1	History_Civics Health_Sports	Civics_Government TeamSports

```
In [25]: 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

==========

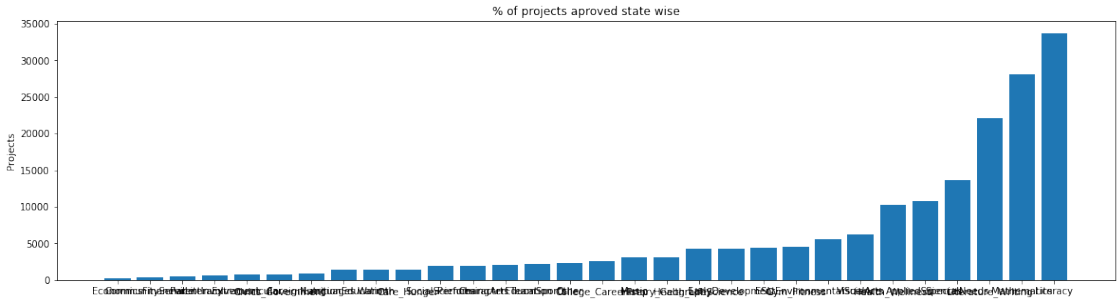
	clean_subcategories	project_is_approved	total	Avg
196	EnvironmentalScience Literacy	389	444	0.876126
127	ESL	349	421	0.828979
79	College_CareerPrep	343	421	0.814727
17	AppliedSciences Literature_Writing	361	420	0.859524
3	AppliedSciences College_CareerPrep	330	405	0.814815

In subcategories, Literacy had the max no of project approved also having the higher approval %

```
In [26]: # 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 [27]: # 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()
```



State wise Literature_Writing had the highest approval with 22179 no of approved project then Mathematics with no of approved project as 28074 and after then we had Literacy with 22179 no of approved project

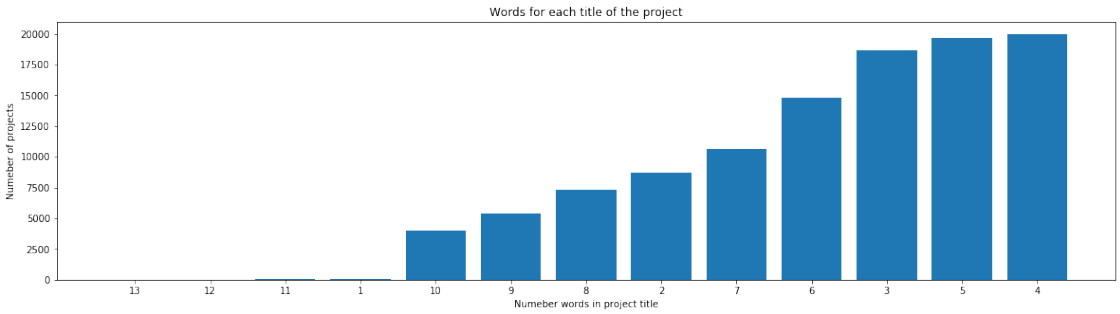
```
In [28]: for i, j in sorted_sub_cat_dict.items():
             print("{:20} :{:10}".format(i,j))
```

```
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
```

2.0.6 1.2.6 Univariate Analysis: Text features (Title)

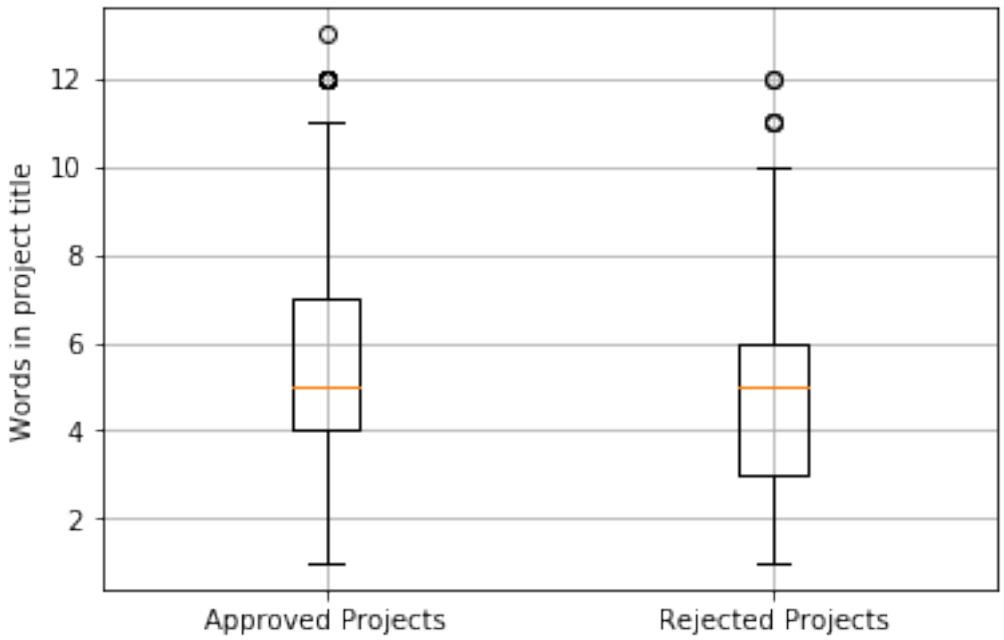
In [29]:
```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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()
```



Project that have 3-4 words are more in numbers

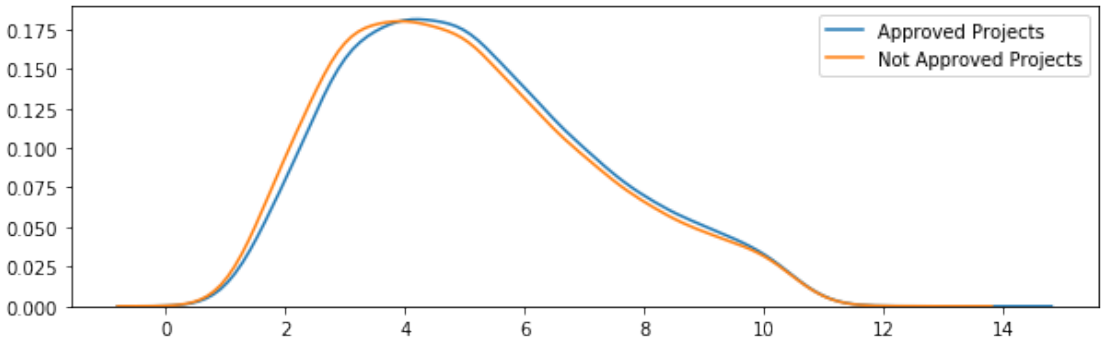
In [30]:
```
approved_title_word_count = project_data[project_data['project_is_approved']==1]['pro
approved_title_word_count = approved_title_word_count.values

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

In [32]:
```
# 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()
```



If the no. of words in project title is more than 6 then chances of getting approved will be high simillarly if the words in project titler is less than 4 then the chances of getting rejected is more.

```
In [33]: 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()
```



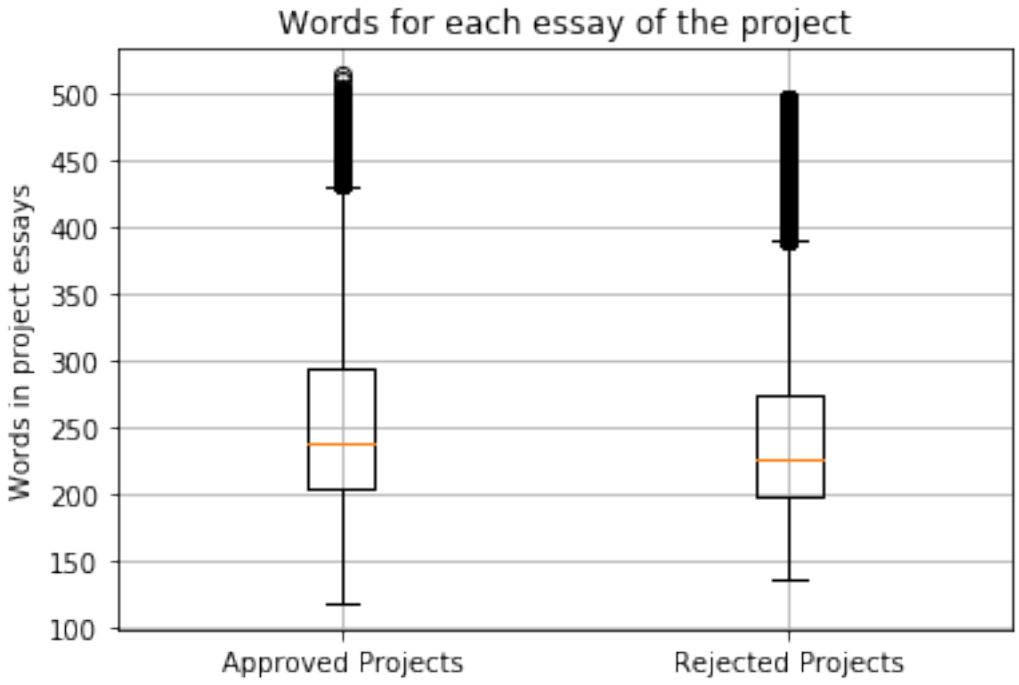
If the word count is below 4 then chances of project getting rejected is sligthly more and vice versa if word count is above 4

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

```
In [34]: # 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)
```

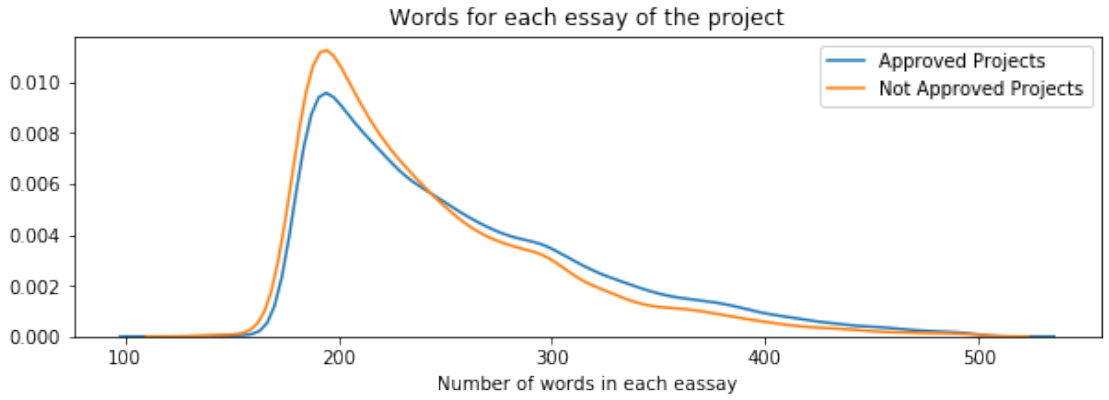
```
In [35]: approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].st
         approved_word_count = approved_word_count.values

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

```
In [36]: # 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()
```



If words in project essay more than 275 then chances is more of getting approved but there is also outlier present So to be on safer side we can say if word count of project essay is between 275 to 280 then chances of getting approved is more

```
In [37]: 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()
```



Project which are rejected has around 170 to 240 word count of project essay and those which are accepted have the word count between 300 to 400

2.0.8 1.2.8 Univariate Analysis: Cost per project

```
In [40]: # we get the cost of the project using resource.csv file
         resource_data.head(2)
```

```
Out[40]:         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
```

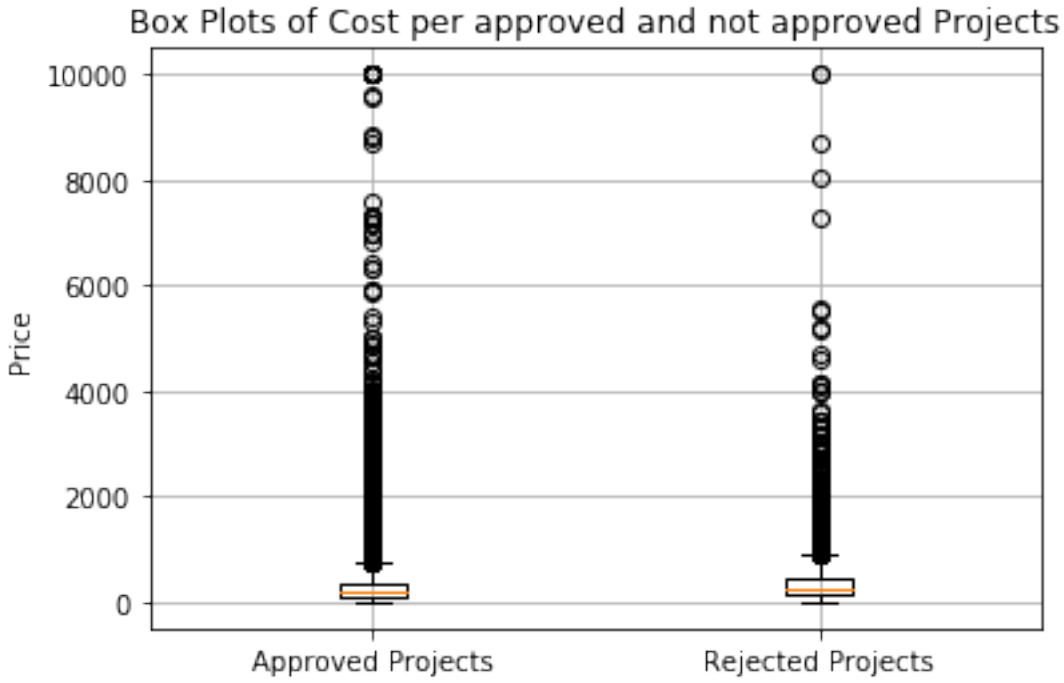
```
In [41]: # https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-
         price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_
         price_data.head(2)
```

```
Out[41]:         id   price  quantity
         0  p000001  459.56         7
         1  p000002  515.89        21
```

```
In [42]: # join two dataframes in python:
         project_data = pd.merge(project_data, price_data, on='id', how='left')
```

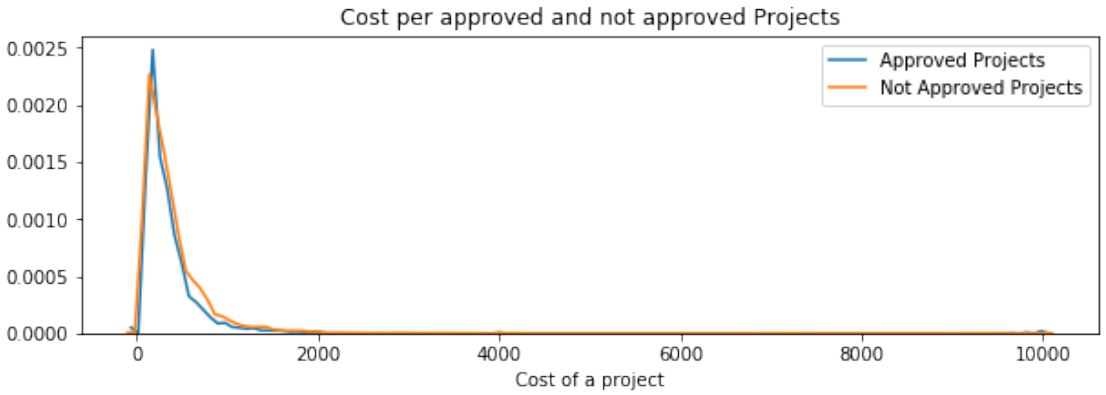
In [43]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].values

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

In [44]:
```
# 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()
```



can't say anything

In [45]:
```
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()
```



If project is approved then cost of it will be sligtly lesser than the project which are not approved

```
In [46]: # 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        |
+------------+--------------------+-----------------------+
```

Project which are approved are having the lower cost than the cost of Rejected projects
1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

2.0.9 Let's see how many data is stored for a each teacher who has submitted project proposal

```
In [48]: print("Number of unique teachers = {}".format(project_data['teacher_id'].nunique()))
         temp = project_data['teacher_id'].value_counts().sort_values(ascending=False).rename_a
         temp.head()
```

```
Number of unique teachers = 72168
```

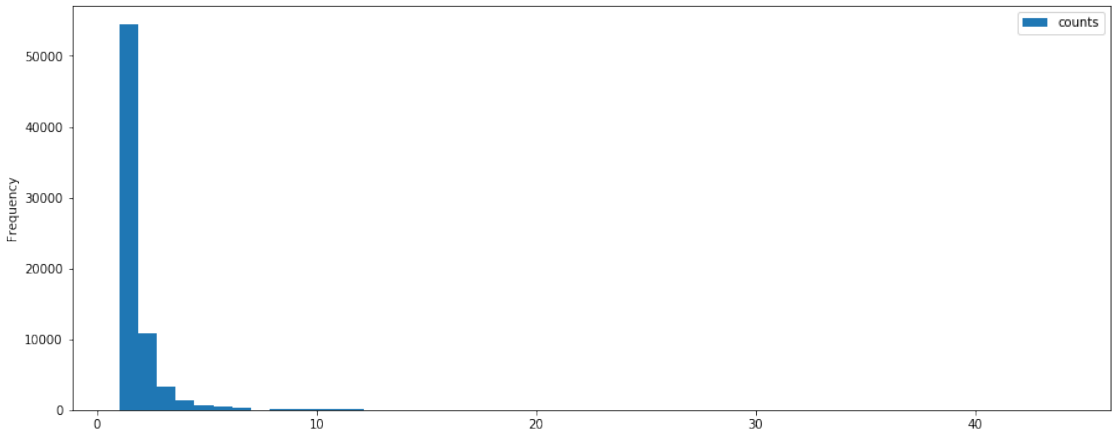
Out[48]:

	unique_teacher_id	counts
0	fa2f220b537e8653fb48878ebb38044d	44
1	df8a4b7ad173b57f7ac52e447cc24043	42
2	1f64dcec848be8e95c4482cc845706b2	42
3	7b17c95da53e3d1f011f84232ad01238	34
4	ae67d8bbc64ec3bf7fd2db1297721160	33

2.0.10 Graph between number of teachers and no of project posted (according to our dataset)

```
In [49]: temp.plot(kind='hist',bins=50,figsize=(15,6))
         print('No. of teacher posted only 1 project is: ',(sum(temp['counts']==1)))
```

```
No. of teacher posted only 1 project is:  54429
```



As you can see the no. of teacher posted only one project is more and is equal to 54429

2.0.11 Let's see the history related to previously posted project of teacher with a given id

```
In [50]: project_data[project_data['teacher_id']=='5de88a574b531a9b11b0b63f664d13e9'][['teache
```

Out[50]:

	teacher_id	project_submitted_datetime	\
103386	5de88a574b531a9b11b0b63f664d13e9	2016-08-02 11:45:06	
90064	5de88a574b531a9b11b0b63f664d13e9	2016-08-04 15:52:32	
16253	5de88a574b531a9b11b0b63f664d13e9	2016-08-24 21:10:29	
65426	5de88a574b531a9b11b0b63f664d13e9	2016-10-11 08:27:51	
96751	5de88a574b531a9b11b0b63f664d13e9	2016-12-19 15:23:32	

	project_title	\
103386	Flexible and Healthy seating options!	
90064	Tablets to ensure differentiation and financia...	
16253	Literacy Program for Kinders!	
65426	Laptop for Centers and Projects!	
96751	Snacks for my students!	

	teacher_number_of_previously_posted_projects	project_is_approved	\
103386	11	0	
90064	11	0	
16253	15	0	
65426	16	1	
96751	20	0	

	clean_categories
103386	Health_Sports
90064	Math_Science
16253	AppliedLearning Literacy_Language
65426	Literacy_Language Math_Science
96751	Health_Sports

2.0.12 creating a dataframe consiting of teacher with a unique id and the no of projects posted by them

```
In [51]: teacher_data = project_data.copy()
         teacher_data.sort_values('teacher_number_of_previously_posted_projects', ascending=Fa
         teacher_data['Total_number_of_project_submitted_so_far'] = teacher_data['teacher_numb
         temp1 = teacher_data[['teacher_id','Total_number_of_project_submitted_so_far']].sort_
```

2.0.13 List of 5 teachers who posted Max no of projects:

```
In [52]: temp1.tail(5)[::-1]
```

Out[52]:

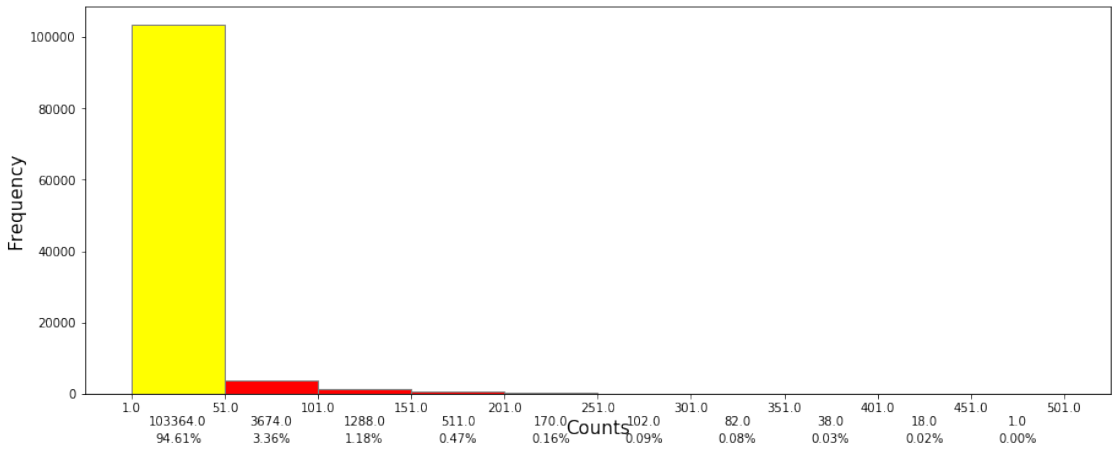
	teacher_id	\
88015	fa2f220b537e8653fb48878ebb38044d	
106026	fa2f220b537e8653fb48878ebb38044d	
50805	fa2f220b537e8653fb48878ebb38044d	
72233	fa2f220b537e8653fb48878ebb38044d	

```
13777    fa2f220b537e8653fb48878ebb38044d

         Total_number_of_project_submitted_so_far
88015                                         452
106026                                        438
50805                                         434
72233                                         433
13777                                         429
```

2.0.14 Histogram plot for total no. of project posted

In [53]:
```
# Code taken from https://stackoverflow.com/a/6353051/6660373. Modified as per requir
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.ticker import FormatStrFormatter
binwidth = 50
data = temp1['Total_number_of_project_submitted_so_far'].values
fig, ax = plt.subplots(figsize=(15,6))
counts, bins, patches = ax.hist(data, facecolor='yellow', edgecolor='gray',bins=range

# Set the ticks to be at the edges of the bins.
ax.set_xticks(bins)
# Set the xaxis's tick labels to be formatted with 1 decimal place...
ax.xaxis.set_major_formatter(FormatStrFormatter('%0.1f'))

# Change the colors of bars at the edges...
twentyfifth, seventyfifth = np.percentile(data, [25, 75])
for patch, rightside, leftside in zip(patches, bins[1:], bins[:-1]):
    if rightside < twentyfifth:
        patch.set_facecolor('green')
    elif leftside > seventyfifth:
        patch.set_facecolor('red')

# Label the raw counts and the percentages below the x-axis...
bin_centers = 0.5 * np.diff(bins) + bins[:-1]
for count, x in zip(counts, bin_centers):
    # Label the raw counts
    ax.annotate(str(count), xy=(x, 0), xycoords=('data', 'axes fraction'),
        xytext=(0, -18), textcoords='offset points', va='top', ha='center')

    # Label the percentages
    percent = "{0:.2f}%".format(100 *float(count) / counts.sum())
    ax.annotate(percent, xy=(x, 0), xycoords=('data', 'axes fraction'),
        xytext=(0, -32), textcoords='offset points', va='top', ha='center')

ax.set_xlabel('Counts', fontsize=15)
ax.set_ylabel('Frequency', fontsize=15)   # relative to plt.rcParams['font.size']
```

```
        # Give ourselves some more room at the bottom of the plot
        plt.subplots_adjust(bottom=0.15)
        plt.show()
```



94% of the teacher posted project in 1 to 51 numbers

2.0.15 Let's see how many projects are approved for each teacher

```
In [54]: col1 = 'teacher_id'
         col2 = 'project_is_approved'
         total = 'total_project_present_in_data'
         tp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).reset

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

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


         print(tp.head(5))
         print("="*50)
         print(tp.tail(5))
```

```
                         teacher_id  project_is_approved  \
70484  fa2f220b537e8653fb48878ebb38044d                   44
8702   1f64dcec848be8e95c4482cc845706b2                   40
62925  df8a4b7ad173b57f7ac52e447cc24043                   40
34570  7b17c95da53e3d1f011f84232ad01238                   34
49149  ae67d8bbc64ec3bf7fd2db1297721160                   31

       total_project_present_in_data       Avg
70484                             44  1.000000
```

```
8702                              42  0.952381
62925                             42  0.952381
34570                             34  1.000000
49149                             33  0.939394
================================================
                            teacher_id  project_is_approved  \
27489  61fdee5b0c34ea70671f52b3c3a01b71                    1
27490  61ff263134490840c2a7f3b11e21cf0e                    1
27491  61ff51943e34dba224a319b49a4d44fa                    0
27492  61ff58caffe3bfef6e929db19e808aeb                    1
72167  ffff8e040521f62207881376ecc964d5                    1

       total_project_present_in_data  Avg
27489                              1  1.0
27490                              1  1.0
27491                              1  0.0
27492                              1  1.0
72167                              1  1.0
```

Here are the list of 5 teachers with highest no of submission and with lowest no of submission respectively. As you can see some of the teacher has got their all the project accepted

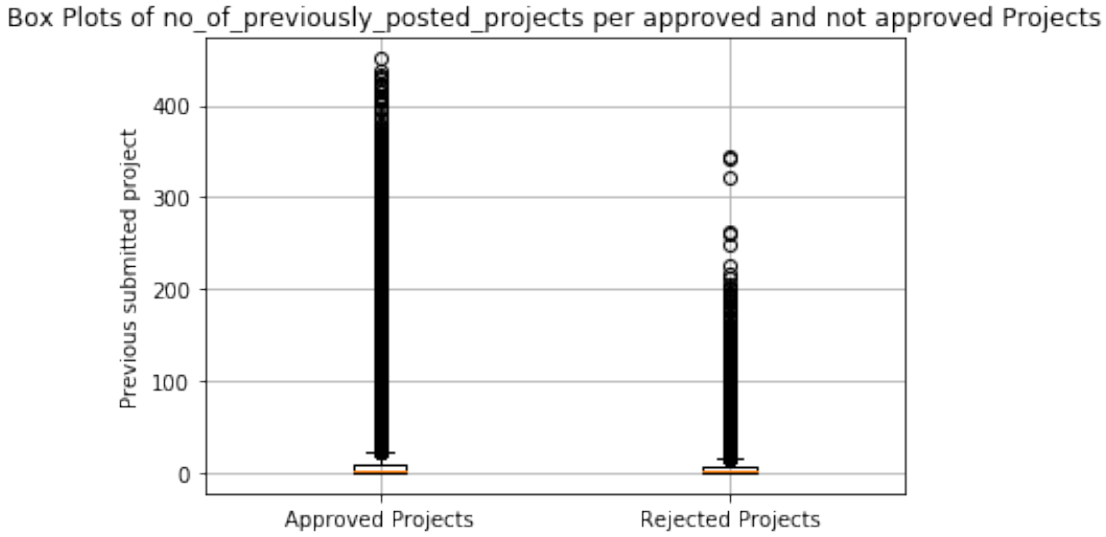
```
In [47]: #univariate_barplots(project_data, 'teacher_id', 'project_is_approved', False)

In [55]: teacher_data.drop(['project_is_approved'], axis = 1, inplace = True, errors = 'ignore

         teacher_proj_data = pd.concat([teacher_data.set_index('teacher_id'),tp.set_index('tea
         teacher_proj_data[['teacher_id','Total_number_of_project_submitted_so_far','project_is
```

Out[55]:

```
                          teacher_id  Total_number_of_project_submitted_so_far  \
0  c90749f5d961ff158d4b4d1e7dc665fc                                          1
1  897464ce9ddc600bced1151f324dd63a                                          8
2  3465aaf82da834c0582ebd0ef8040ca0                                          2
3  f3cb9bffbba169bef1a77b243e620b60                                          5
4  be1f7507a41f8479dc06f047086a39ec                                          2

   project_is_approved  total_project_present_in_data       Avg
0                    0                             1  0.000000
1                    4                             4  1.000000
2                    2                             3  0.666667
3                    2                             2  1.000000
4                    1                             1  1.000000
```

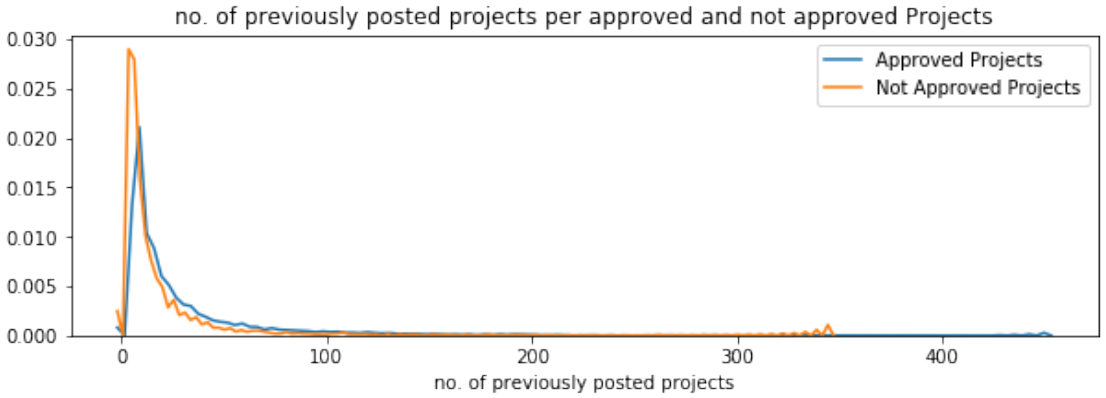
```
In [56]: approved_proj = project_data[project_data['project_is_approved']==1]['teacher_number_

         rejected_proj = project_data[project_data['project_is_approved']==0]['teacher_number_
         # https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
```

```
plt.boxplot([approved_proj, rejected_proj])
plt.title('Box Plots of no_of_previously_posted_projects per approved and not approved
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Previous submitted project')
plt.grid()
plt.show()
```



can't say anything

In [57]:
```
plt.figure(figsize=(10,3))
sns.kdeplot(approved_proj,label="Approved Projects", bw=0.6)
sns.kdeplot(rejected_proj,label="Not Approved Projects", bw=0.6)
plt.title('no. of previously posted projects per approved and not approved Projects')
plt.xlabel('no. of previously posted projects')
plt.legend()
plt.show()
```



Project which are accepted have 30 to 90 no. of previously posted project

```
In [58]: # 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_proj,i), 3), np.round(np.percentile(
         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

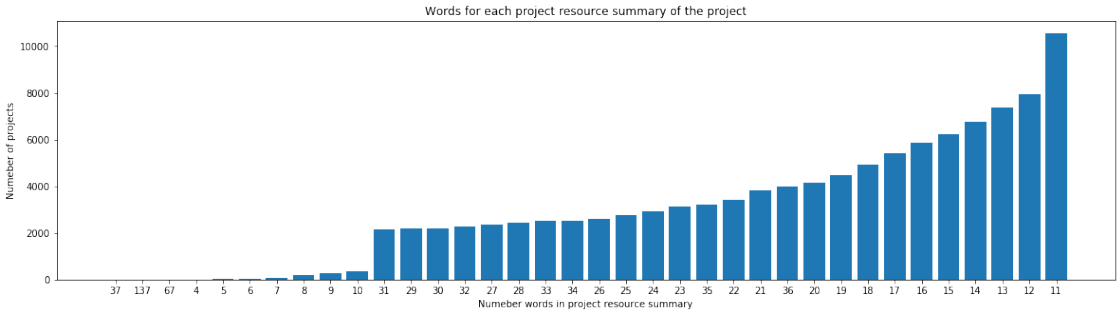
Project which are accepted have the no. of previously posted project higher than that of rejected ones

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

1.2.10 Univariate Analysis: project_resource_summary

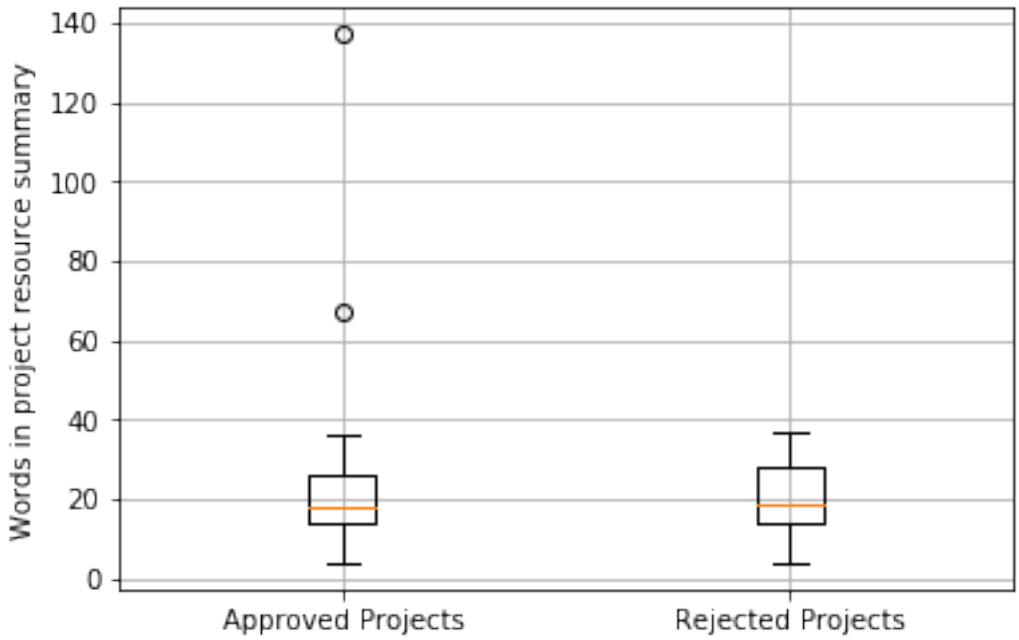
```
In [59]: #How to calculate number of words in a string in DataFrame: https://stackoverflow.com
         word_count = project_data['project_resource_summary'].str.split().apply(len).value_co
```

```
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 resource summary')
plt.title('Words for each project resource summary of the project')
plt.xticks(ind, list(word_dict.keys()))
plt.show()
```



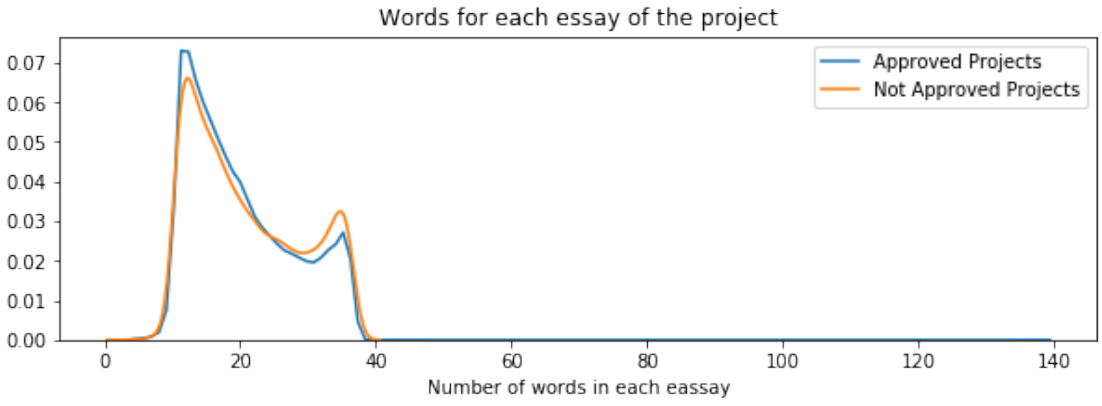
Projects which are having more no. of words are larger in numbers.

```
In [60]: approved_summary_word_count = project_data[project_data['project_is_approved']==1]['pr
         approved_summary_word_count = approved_summary_word_count.values

         rejected_summary_word_count = project_data[project_data['project_is_approved']==0]['pr
         rejected_summary_word_count = rejected_summary_word_count.values

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



Can't infer much detail as they are overlapping

```
In [61]: plt.figure(figsize=(10,3))
         sns.distplot(approved_summary_word_count, hist=False, label="Approved Projects")
         sns.distplot(rejected_summary_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()
```



Can't infer much info

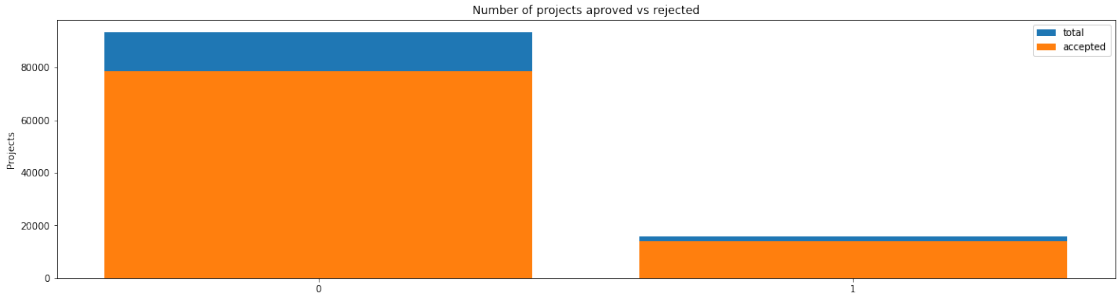
2.0.16 To find whether numerical digits in project summary exist or not

```
In [62]: #re.findall(r'\d+',)
         def find_num(text):
             if re.findall(r'\d+', text):
                 return 1
             return 0
```

```
In [114]: project_data['numerical_digits'] = project_data['project_resource_summary'].apply(la
          # https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for
          num_data = project_data.groupby('id').agg({'numerical_digits':'sum'}).reset_index()
          #project_data = pd.merge(project_data, num_data, on='id', how='left').reset_index()
```

```
In [112]: project_data['numerical_digits'] = project_data['numerical_digits'].apply(str).head()
```

```
In [115]: univariate_barplots(project_data, 'numerical_digits', 'project_is_approved', top=20)
```




```
   numerical_digits  project_is_approved  total       Avg
0                 0                78616  93492  0.840885
1                 1                14090  15756  0.894263
==================================================
   numerical_digits  project_is_approved  total       Avg
0                 0                78616  93492  0.840885
1                 1                14090  15756  0.894263
```

Project which are approved have the numerical digits present but not all of the project with numerical digit guranteed that the project will get approved. We can say that chances of getting the project approved is high when there is numerical digit present in the 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.

2.1 1.3 Text preprocessing

2.1.1 1.3.1 Essay Text

In [66]:
```
project_data.head(2)
```

Out[66]:

	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_4	project_resource_summary	teacher_number_of_previously_posted_projects	project_is_approved	clean_categories	clean_subcategories	essay	price	quantity	numerical_digits
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educational Support for English Learners at Home	My students are English learners that are work...	\"The limits of your language are the limits o...	...	NaN	My students need opportunities to practice beg...	0	0	Literacy_Language	ESL Literacy	My students are English learners that are work...	154.6	23	0
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Wanted: Projector for Hungry Learners	Our students arrive to our school eager to lea...	The projector we need for our school is very c...	...	NaN	My students need a projector to help with view...	7	1	History_Civics Health_Sports	Civics_Government TeamSports	Our students arrive to our school eager to lea...	299.0	1	0

[2 rows x 21 columns]

In [67]:
```
# printing some random essays.
print(project_data['essay'].values[0])
```

```
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, a
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogn
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The g
==================================================
```

In [68]:
```
# 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 [69]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

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

In [70]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

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

In [71]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cogni
```

In [72]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him'
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', '
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'a
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'throug
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', '
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'a
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'to
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", '
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mi
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
            'won', "won't", 'wouldn', "wouldn't"]
```

In [73]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|| 109248/109248 [02:23<00:00, 760.67it/s]
```

```
In [74]: # after preprocesing
         preprocessed_essays[20000]
```

```
Out[74]: 'my kindergarten students varied disabilities ranging speech language delays cognitive
```

1.3.2 Project title Text

```
In [75]: # Combining all the above statemennts
         from tqdm import tqdm
         preprocessed_titles = []
         # tqdm is for printing the status bar
         for sentance in tqdm(project_data['project_title'].values):
             sent = decontracted(sentance)
             sent = sent.replace('\\r', ' ')
             sent = sent.replace('\\"', ' ')
             sent = sent.replace('\\n', ' ')
             sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
             # https://gist.github.com/sebleier/554280
             sent = ' '.join(e for e in sent.split() if e not in stopwords)
             preprocessed_titles.append(sent.lower().strip())
```

```
100%|| 109248/109248 [00:06<00:00, 16491.06it/s]
```

```
In [76]: # after preprocesing
         preprocessed_titles[20000]
```

```
Out[76]: 'we need to move it while we input it'
```

2.2 1. 4 Preparing data for models

```
In [77]: project_data.columns
```

```
Out[77]: 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',
                'numerical_digits'],
               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 [120]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=Fals
vectorizer.fit(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', '
Shape of matrix after one hot encodig  (109248, 9)
```

In [124]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


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

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

In [122]:
```
# Please do the similar feature encoding with state, teacher_prefix and project_grad
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=Fals
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'I
Shape of matrix after one hot encodig  (109248, 9)
```

```
In [130]: categories_one_hot_num_dig = pd.get_dummies(project_data['numerical_digits'].values)
          print("Shape of matrix after one hot encodig ",categories_one_hot_num_dig.shape)
```

```
Shape of matrix after one hot encodig  (109248, 2)
```

```
In [82]: print(project_data['teacher_prefix'].unique())
         project_data[['teacher_id','teacher_prefix']].groupby('teacher_prefix').count()
```

```
['Mrs.' 'Mr.' 'Ms.' 'Teacher' nan 'Dr.']
```

```
Out[82]:                 teacher_id
         teacher_prefix
         Dr.                     13
         Mr.                  10648
         Mrs.                 57269
         Ms.                  38955
         Teacher               2360
```

```
In [83]: project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
```

```
In [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
         vectorizer.fit(project_data['teacher_prefix'].values)
         print(vectorizer.get_feature_names())


         categories_one_hot_teacher_prefix = vectorizer.transform(project_data['teacher_prefix
         print("Shape of matrix after one hot encodig ",categories_one_hot_teacher_prefix.shape
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'I
Shape of matrix after one hot encodig  (109248, 9)
```

```
In [85]: # 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(project_data['project_grade_category'].values)
         print(vectorizer.get_feature_names())


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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'I
Shape of matrix after one hot encodig  (109248, 9)
```

2.2.2 1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

```
In [86]: # We are considering only the words which appeared in at least 10 documents(rows or p
         vectorizer = CountVectorizer(min_df=10)
         text_bow_essays = vectorizer.fit_transform(preprocessed_essays)
         print("Shape of matrix after one hot encodig ",text_bow_essays.shape)
```

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

1.4.2.2 Bag of Words on `project_title`

```
In [87]: # you can vectorize the title also
         # before you vectorize the title make sure you preprocess it
         # We are considering only the words which appeared in at least 10 documents(rows or p
         vectorizer = CountVectorizer(min_df=10)
         text_bow_titles = vectorizer.fit_transform(preprocessed_titles)
         print("Shape of matrix after one hot encodig ",text_bow_titles.shape)
```

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

1.4.2.3 TFIDF vectorizer

```
In [88]: from sklearn.feature_extraction.text import TfidfVectorizer
         vectorizer = TfidfVectorizer(min_df=10)
         text_tfidf_essays = vectorizer.fit_transform(preprocessed_essays)
         print("Shape of matrix after one hot encodig ",text_tfidf_essays.shape)
```

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

1.4.2.4 TFIDF Vectorizer on `project_title`

```
In [89]: # Similarly you can vectorize for title also
         from sklearn.feature_extraction.text import TfidfVectorizer
         vectorizer = TfidfVectorizer(min_df=10)
         text_tfidf_titles = vectorizer.fit_transform(preprocessed_titles)
         print("Shape of matrix after one hot encodig ",text_tfidf_titles.shape)
```

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

1.4.2.5 Using Pretrained Models: Avg W2V

```
In [90]: '''
         # 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

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

```
        '''
```

```
Out[90]: '\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\nd
```

```
In [91]: # stronging variables into pickle files python: http://www.jessicayung.com/how-to-use
         # 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 [92]: # average Word2Vec
         # compute average word2vec for each review.
         avg_w2v_vectors_essays = []; # the avg-w2v for each sentence/review is stored in this
         for sentence in tqdm(preprocessed_essays): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             cnt_words =0; # num of words with a valid vector in the sentence/review
             for word in sentence.split(): # for each word in a review/sentence
                 if word in glove_words:
                     vector += model[word]
                     cnt_words += 1
             if cnt_words != 0:
                 vector /= cnt_words
             avg_w2v_vectors_essays.append(vector)

         print(len(avg_w2v_vectors_essays))
         print(len(avg_w2v_vectors_essays[0]))
```

```
100%|| 109248/109248 [01:08<00:00, 1597.08it/s]


109248
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

```
In [93]: # Similarly you can vectorize for title also
         # average Word2Vec
         # compute average word2vec for each review.
         avg_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stored in this
         for sentence in tqdm(preprocessed_titles): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             cnt_words =0; # num of words with a valid vector in the sentence/review
             for word in sentence.split(): # for each word in a review/sentence
                 if word in glove_words:
                     vector += model[word]
                     cnt_words += 1
```

```
            if cnt_words != 0:
                vector /= cnt_words
            avg_w2v_vectors_titles.append(vector)

        print(len(avg_w2v_vectors_titles))
        print(len(avg_w2v_vectors_titles[0]))
```

```
100%|| 109248/109248 [00:04<00:00, 23938.31it/s]


109248
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

```
In [94]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
         tfidf_model = TfidfVectorizer()
         tfidf_model.fit(preprocessed_essays)
         # we are converting a dictionary with word as a key, and the idf as a value
         dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
         tfidf_words = set(tfidf_model.get_feature_names())
```

```
In [95]: # average Word2Vec
         # compute average word2vec for each review.
         tfidf_w2v_vectors_essays = []; # the avg-w2v for each sentence/review is stored in th
         for sentence in tqdm(preprocessed_essays): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             tf_idf_weight =0; # num of words with a valid vector in the sentence/review
             for word in sentence.split(): # for each word in a review/sentence
                 if (word in glove_words) and (word in 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_essays.append(vector)

         print(len(tfidf_w2v_vectors_essays))
         print(len(tfidf_w2v_vectors_essays[0]))
```

```
100%|| 109248/109248 [08:11<00:00, 222.46it/s]


109248
300
```

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

```
In [96]: # Similarly you can vectorize for title also
         # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
         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 [97]: # average Word2Vec
         # compute average word2vec for each review.
         tfidf_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stored in th
         for sentence in tqdm(preprocessed_titles): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             tf_idf_weight =0; # num of words with a valid vector in the sentence/review
             for word in sentence.split(): # for each word in a review/sentence
                 if (word in glove_words) and (word in tfidf_words):
                     vec = model[word] # getting the vector for each word
                     # here we are multiplying idf value(dictionary[word]) and the tf value((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_titles.append(vector)

         print(len(tfidf_w2v_vectors_titles))
         print(len(tfidf_w2v_vectors_titles[0]))
```

```
100%|| 109248/109248 [00:08<00:00, 13401.25it/s]


109248
300
```

2.2.3 1.4.3 Vectorizing Numerical features

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

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

         price_scalar = StandardScaler()
```

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

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

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

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

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

```
In [100]: # 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)

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

          # Now standardize the data with above maen and variance.
          project_standardized = project_scalar.transform(project_data['teacher_number_of_previ
```

```
C:\Users\Kriti chaudhary\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConve

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

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

```
C:\Users\Kriti chaudhary\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConve

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

```
In [101]: project_standardized
```

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

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 [131]: print(categories_one_hot.shape)
          print(sub_categories_one_hot.shape)
          print(categories_one_hot_teacher_prefix.shape)
          print(categories_one_hot_grade.shape)
          print(categories_one_hot_num_dig.shape)
          print(categories_one_hot_state.shape)
          print(text_bow_essays.shape)
          print(text_bow_titles.shape)
          print(price_standardized.shape)
          print(project_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 9)
(109248, 9)
(109248, 2)
(109248, 9)
(109248, 16623)
(109248, 3329)
(109248, 1)
(109248, 1)
```

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

```
In [132]: from scipy.sparse import hstack
          X = hstack((sub_categories_one_hot,categories_one_hot,categories_one_hot_state,categ
          X.shape
```

```
Out[132]: (109248, 70)
```

```
In [133]: A = hstack((X,text_bow_titles))
          B = hstack((X,text_tfidf_titles))
          C = hstack((X,avg_w2v_vectors_titles))
          D = hstack((X,tfidf_w2v_vectors_titles))

In [134]: m = hstack((A,B,C,D))
          A.shape,B.shape,C.shape,D.shape, m.shape

Out[134]: ((109248, 3399), (109248, 3399), (109248, 370), (109248, 370), (109248, 7538))
```

Assignment 2: Apply TSNE

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

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

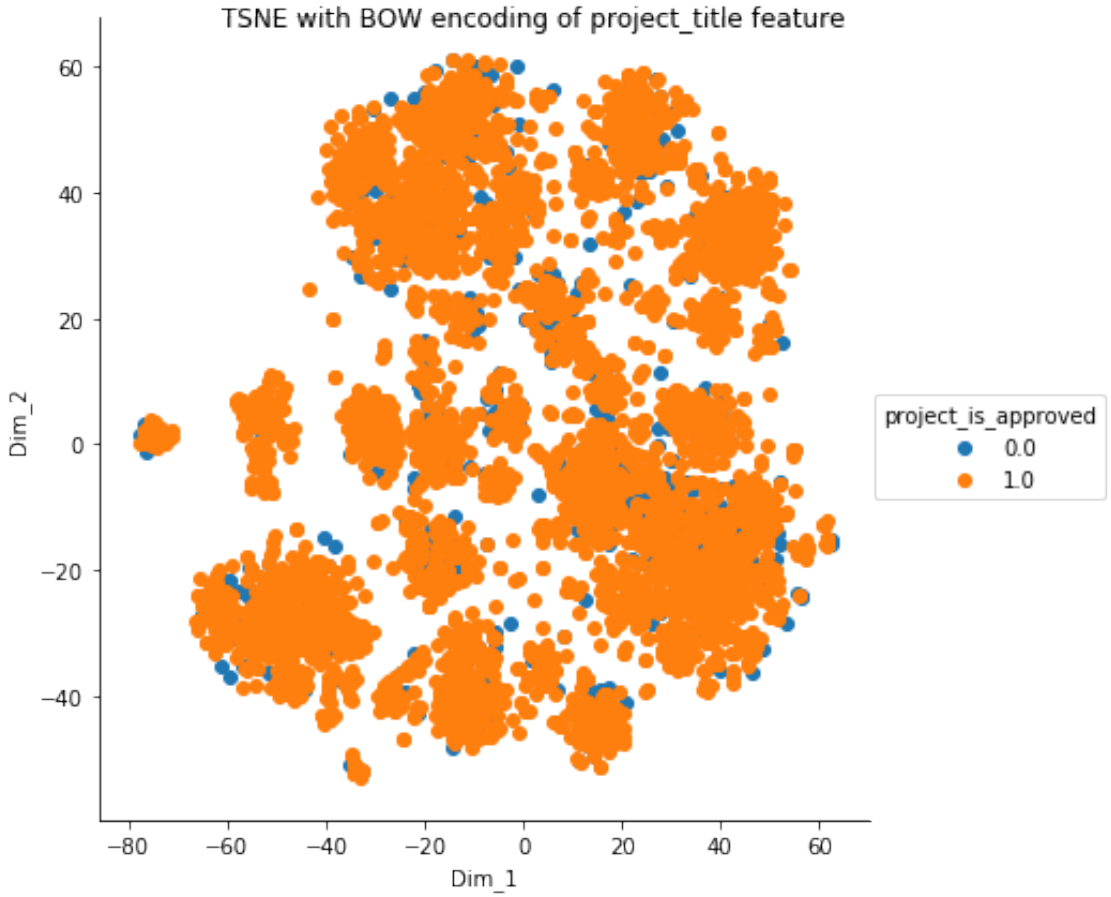
```
In [147]: def convert2matrix(data,no_of_points):
              k=csr_matrix(data)[:no_of_points]
              u = k.todense()
              u = u[:]
              return u
```

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

In [149]:
```
# please write all of the code with proper documentation and proper titles for each
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
from sklearn.manifold import TSNE
from scipy.sparse import csr_matrix
import time

time_start = time.time()
npoints = 5000
data_5000_1 = convert2matrix(A,npoints)

labels_5000_1 = project_data['project_is_approved'][:npoints]

model = TSNE(n_components=2,verbose=1, random_state=0,n_iter=2000)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(data_5000_1)
print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))


# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_5000_1)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "project_is_approve

# Ploting the result of tsne
g = sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_
g.fig.suptitle('TSNE with BOW encoding of project_title feature')
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 1.353s...
[t-SNE] Computed neighbors for 5000 samples in 205.160s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 0.838553
[t-SNE] KL divergence after 250 iterations with early exaggeration: 86.509628
[t-SNE] Error after 2000 iterations: 1.848597
```

```
t-SNE done! Time elapsed: 724.5056474208832 seconds
```



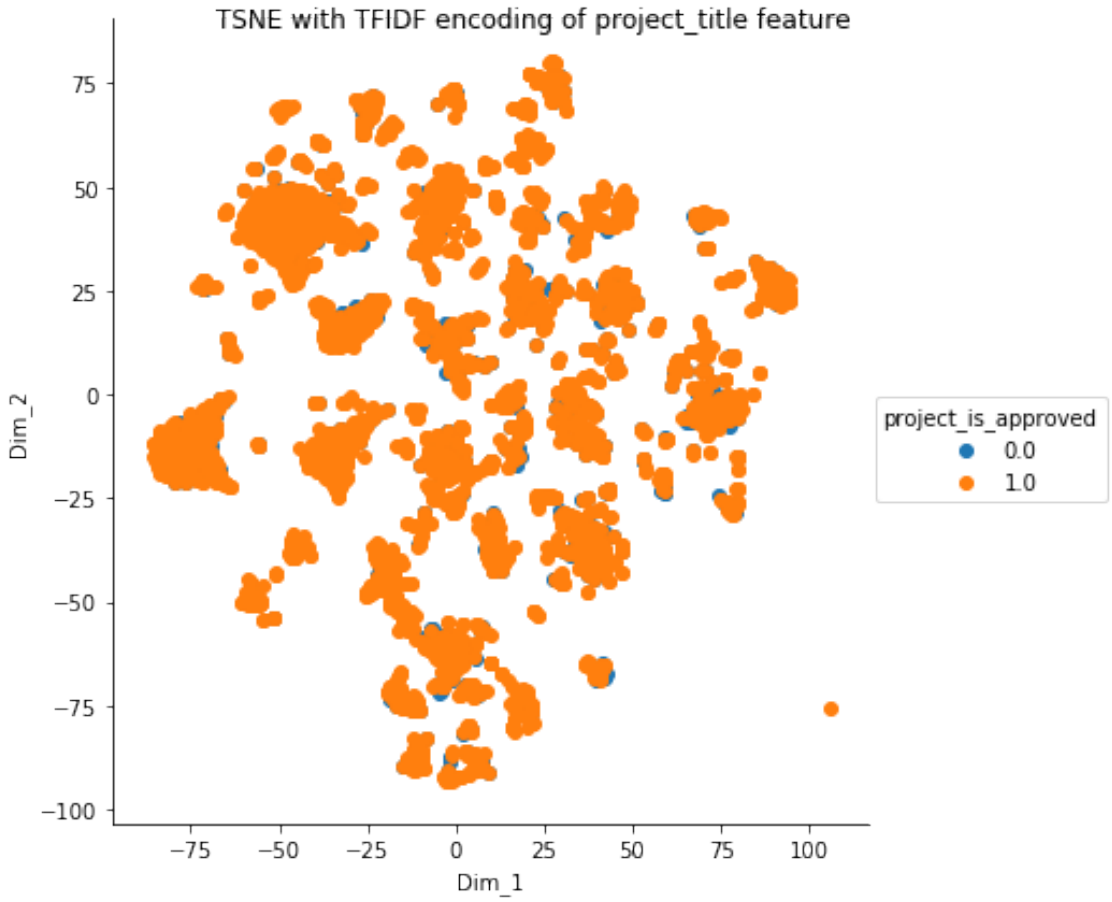
As you can see see there are clusters of points but blue points and orange points are overlapping each other so it's hard to interpret Since we are having less no. of points for class 0. But we are sure of one thing that its high-dimensional data contains sufficient variability because there are cluster of orange points So we can use these features for project classification

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

```
In [150]: # please write all the code with proper documentation, and proper titles for each su
          # when you plot any graph make sure you use
              # a. Title, that describes your plot, this will be very helpful to the reader
              # b. Legends if needed
              # c. X-axis label
              # d. Y-axis label

          from sklearn.manifold import TSNE
          from scipy.sparse import csr_matrix
          import time
```

```
time_start = time.time()
npoints = 5000
# Picking the top 1000 points as TSNE takes a lot of time for 15K points
data_5000_2 = convert2matrix(B,npoints)

labels_5000_2 = project_data['project_is_approved'][:npoints]

model = TSNE(n_components=2,verbose=1, random_state=0,n_iter=2000)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(data_5000_2)
print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))


# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_5000_2)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "project_is_approve

# Ploting the result of tsne
g = sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_
g.fig.suptitle('TSNE with TFIDF encoding of project_title feature')
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 1.236s...
[t-SNE] Computed neighbors for 5000 samples in 188.749s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 0.475031
[t-SNE] KL divergence after 250 iterations with early exaggeration: 74.491501
[t-SNE] Error after 2000 iterations: 0.952860
t-SNE done! Time elapsed: 626.5070950984955 seconds
```



Here we can see better clustering of orange points hence the data can be reduced to lower dimensions. Also these features would be best to work with for classifying the projects whether it get approved or rejected

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

```
In [151]: # please write all the code with proper documentation, and proper titles for each su
          # when you plot any graph make sure you use
              # a. Title, that describes your plot, this will be very helpful to the reader
              # b. Legends if needed
              # c. X-axis label
              # d. Y-axis label

          from sklearn.manifold import TSNE
          from scipy.sparse import csr_matrix
          import time

          time_start = time.time()
          npoints = 5000
          # Picking the top 1000 points as TSNE takes a lot of time for 15K points
          data_5000_3 = convert2matrix(C,npoints)
```

```
labels_5000_3 = project_data['project_is_approved'][:npoints]

model = TSNE(n_components=2,verbose=1, random_state=0,n_iter=2000)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(data_5000_3)
print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))


# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_5000_3)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "project_is_approve

# Ploting the result of tsne
g = sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_
g.fig.suptitle('TSNE with AVG W2V encoding of project_title feature')
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 0.171s...
[t-SNE] Computed neighbors for 5000 samples in 28.874s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 1.120775
[t-SNE] KL divergence after 250 iterations with early exaggeration: 86.365677
[t-SNE] Error after 2000 iterations: 2.318992
t-SNE done! Time elapsed: 580.4425520896912 seconds
```

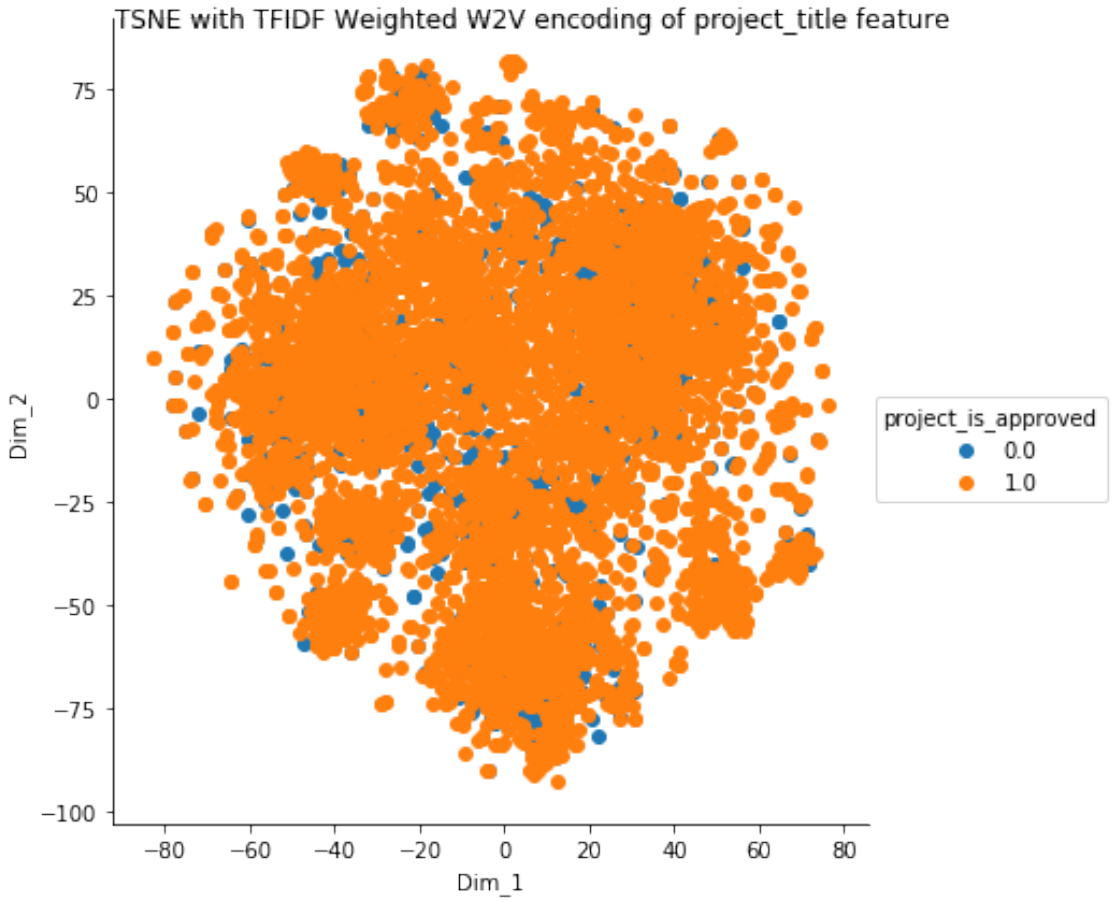
We got a T-Sne graph with lots of overlapping data so odds are high that our classifier will perform badly.

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

```
In [152]: # please write all the code with proper documentation, and proper titles for each su
          # when you plot any graph make sure you use
              # a. Title, that describes your plot, this will be very helpful to the reader
              # b. Legends if needed
              # c. X-axis label
              # d. Y-axis label

          from sklearn.manifold import TSNE
          from scipy.sparse import csr_matrix
          import time

          time_start = time.time()
          npoints = 5000
          # Picking the top 1000 points as TSNE takes a lot of time for 15K points
          data_5000_4 = convert2matrix(D,npoints)
```

```
        labels_5000_4 = project_data['project_is_approved'][:npoints]

        model = TSNE(n_components=2,verbose=1, random_state=0,n_iter=2000)
        # configuring the parameteres
        # the number of components = 2
        # default perplexity = 30
        # default learning rate = 200
        # default Maximum number of iterations for the optimization = 1000

        tsne_data = model.fit_transform(data_5000_4)
        print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))


        # creating a new data frame which help us in ploting the result data
        tsne_data = np.vstack((tsne_data.T, labels_5000_4)).T
        tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "project_is_approve

        # Ploting the result of tsne
        g = sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_
        g.fig.suptitle('TSNE with TFIDF Weighted W2V encoding of project_title feature')
        plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 0.186s...
[t-SNE] Computed neighbors for 5000 samples in 27.740s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 1.185279
[t-SNE] KL divergence after 250 iterations with early exaggeration: 86.269173
[t-SNE] Error after 2000 iterations: 2.388887
t-SNE done! Time elapsed: 587.793505191803 seconds
```



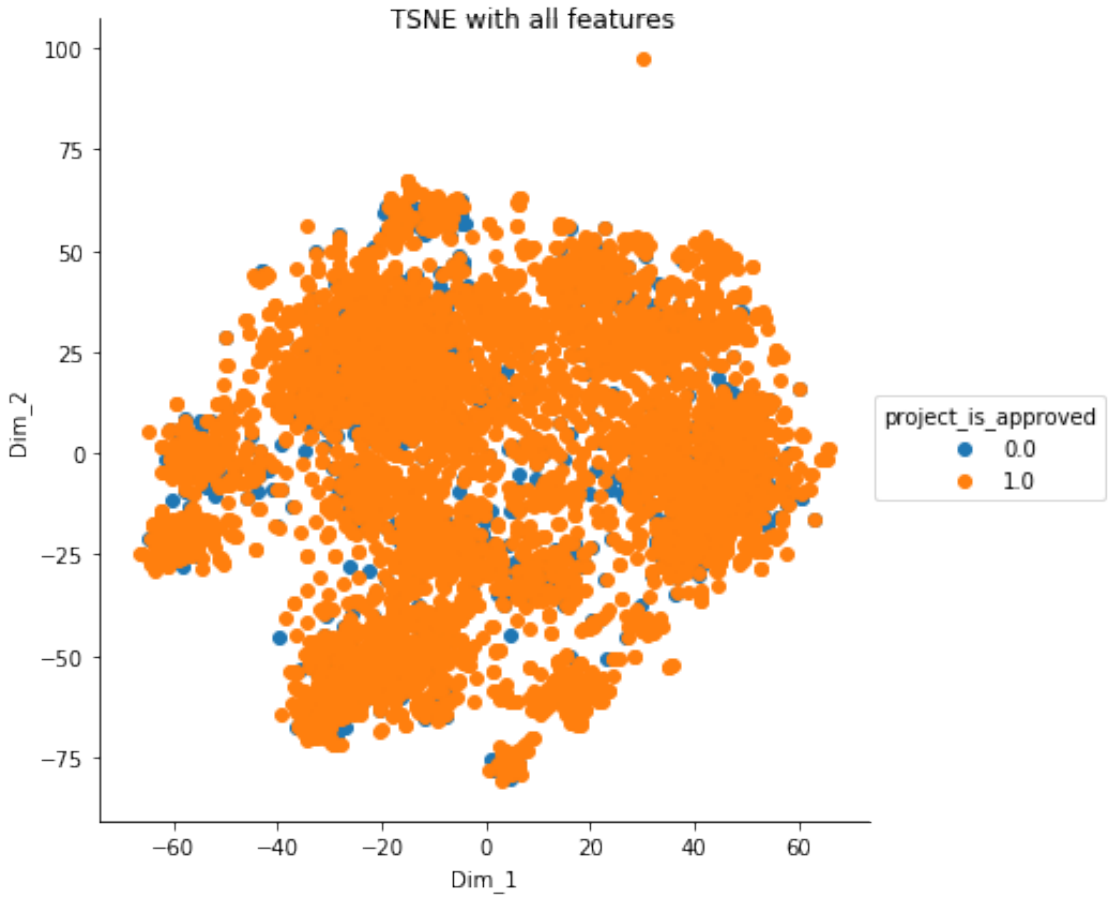
We got a T-Sne graph with lots of overlapping data so odds are high that our classifier will perform badly.

2.5 TSNE on the final data matrix

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

          time_start = time.time()
          npoints = 5000
          data_5000 = convert2matrix(m,npoints)

          labels_5000 = project_data['project_is_approved'][:npoints]
```

```
        model = TSNE(n_components=2,verbose=1, random_state=0,n_iter=1000)
        # configuring the parameteres
        # the number of components = 2
        # default perplexity = 30
        # default learning rate = 200
        # default Maximum number of iterations for the optimization = 1000

        tsne_data = model.fit_transform(data_5000)
        print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))


        # creating a new data frame which help us in ploting the result data
        tsne_data = np.vstack((tsne_data.T, labels_5000)).T
        tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "project_is_approve

        # Ploting the result of tsne
        g = sns.FacetGrid(tsne_df, hue="project_is_approved", size=6).map(plt.scatter, 'Dim_
        g.fig.suptitle('TSNE with all features')
        plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 4.317s...
[t-SNE] Computed neighbors for 5000 samples in 494.754s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 1.953713
[t-SNE] KL divergence after 250 iterations with early exaggeration: 86.496468
[t-SNE] Error after 1000 iterations: 2.182801
t-SNE done! Time elapsed: 1072.4603147506714 seconds
```



We got a T-Sne graph with lots of overlapping data so odds are high that our classifier will perform badly.

2.5 Summary

Project which were accepted: - mainly from math science and Literacy language category - subcategories as Literacy, and literacy-mathematics had the higher chances of approvance. - had no. of words in project title more than 6. - had no. of words in project essay, between 275 to 280. - had 30 to 90 no. of previously posted project. - had the numerical digits in most of the cases.

categorical, numerical features + project_title(TFIDF), under these features classifier will work better than rest of the features mentioned in other cases, by looking at Tsne plot of it we can say it.