Copy_of_2_DonorsChoose_EDA_TSNE(1)

March 6, 2019

1 DonorsChoose

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

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main p

How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly a
How to increase the consistency of project vetting across different volunteers to improve the experience for teache
How to focus volunteer time on the applications that need the most assistance

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 [3]: #importing all relevant packages

```
%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 [4]:
```
#since im using google_colab, i have to mount the gdrive folder for accessing the files

#from google.colab import drive
#drive.mount('/content/gdrive')
```

In [5]:
```
#reading the datasets, i have taken only 5000 datapoints into consideration for avoiding mermory issues

project_data = pd.read_csv('train_data.csv', nrows=5000)
resource_data = pd.read_csv('resources.csv')
```

In [6]:
```
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 (5000, 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 [7]:
```
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[7]:

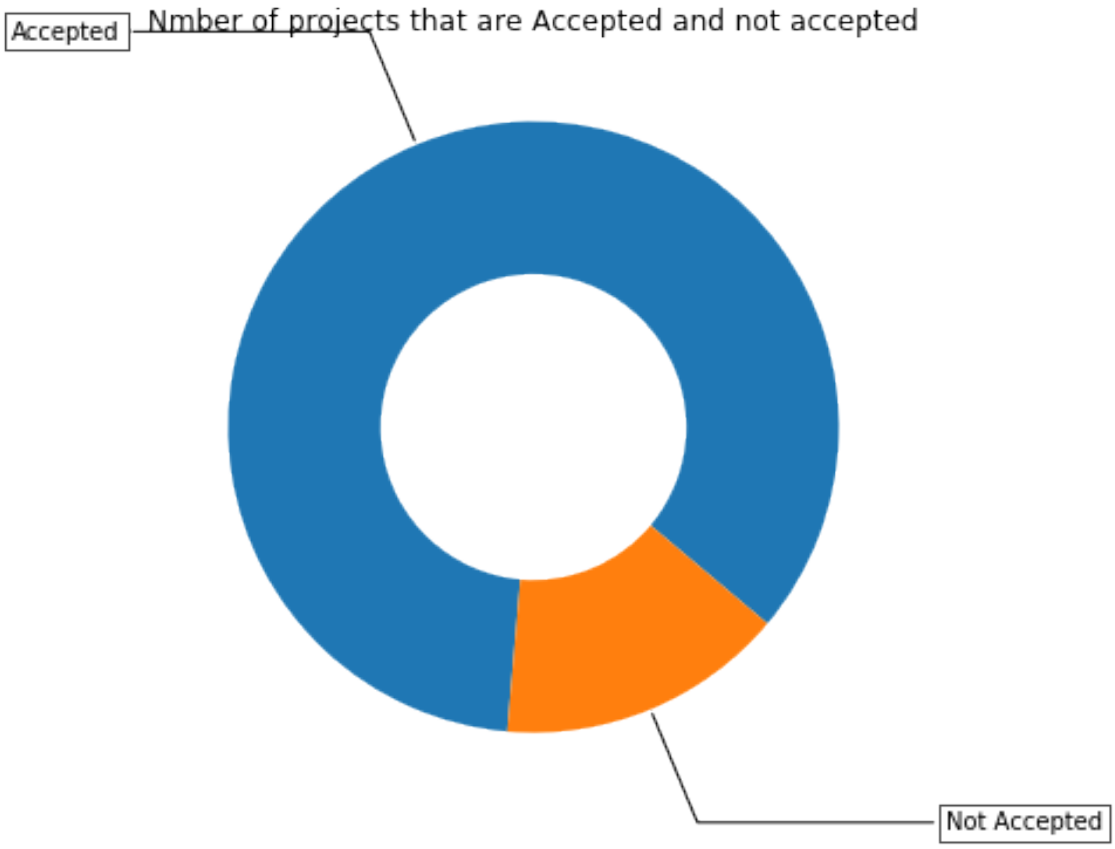
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

2 1.2 Data Analysis

In [8]:
```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-glr-gallery-pie


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

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 4237 , (84.74000000000001 %)
Number of projects thar are not approved for funding 763 , (15.260000000000002 %)



2.1 Observations

- the number of projects accepted for funding are in majority close to 85%

2.1.1 1.2.1 Univariate Analysis: School State

In [9]:
```
# Pandas dataframe groupby count, mean: https://stackoverflow.com/a/19385591/4084039

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

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

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

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

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

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

Out[9]:
```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n\nscl = [[0.0, \'rgb(24
```

In [10]:
```
# https://www.csi.cuny.edu/sites/default/files/pdf/administration/ops/2letterstabbrev.pdf
temp.sort_values(by=['num_proposals'], inplace=True)
print("States with lowest % approvals")
print(temp.head(5))
```

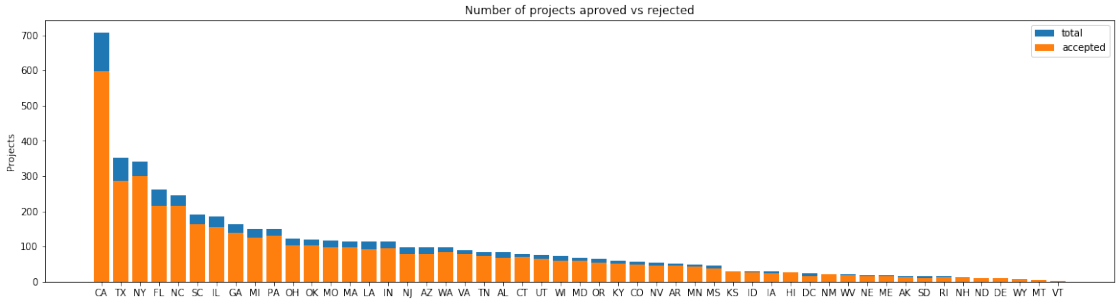
```
        print('='*50)
        print("States with highest % approvals")
        print(temp.tail(5))
```

```
States with lowest % approvals
   state_code  num_proposals
46         VT       0.500000
41         SD       0.687500
7          DC       0.695652
0          AK       0.705882
50         WY       0.777778
==================================================
States with highest % approvals
   state_code  num_proposals
11         HI       0.964286
16         KS       0.966667
28         ND       1.000000
8          DE       1.000000
30         NH       1.000000
```

```
In [11]: #stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.htm
        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 [12]: def univariate_barplots(data, col1, col2='project_is_approved', top=False):
            # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/4084039
            temp = pd.DataFrame(project_data.groupby(col1)[col2].agg(lambda x: x.eq(1).sum())).reset_index()

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

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

            if top:
                temp = temp[0:top]
```

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

In [13]: univariate_barplots(project_data, 'school_state', 'project_is_approved', False)



	school_state	project_is_approved	total	Avg
4	CA	597	707	0.844413
43	TX	286	352	0.812500
34	NY	299	342	0.874269
9	FL	215	261	0.823755
27	NC	216	246	0.878049

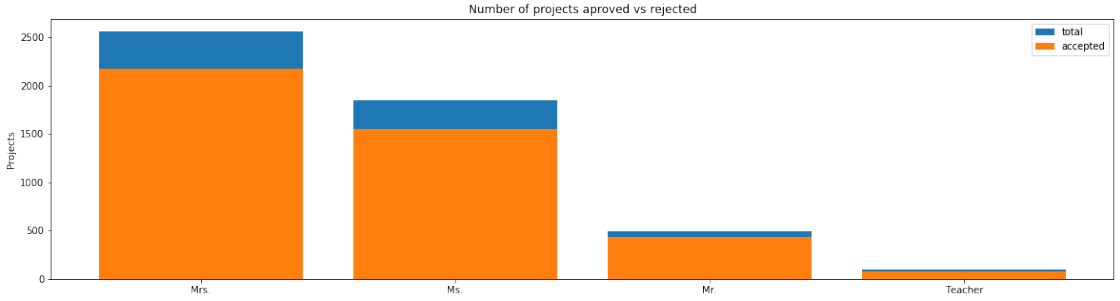
==========

	school_state	project_is_approved	total	Avg
28	ND	11	11	1.000000
8	DE	11	11	1.000000
50	WY	7	9	0.777778
26	MT	5	6	0.833333
46	VT	1	2	0.500000

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

2.1.2 1.2.2 Univariate Analysis: teacher_prefix

In [14]: univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)



	teacher_prefix	project_is_approved	total	Avg
1	Mrs.	2173	2560	0.848828
2	Ms.	1554	1845	0.842276
0	Mr.	433	495	0.874747
3	Teacher	77	100	0.770000

==========

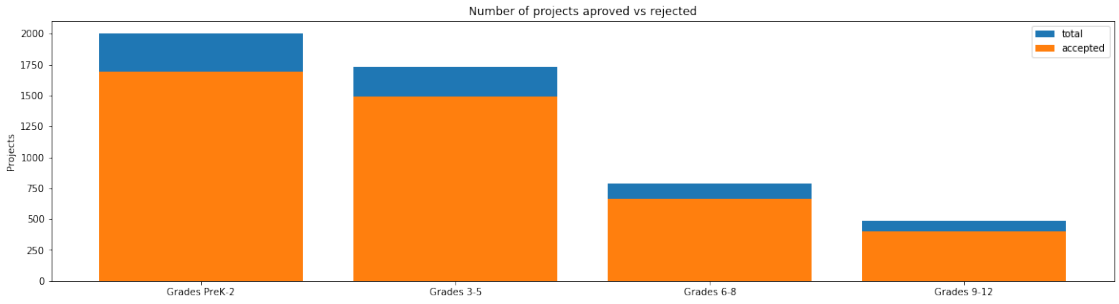
	teacher_prefix	project_is_approved	total	Avg
1	Mrs.	2173	2560	0.848828
2	Ms.	1554	1845	0.842276
0	Mr.	433	495	0.874747
3	Teacher	77	100	0.770000

2.2 Observations

- the teacher having prefix as "Mr" is having the highest project approval of 87%
- teachers having initials Mr/Ms/Mrs are having approval percent close to each other
- the teacher having prefix as "teacher" is having the lowest project approval of 77%

2.2.1 1.2.3 Univariate Analysis: project_grade_category

In [15]: univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', top=False)



	project_grade_category	project_is_approved	total	Avg
3	Grades PreK-2	1689	2002	0.843656
0	Grades 3-5	1491	1729	0.862348
1	Grades 6-8	660	785	0.840764
2	Grades 9-12	397	484	0.820248

==========

	project_grade_category	project_is_approved	total	Avg
3	Grades PreK-2	1689	2002	0.843656
0	Grades 3-5	1491	1729	0.862348
1	Grades 6-8	660	785	0.840764
2	Grades 9-12	397	484	0.820248

2.3 Observations

- lower grades are having more project proposals and approval rate is also high as in GRADE 3-5 of 86%
- higher grades i,e 9-12 having less project proposal and less approval rate as in GRADE 9-12 of 82%

2.3.1 1.2.4 Univariate Analysis: project_subject_categories

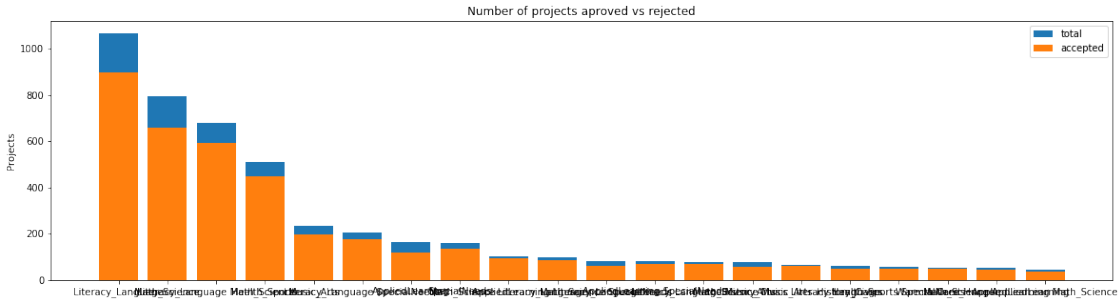
```
In [16]: catogories = list(project_data['project_subject_categories'].values)
         # remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

```
In [17]: project_data['clean_categories'] = cat_list
         project_data.drop(['project_subject_categories'], axis=1, inplace=True)
         project_data.head(2)
```

```
Out[17]:   Unnamed: 0       id                         teacher_id teacher_prefix  \
         0       160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc          Mrs.
         1       140945  p258326  897464ce9ddc600bced1151f324dd63a           Mr.

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

              project_subject_subcategories  \
         0                  ESL, Literacy
         1  Civics & Government, Team Sports

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

                                       project_essay_1  \
         0  My students are English learners that are work...
```

```
1  Our students arrive to our school eager to lea...

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

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

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0
1                                             7                    1

             clean_categories
0           Literacy_Language
1  History_Civics Health_Sports
```

In [18]: univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=20)



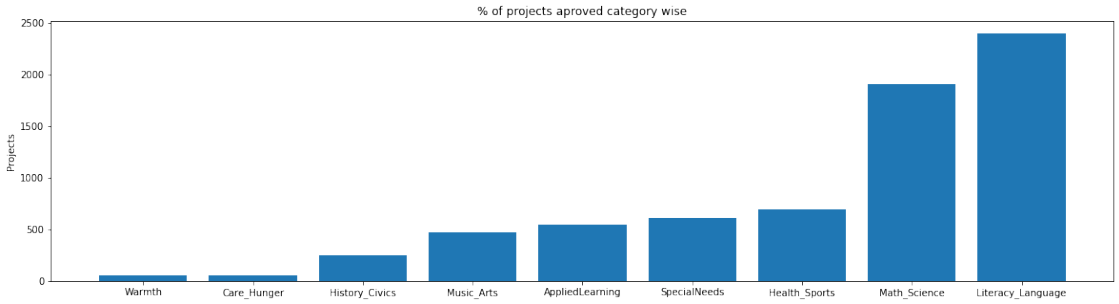

```
              clean_categories  project_is_approved  total       Avg
23              Literacy_Language                  900   1067  0.843486
30                   Math_Science                  659    795  0.828931
26  Literacy_Language Math_Science                  594    679  0.874816
8                 Health_Sports                  447    509  0.878193
37                    Music_Arts                  199    233  0.854077
==================================================================
             clean_categories  project_is_approved  total       Avg
16               History_Civics                   47     63  0.746032
14   Health_Sports SpecialNeeds                   49     57  0.859649
46          Warmth Care_Hunger                    47     53  0.886792
31  Math_Science AppliedLearning                  44     52  0.846154
4   AppliedLearning Math_Science                  35     44  0.795455
```

2.4 Observations

- highest number of project proposals are in literacy language category
- combination of literacy language and maths has highest approval of 87.48% and also health sports of 87%
- more focus is on health and sports and language as they are basic and important components

In [19]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

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()
```



2.5 Observations

- percentage of project approval is lowest in "Warmth" category

In [21]:
```
for i, j in sorted_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Warmth               :        58
Care_Hunger          :        58
History_Civics       :       252
```

```
Music_Arts           :      476
AppliedLearning      :      547
SpecialNeeds         :      614
Health_Sports        :      697
Math_Science         :     1910
Literacy_Language    :     2400
```

2.5.1 1.2.5 Univariate Analysis: project_subject_subcategories

In [22]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

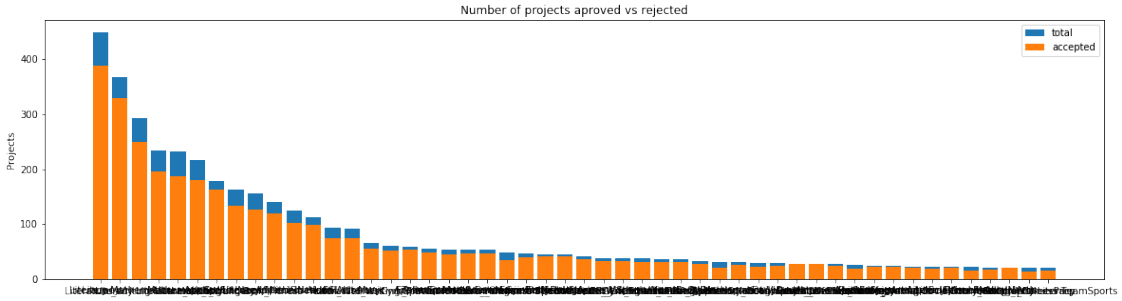
In [23]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[23]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_essay_1
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...
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...

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

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

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0
1                                             7                    1

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

In [24]: univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top=50)




```
            clean_subcategories  project_is_approved  total       Avg
189                    Literacy                  389    449  0.866370
191           Literacy Mathematics                329    368  0.894022
201  Literature_Writing Mathematics                250    293  0.853242
190     Literacy Literature_Writing                195    234  0.833333
209                    Mathematics                188    232  0.810345
==================================================================
          clean_subcategories  project_is_approved  total       Avg
23   AppliedSciences VisualArts                   15     22  0.681818
230          Other SpecialNeeds                   17     21  0.809524
181  History_Geography Literacy                   20     21  0.952381
56           College_CareerPrep                   14     20  0.700000
177  Health_Wellness TeamSports                   15     20  0.750000
```

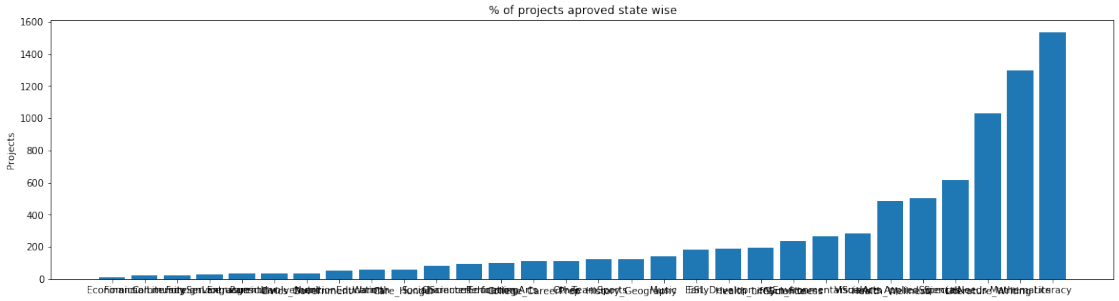
2.6 Observations

- project sub category " History_Geography Literacy" has the highest approval percent of 95%

- project sub category " AppliedSciences VisualArts has the highest approval percent of 68%

In [25]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

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



2.7 Observations

- project subcategory economics has lowest approval percentage.
- project subcategory lowest has highest approval percentage

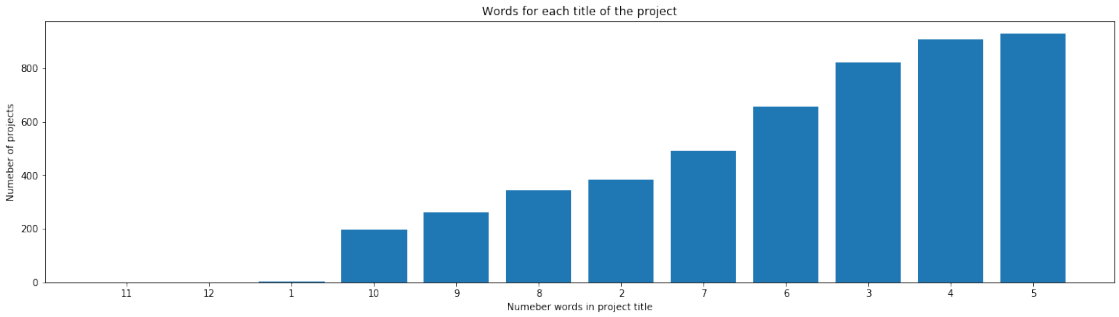
In [27]:
```
#number of projects under project subcategory

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

```
Economics            :        14
FinancialLiteracy    :        23
CommunityService     :        26
ForeignLanguages     :        29
```

```
Extracurricular      :      33
ParentInvolvement    :      34
Civics_Government    :      36
NutritionEducation   :      54
Warmth               :      58
Care_Hunger          :      58
SocialSciences       :      82
CharacterEducation   :      95
PerformingArts       :     102
College_CareerPrep   :     113
Other                :     114
TeamSports           :     123
History_Geography    :     124
Music                :     142
ESL                  :     182
EarlyDevelopment     :     189
Health_LifeScience   :     196
Gym_Fitness          :     237
EnvironmentalScience :     265
VisualArts           :     282
Health_Wellness      :     486
AppliedSciences      :     504
SpecialNeeds         :     614
Literature_Writing   :    1032
Mathematics          :    1295
Literacy             :    1534
```

2.7.1 1.2.6 Univariate Analysis: Text features (Title)

In [28]:
```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/408
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()
```

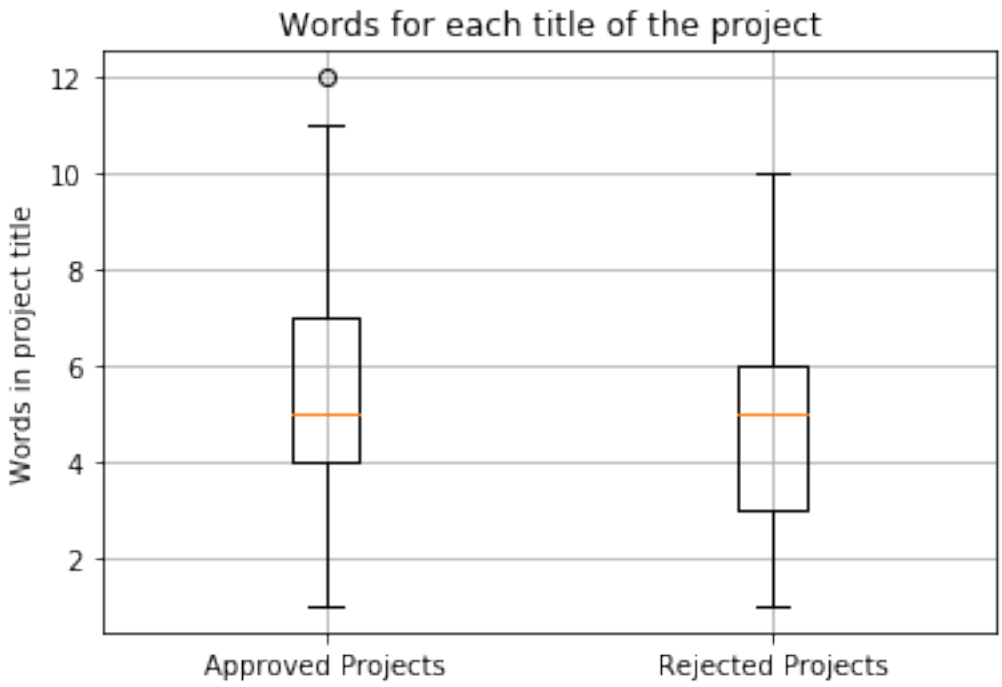


2.8 Observations

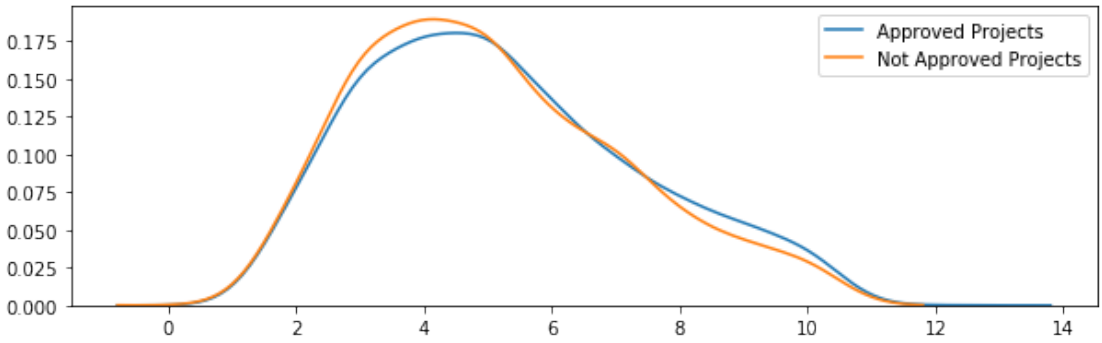
- more than 800 projects are having three or greater than three words in their project title
- only one project is having 10 words in its project title

In [29]:
```
approved_title_word_count = project_data[project_data['project_is_approved']==1]['project_title'].st
approved_title_word_count = approved_title_word_count.values

rejected_title_word_count = project_data[project_data['project_is_approved']==0]['project_title'].str.s
rejected_title_word_count = rejected_title_word_count.values
```

In [30]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.title('Words for each title of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



In [31]:
```
plt.figure(figsize=(10,3))
sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)
sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6)
plt.legend()
plt.show()
```



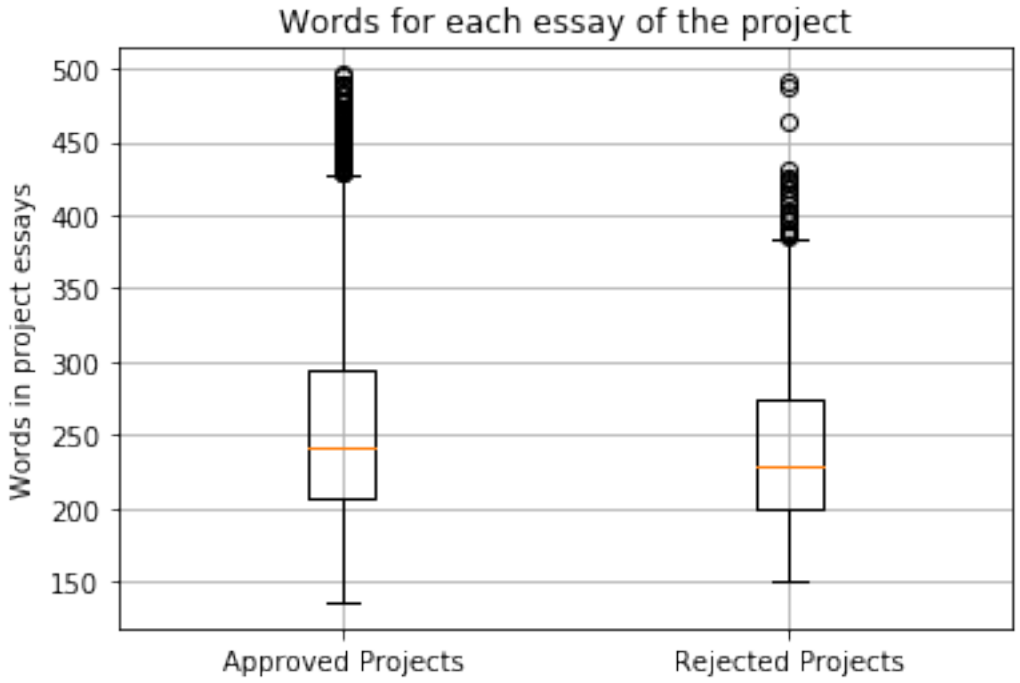
2.8.1 1.2.7 Univariate Analysis: Text features (Project Essay's)

In [32]:
```
# 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 [33]:
```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.split().apply
approved_word_count = approved_word_count.values

rejected_word_count = project_data[project_data['project_is_approved']==0]['essay'].str.split().apply(l
rejected_word_count = rejected_word_count.values
```

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

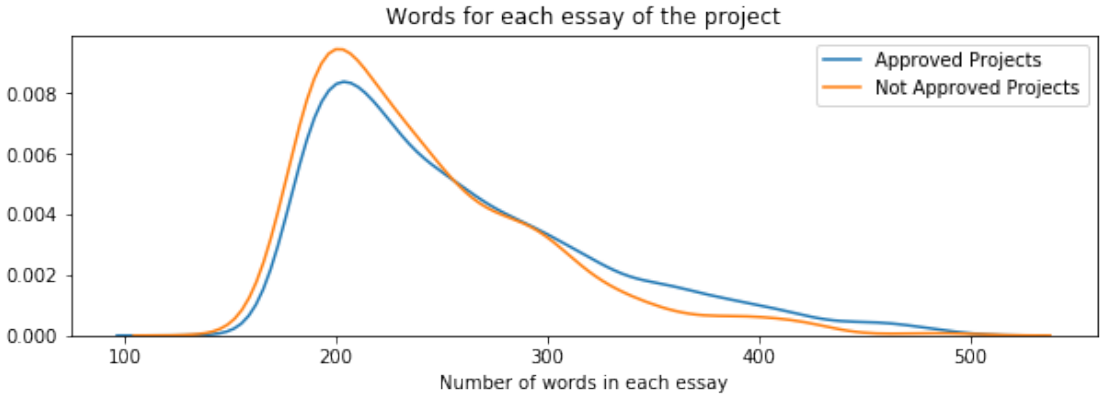


2.9 Observations

- 50th percentile of both approved project and rejected projects are close enough which means 50 percent of projects are having close to 250 words in their project essays in both the categories(approved and not approved)

In [35]:
```
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 essay')
plt.legend()
plt.show()
```



2.10 Observations

- majority of projects are having close to 200 words in their essays in both the categories approved as well as not approved
- majority of project essays are having words in the range of 150 to 350
- we cant find much differnce in number of words in both categories.

2.10.1 1.2.8 Univariate Analysis: Cost per project

In [36]:
```
# we get the cost of the project using resource.csv file
resource_data.head(3)
```

Out[36]:

	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	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45

In [37]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(3)
```

Out[37]:

	id	price	quantity
0	p000001	459.56	7

```
1  p000002  515.89        21
2  p000003  298.97         4
```

In [38]: price_data.dtypes

Out[38]:
```
id           object
price       float64
quantity      int64
dtype: object
```

In [39]: project_data.dtypes

Out[39]:
```
Unnamed: 0                                      int64
id                                             object
teacher_id                                     object
teacher_prefix                                 object
school_state                                   object
project_submitted_datetime                     object
project_grade_category                         object
project_title                                  object
project_essay_1                                object
project_essay_2                                object
project_essay_3                                object
project_essay_4                                object
project_resource_summary                       object
teacher_number_of_previously_posted_projects    int64
project_is_approved                             int64
clean_categories                               object
clean_subcategories                            object
essay                                          object
dtype: object
```

In [40]:
```
# join two dataframes in python:


project_data = pd.merge(project_data, price_data, on='id', how='left')
```

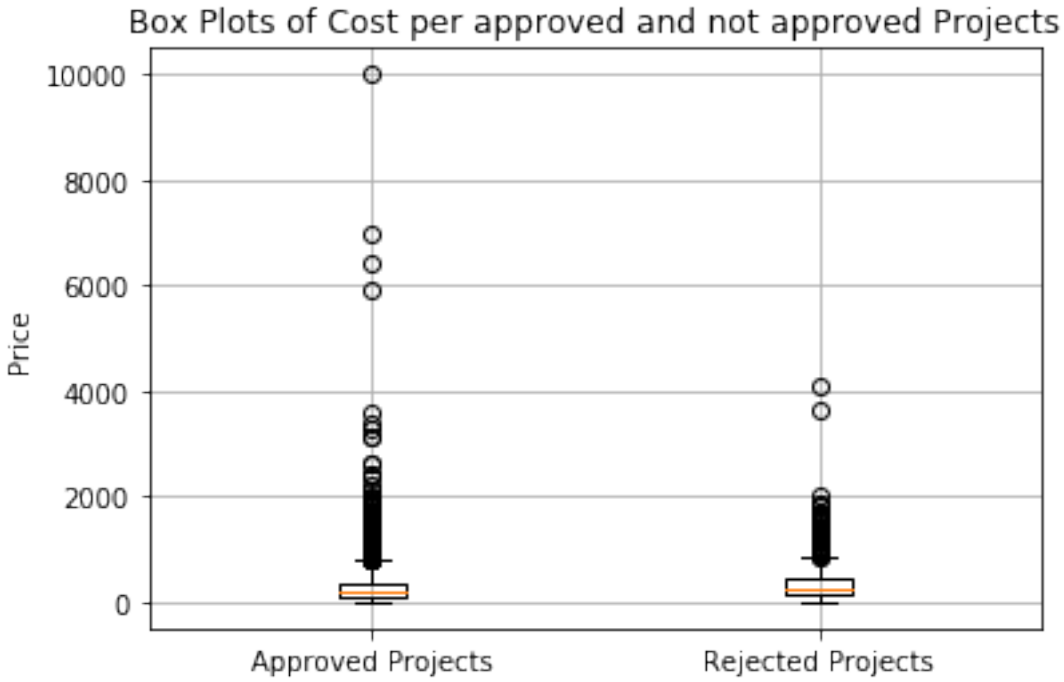
In [41]: project_data.head(1)

Out[41]:
```
   Unnamed: 0       id                         teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.

  school_state project_submitted_datetime project_grade_category  \
0           IN      2016-12-05 13:43:57          Grades PreK-2

                                project_title  \
0  Educational Support for English Learners at Home

                                project_essay_1  \
```

```
0  My students are English learners that are work...

                                project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...         NaN

  project_essay_4                       project_resource_summary  \
0             NaN  My students need opportunities to practice beg...

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0

    clean_categories clean_subcategories  \
0  Literacy_Language        ESL Literacy

                                              essay  price  quantity
0  My students are English learners that are work...  154.6        23
```

In [42]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].values

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

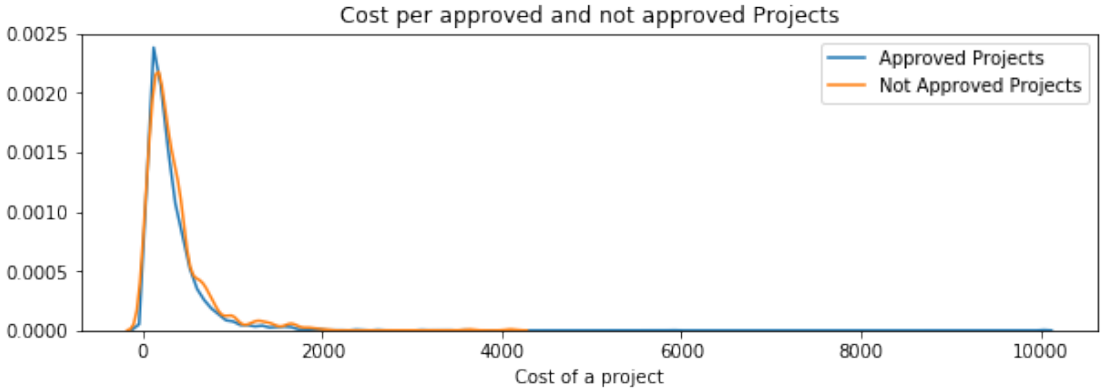
In [43]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Price')
plt.grid()
plt.show()
```



2.11 Observations

- we cant infer much from this plot, much detailed observations are made further

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



2.12 Observations

- Majority of projects are having cost less than close to 900 approx,

In [46]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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

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

```
+------------+------------------+----------------------+
| Percentile | Approved Projects | Not Approved Projects |
```

0	1.44	5.19
5	14.664	40.045
10	35.41	75.106
15	56.788	104.181
20	75.848	126.288
25	100.21	145.665
30	119.948	159.996
35	139.99	180.088
40	159.43	207.546
45	179.0	232.279
50	200.77	258.07
55	229.636	289.256
60	259.744	314.946
65	288.936	357.846
70	326.598	393.798
75	376.51	428.41
80	423.724	483.844
85	495.786	604.884
90	602.35	715.232
95	820.454	1008.799
100	9999.0	4102.47

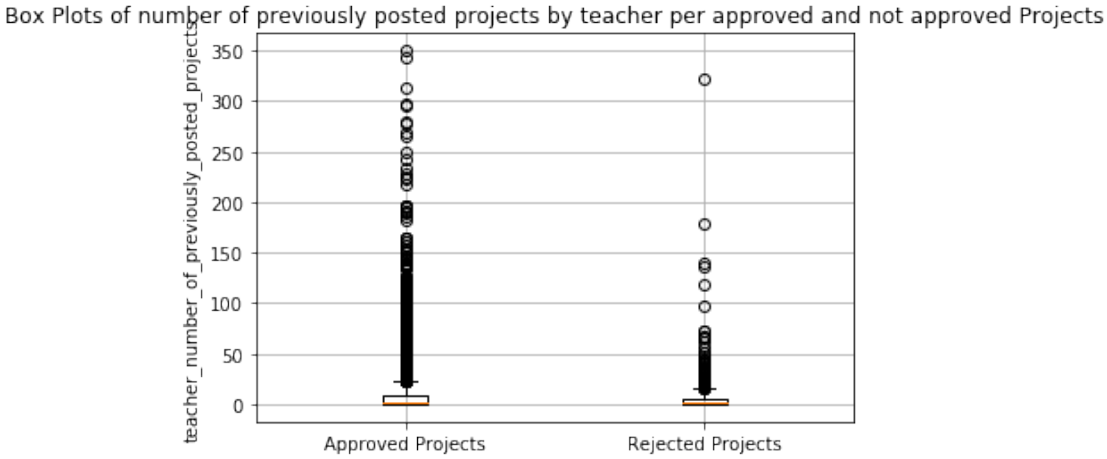
2.13 Observations

- 50 percent of projects are having cost less than or equal to 200.77
- 85 percent of projects are having cost less than or equal to 495.785

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [47]:
```
approved_tnppp = project_data[project_data['project_is_approved']==1]['teacher_number_of_previo

rejected_tnppp = project_data[project_data['project_is_approved']==0]['teacher_number_of_previousl
```

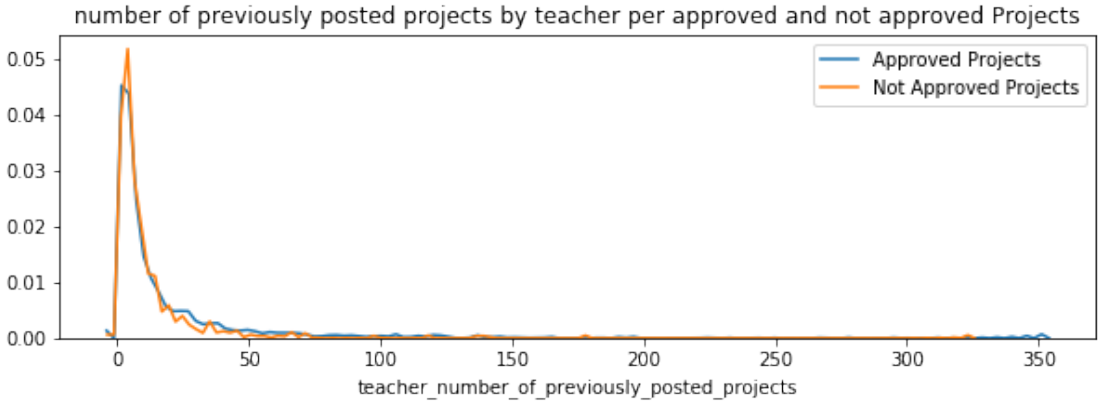
In [48]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_tnppp, rejected_tnppp])
plt.title('Box Plots of number of previously posted projects by teacher per approved and not approved Pro
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('teacher_number_of_previously_posted_projects')
plt.grid()
plt.show()
```



2.14 Observations

- Cannot infer much from this plot

In [49]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_tnppp, hist=False, label="Approved Projects")
sns.distplot(rejected_tnppp, hist=False, label="Not Approved Projects")
plt.title('number of previously posted projects by teacher per approved and not approved Projects')
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.legend()
plt.show()
```



2.15 Observations

- Majority of teachers have posted projects less than 19 previously close to 85%

In [50]:
```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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

for i in range(0,101,5):
    x.add_row([i,np.round(np.percentile(approved_tnppp,i), 3), np.round(np.percentile(rejected_tnppp,i),
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        |         4.0          |
|     70     |       6.0        |         5.0          |
|     75     |       9.0        |         6.0          |
|     80     |       13.0       |         8.0          |
|     85     |       19.0       |         11.0         |
|     90     |       29.0       |         15.0         |
|     95     |       55.0       |         26.9         |
|    100     |      350.0       |         322.0        |
+------------+------------------+----------------------+
```

2.16 Observations

- 85% of teachers have posted less than or equal to 19 projects previously which are approved and less than or equal to 11 projects previously which are not approved.

1.2.10 Univariate Analysis: project_resource_summary

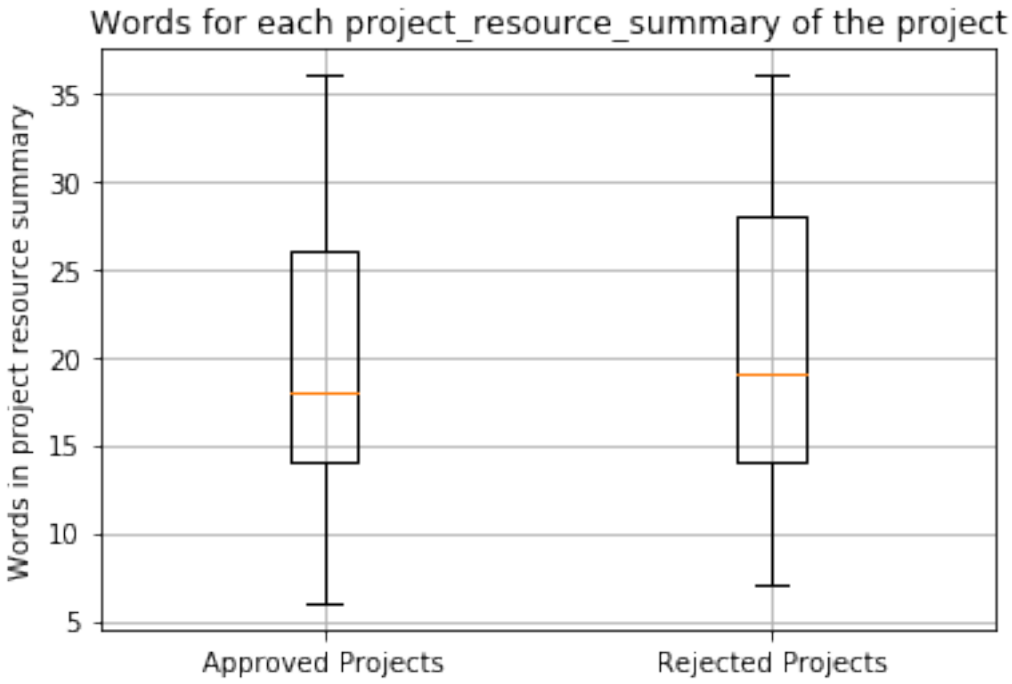
Please do this on your own based on the data analysis that was done in the above cells

Check if the presence of the numerical digits in the project_resource_summary effects the acceptance of the project or not. If you observe that presence of the numerical digits is helpful in the classification, please include it for further process or you can ignore it.

In [51]:
```
approved_PRS_word_count = project_data[project_data['project_is_approved']==1]['project_resourc
approved_PRS_word_count = approved_PRS_word_count.values
```

In [52]:
```
rejected_PRS_word_count = project_data[project_data['project_is_approved']==0]['project_resource
rejected_PRS_word_count = rejected_PRS_word_count.values
```

In [53]:
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_PRS_word_count, rejected_PRS_word_count])
plt.title('Words for each project_resource_summary of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project resource summary')
plt.grid()
plt.show()
```

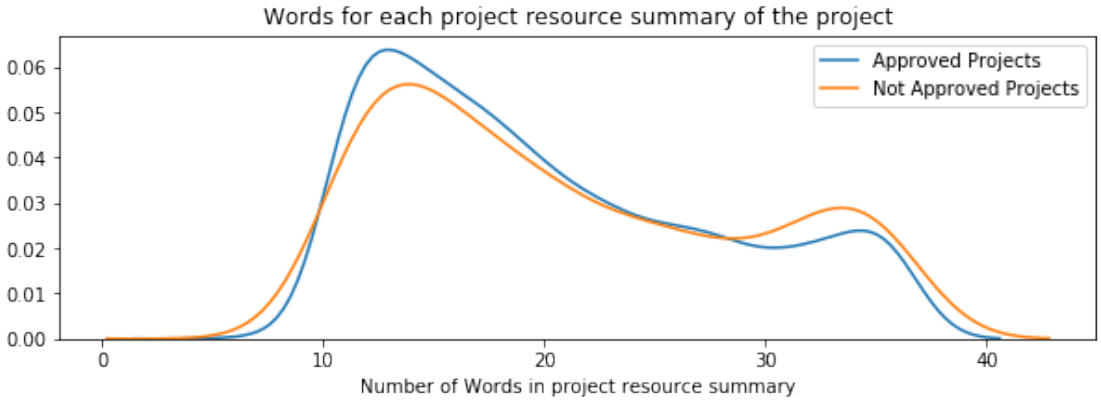


2.17 Observations

- 50th percentile for both the categories are located close enough, there is not much difference in number of words in project resource summary which are approved as well as not approved.

In [54]:
```
plt.figure(figsize=(10,3))
sns.distplot(approved_PRS_word_count, hist=False, label="Approved Projects")
sns.distplot(rejected_PRS_word_count, hist=False, label="Not Approved Projects")
plt.title('Words for each project resource summary of the project')
plt.xlabel('Number of Words in project resource summary')
```

```
plt.legend()
plt.show()
```



2.18 Observations

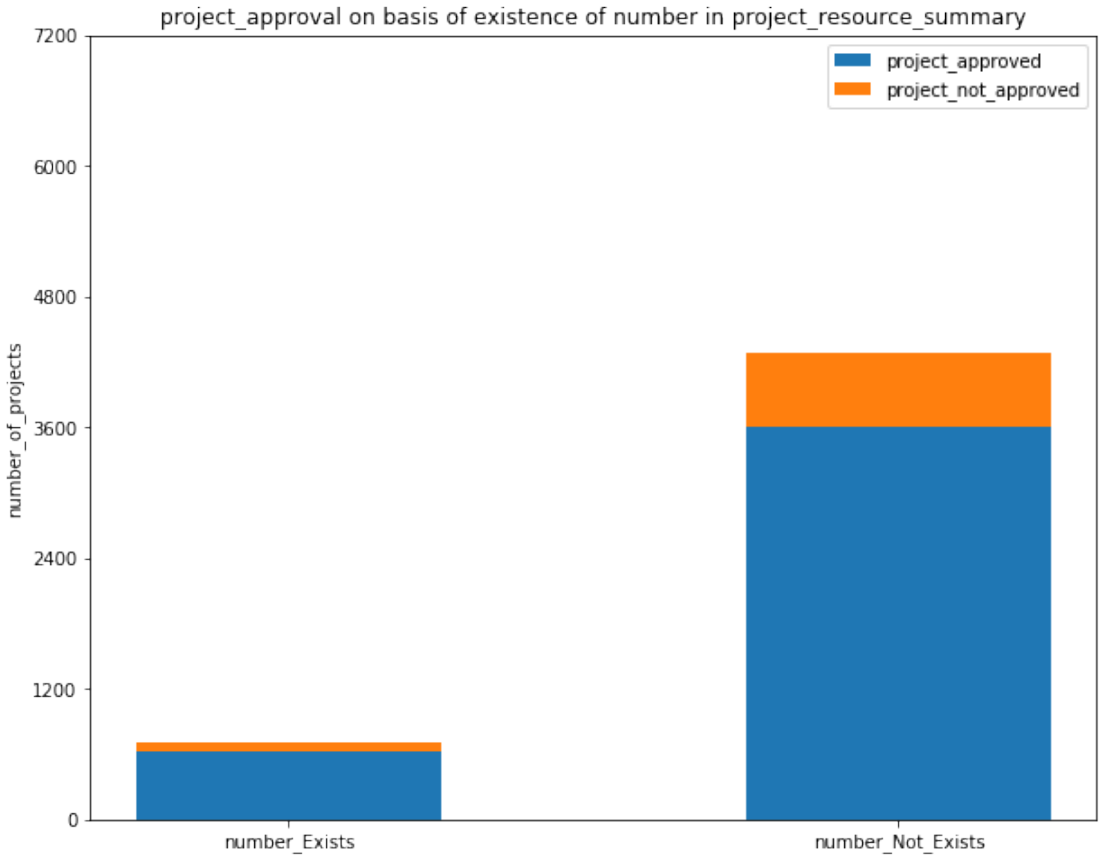
- major number of projects having project resource summary are having number of words in the range 10 to 30 in both the categories.
- there is not much differnce in the plots of both categories.

```
In [55]: catogories = list(project_data['project_resource_summary'].values)
         # remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

In [56]: new_project_data = project_data.copy()
         new_project_data['project_resource_summary_clean'] = cat_list
         new_project_data.drop(['project_resource_summary'], axis=1, inplace=True)

In [57]: prsA = new_project_data[new_project_data['project_is_approved']==1]['project_resource_summary_
         prsNA = new_project_data[new_project_data['project_is_approved']==0]['project_resource_summary_
```

In [58]:
```
prsAlist = list(prsA.values) #list of strings in proj res summary which are approved
prsNAlist = list(prsNA.values)  #list of strings in proj res summary which are not approved
```

In [59]:
```
##https://stackoverflow.com/questions/6649096/detect-numbers-in-string

#whether string contains a number or not ?

Aindex = []
for i in range(len(prsAlist)) :
    if(any(c.isdigit() for c in prsAlist[i])) :
     Aindex.append(i)




NAindex = []
for i in range(len(prsNAlist)) :
    if(any(c.isdigit() for c in prsNAlist[i])) :
     NAindex.append(i)
```

In [60]:
```
# https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html

import numpy as np
import matplotlib.pyplot as plt


N = 2
no_proj_approved = (len(Aindex), len(prsAlist) - len(Aindex))
no_proj_not_approved = (len(NAindex), len(prsNAlist) - len(NAindex))

ind = np.arange(N)    # the x locations for the groups
plt.figure(figsize=(10,8))
width = 0.5       # the width of the bars: can also be len(x) sequence

p1 = plt.bar(ind, no_proj_approved, width)
p2 = plt.bar(ind, no_proj_not_approved, width, bottom=no_proj_approved)

plt.ylabel('number_of_projects')
plt.title('project_approval on basis of existence of number in project_resource_summary')
plt.xticks(ind, ('number_Exists', 'number_Not_Exists'))
plt.yticks(np.arange(0, 7201, 1200))
plt.legend((p1[0], p2[0]), ('project_approved', 'project_not_approved'))



plt.show()
```



2.19 Observations

- as seen from the plot, even if a number exists in project resource summary, it is not contributing to the approval of project for funding

2.20 1.3 Text preprocessing

2.20.1 1.3.1 Essay Text

In [61]: project_data.head(2)

Out[61]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educational Support for English Learners at Home
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	

```
1              Wanted: Projector for Hungry Learners

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

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

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

   teacher_number_of_previously_posted_projects  project_is_approved  \
0                                             0                    0
1                                             7                    1

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

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

In [62]:
```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
#print(project_data['essay'].values[20000])
#print("="*50)
#print(project_data['essay'].values[99999])
#print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. We are a meltin
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the t
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a
==================================================
```

1.3.2 Project title Text

In [63]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
```

```
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "
            'won', "won't", 'wouldn', "wouldn't"]
```

In [64]:
```
# similarly you can preprocess the titles also

# 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 [65]:
```
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_project_title.append(sent.lower().strip())
```

100%|| 5000/5000 [00:00<00:00, 26958.11it/s]

2.21 1. 4 Preparing data for models

In [66]: project_data.columns

Out[66]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
'project_submitted_datetime', 'project_grade_category', 'project_title',
'project_essay_1', 'project_essay_2', 'project_essay_3',
'project_essay_4', 'project_resource_summary',
'teacher_number_of_previously_posted_projects', 'project_is_approved',
'clean_categories', 'clean_subcategories', 'essay', 'price',
'quantity'],
dtype='object')

we are going to consider

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

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

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

2.21.1 1.4.1 Vectorizing Categorical data

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

1.4.1.1 clean_categories

In [67]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())


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

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', '
Shape of matrix after one hot encodig (5000, 9)

1.4.1.2 clean_subcategories

In [68]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=Tru
vectorizer.fit(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())


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

['Economics', 'FinancialLiteracy', 'CommunityService', 'ForeignLanguages', 'Extracurricular', 'ParentInvolvement'
Shape of matrix after one hot encodig (5000, 30)

1.4.1.3 school_state

In [69]:
```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category also

# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=Tru
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


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

['Economics', 'FinancialLiteracy', 'CommunityService', 'ForeignLanguages', 'Extracurricular', 'ParentInvolvement'
Shape of matrix after one hot encodig (5000, 30)

1.4.1.4 teacher_prefix

In [70]:
```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category also

# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=Tru
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())


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

['Economics', 'FinancialLiteracy', 'CommunityService', 'ForeignLanguages', 'Extracurricular', 'ParentInvolvement'
Shape of matrix after one hot encodig (5000, 30)

2.21.2 1.4.2 Vectorizing Text data

1.4.2.1 Bag of words 1.4.2.2 Bag of Words on project_title

In [71]:
```
# 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 projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_project_title)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

Shape of matrix after one hot encodig (5000, 382)

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

1.4.2.2 TFIDF Vectorizer on project_title

In [73]:
```
# Similarly you can vectorize for title also

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

Shape of matrix after one hot encodig (5000, 382)

1.4.2.3 Using Pretrained Models: AVG W2V on project_title

In [74]:
```
# Similarly you can vectorize for title also

# Train your own Word2Vec model using your own text corpus
i=0
list_of_sentance=[]
for sentance in preprocessed_project_title:
    list_of_sentance.append(sentance.split())
```

In [75]:
```
# Using Google News Word2Vectors

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file wich contains a dict ,
```

```
# and it contains all our courpus words as keys and  model[word] as values
# To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.


# http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17SRFAzZPY
# you can comment this whole cell
# or change these varible according to your need

is_your_ram_gt_16g=False
want_to_use_google_w2v = False
want_to_train_w2v = True

if want_to_train_w2v:
    # min_count = 5 considers only words that occured atleast 5 times
    w2v_model=Word2Vec(list_of_sentance,min_count=5,size=50, workers=4)
    print(w2v_model.wv.most_similar('great'))
    print('='*50)
    #print(w2v_model.wv.most_similar('worst'))

elif want_to_use_google_w2v and is_your_ram_gt_16g:
    if os.path.isfile('GoogleNews-vectors-negative300.bin'):
        w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=T
        print(w2v_model.wv.most_similar('great'))
        print(w2v_model.wv.most_similar('worst'))
    else:
        print("you don't have gogole's word2vec file, keep want_to_train_w2v = True, to train your own w
```

```
[('new', 0.9980190992355347), ('time', 0.997730016708374), ('special', 0.9976849555969238), ('readers', 0.99768364
==================================================
```

In [76]:
```
w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words))
print("sample words ", w2v_words[0:50])
```

```
number of words that occured minimum 5 times  732
sample words  ['educational', 'support', 'english', 'learners', 'home', 'wanted', 'hungry', 'soccer', 'equipment', 'awes
```

In [77]:
```
# average Word2Vec
# compute average word2vec for each review.
sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sent in tqdm(list_of_sentance): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, you might need to change this to 300
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
```

```
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
        if cnt_words != 0:
            sent_vec /= cnt_words
        sent_vectors.append(sent_vec)
    print(len(sent_vectors))
    print(len(sent_vectors[0]))
```

```
100%|| 5000/5000 [00:00<00:00, 11883.48it/s]

5000
50
```

1.4.2.4 Using Pretrained Models: TFIDF weighted W2V on project_title

```
In [78]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
    model = TfidfVectorizer()
    model.fit(preprocessed_project_title)
    # we are converting a dictionary with word as a key, and the idf as a value
    dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))
```

```
In [79]: # TF-IDF weighted Word2Vec
    tfidf_feat = model.get_feature_names() # tfidf words/col-names
    # final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = tfidf

    tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored in this list
    row=0;
    for sent in tqdm(list_of_sentance): # for each review/sentence
        sent_vec = np.zeros(50) # as word vectors are of zero length
        weight_sum =0; # num of words with a valid vector in the sentence/review
        for word in sent: # for each word in a review/sentence
            if word in w2v_words and word in tfidf_feat:
                vec = w2v_model.wv[word]
    #             tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
                # to reduce the computation we are
                # dictionary[word] = idf value of word in whole courpus
                # sent.count(word) = tf valeus of word in this review
                tf_idf = dictionary[word]*(sent.count(word)/len(sent))
                sent_vec += (vec * tf_idf)
                weight_sum += tf_idf
        if weight_sum != 0:
            sent_vec /= weight_sum
        tfidf_sent_vectors.append(sent_vec)
        row += 1
```

100%|| 5000/5000 [00:01<00:00, 3744.50it/s]

2.21.3 1.4.3 Vectorizing Numerical features

Price

In [80]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Standa
from sklearn.preprocessing import StandardScaler

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

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

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

Mean : 302.78363, Standard deviation : 376.3799456048676

TNPPP

In [81]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Standa
from sklearn.preprocessing import StandardScaler

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

tnppp_scalar = StandardScaler()
tnppp_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)) #
print(f"Mean : {tnppp_scalar.mean_[0]}, Standard deviation : {np.sqrt(tnppp_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
tnppp_standardized = tnppp_scalar.transform(project_data['teacher_number_of_previously_posted_pr
```

Mean : 10.7566, Standard deviation : 26.79330058876659

/home/pritam_sk/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:475: DataConversionWarni

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

/home/pritam_sk/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:475: DataConversionWarni

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

- Mapping class_labels 1:yes, 0:no

In [82]:
```
project_data['project_is_approved'] = project_data['project_is_approved'].map({1:"yes", 0:"no"})
project_data['project_is_approved'] = project_data['project_is_approved'].astype('category')
class_labels = project_data['project_is_approved'].values
```

Assignment 2: Apply TSNE

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

 In the above cells we have plotted and analyzed many features. Please observe the plots and write the observatio
 EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects

Build the data matrix using these features
school_state : categorical data (one hot encoding)
clean_categories : categorical data (one hot encoding)
clean_subcategories : categorical data (one hot encoding)
teacher_prefix : categorical data (one hot encoding)
project_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)
price : numerical
teacher_number_of_previously_posted_projects : numerical

 Now, plot FOUR t-SNE plots with each of these feature sets.

categorical, numerical features + project_title(BOW)
categorical, numerical features + project_title(TFIDF)
categorical, numerical features + project_title(AVG W2V)
categorical, numerical features + project_title(TFIDF W2V)

 Concatenate all the features and Apply TNSE on the final data matrix
 Note 1: The TSNE accepts only dense matrices
 Note 2: Consider only 5k to 6k data points to avoid memory issues. If you run into memory e
poins you are using

3 Plotting t-SNE on features

3.1 2.1 TSNE with BOW encoding of project_title feature

In [83]:
```
# please write all of the code with proper documentation and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label


from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Xbow = hstack((categories_one_hot, price_standardized, text_bow))
XbowS = Xbow.toarray()
XbowS.shape
```

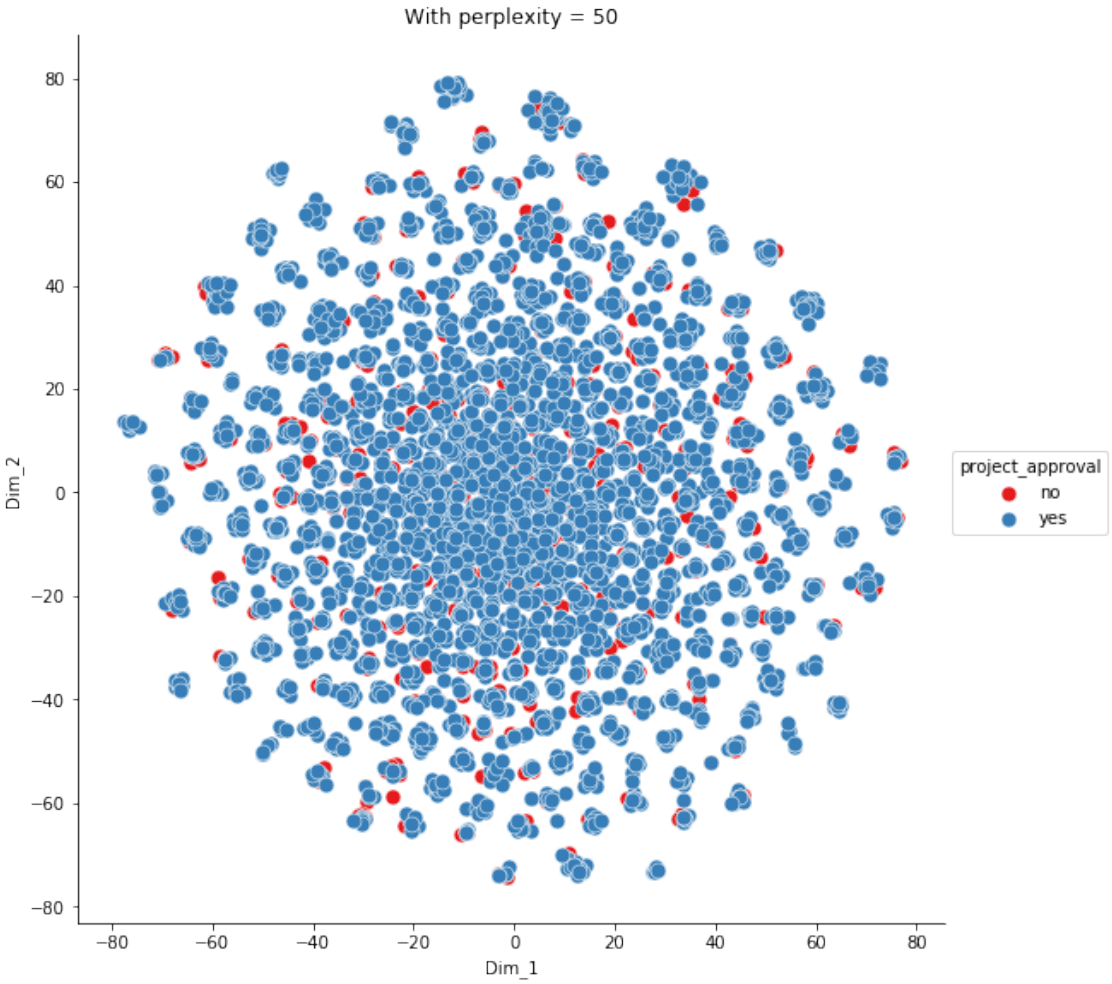
Out[83]: (5000, 392)

3.1.1 tSNE (BoW)

In [84]:
```
# Data-preprocessing: Standardizing the data

from sklearn.preprocessing import StandardScaler
standardized_data_bow = StandardScaler().fit_transform(XbowS)
print(standardized_data_bow.shape)
```

(5000, 392)

In [85]:
```
from sklearn.manifold import TSNE
import seaborn as sn


model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(standardized_data_bow)

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

# Ploting the result of tsne
kws = dict(s=80, linewidth=.5, edgecolor="w")
sn.FacetGrid(tsne_df, hue="project_approval", size=8, palette="Set1").map(plt.scatter, 'Dim_1', 'Dim_
plt.title('With perplexity = 50')
plt.show()
```



3.2 2.2 TSNE with TFIDF encoding of project_title feature

In [86]: # please write all the code with proper documentation, and proper titles for each subsection
when you plot any graph make sure you use
a. Title, that describes your plot, this will be very helpful to the reader
b. Legends if needed
c. X-axis label
d. Y-axis label

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Xtfidf = hstack((sub_categories_one_hot, price_standardized, text_tfidf))
XtfidfS = Xtfidf.toarray()
XtfidfS.shape
```

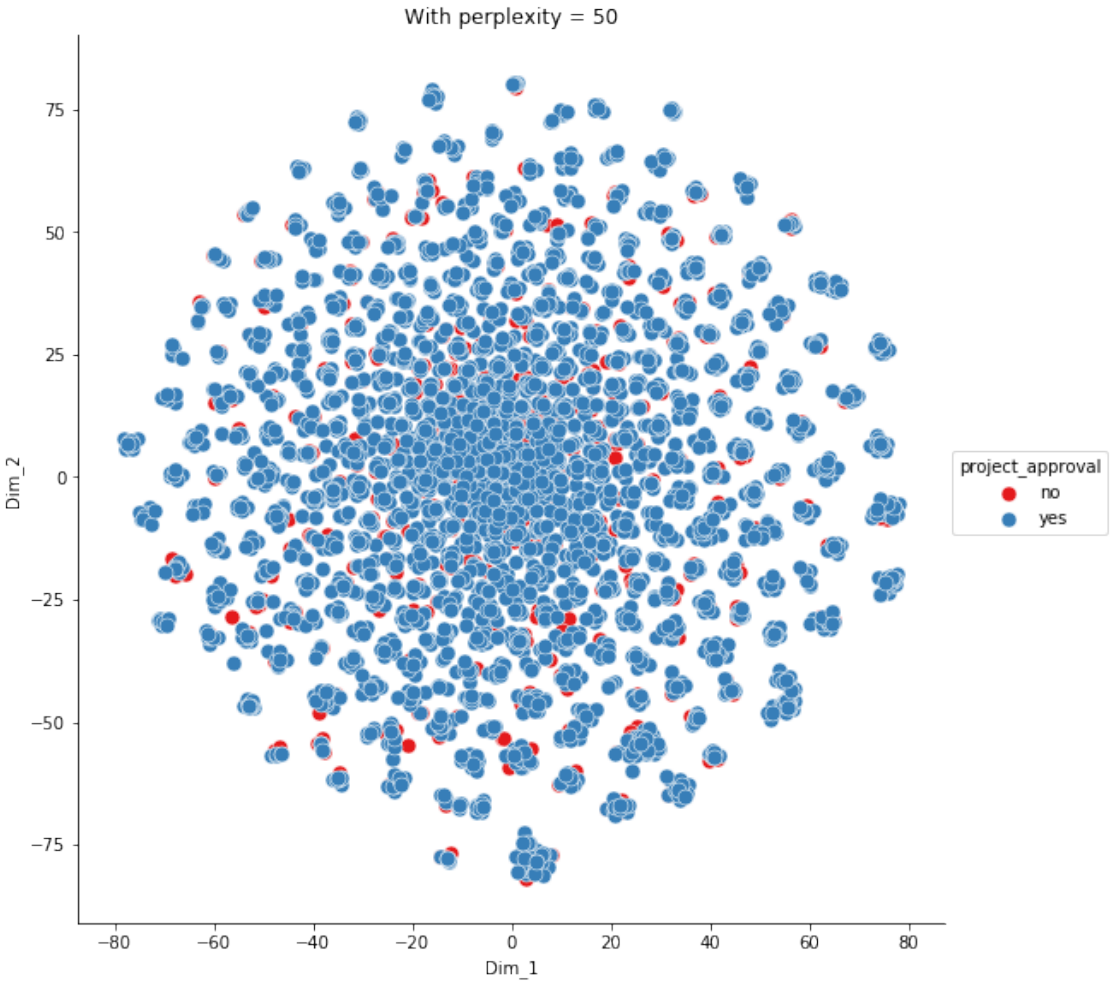
Out[86]: (5000, 413)

3.2.1 tSNE (tfidf)

In [87]:
```
# Data-preprocessing: Standardizing the data

from sklearn.preprocessing import StandardScaler
standardized_data_tfidf = StandardScaler().fit_transform(XtfidfS)
print(standardized_data_tfidf.shape)
```

(5000, 413)

In [88]:
```
from sklearn.manifold import TSNE
import seaborn as sn


model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(standardized_data_tfidf)

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

# Ploting the result of tsne
kws = dict(s=80, linewidth=.5, edgecolor="w")
sn.FacetGrid(tsne_df, hue="project_approval", size=8, palette="Set1").map(plt.scatter, 'Dim_1', 'Dim_
plt.title('With perplexity = 50')
plt.show()
```



3.3 2.3 TSNE with AVG W2V encoding of project_title feature

In [89]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Xw2v = hstack((sub_categories_one_hot, tnppp_standardized, sent_vectors))
Xw2vS = Xw2v.toarray()
Xw2vS.shape
```

Out[89]: (5000, 81)

3.3.1 tSNE (Avg W2V)

In [90]:
```
# Data-preprocessing: Standardizing the data

from sklearn.preprocessing import StandardScaler
standardized_data_w2v = StandardScaler().fit_transform(Xw2vS)
print(standardized_data_w2v.shape)
```

(5000, 81)

In [91]:
```
from sklearn.manifold import TSNE
import seaborn as sn


model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(standardized_data_w2v)

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

# Ploting the result of tsne
kws = dict(s=80, linewidth=.5, edgecolor="w")
sn.FacetGrid(tsne_df, hue="project_approval", size=8, palette="Set1").map(plt.scatter, 'Dim_1', 'Dim_
plt.title('With perplexity = 50')
plt.show()
```

3.4 2.4 TSNE with TFIDF Weighted W2V encoding of project_title feature

In [92]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Xtfidfw2v = hstack((categories_one_hot, tnppp_standardized, tfidf_sent_vectors))
Xtfidfw2vS = Xtfidfw2v.toarray()
Xtfidfw2vS.shape
```

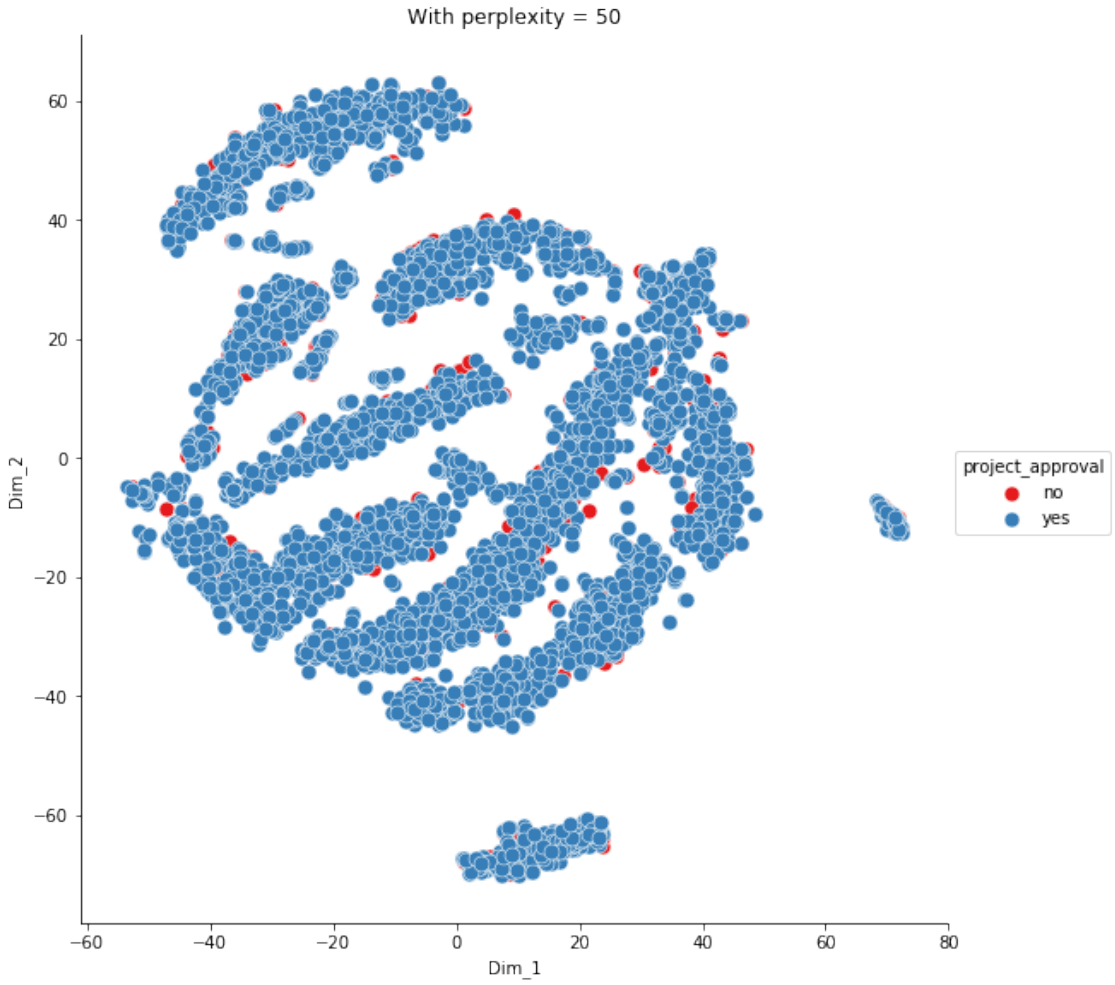
Out[92]: (5000, 60)

3.4.1 tSNE (tfidf_W2V)

In [93]:
```
# Data-preprocessing: Standardizing the data

from sklearn.preprocessing import StandardScaler
standardized_data_tfidfw2v = StandardScaler().fit_transform(Xtfidfw2vS)
print(standardized_data_tfidfw2v.shape)
```

(5000, 60)

In [94]:
```
from sklearn.manifold import TSNE
import seaborn as sn


model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(standardized_data_tfidfw2v)

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

# Ploting the result of tsne
kws = dict(s=80, linewidth=.5, edgecolor="w")
sn.FacetGrid(tsne_df, hue="project_approval", size=8, palette="Set1").map(plt.scatter, 'Dim_1', 'Dim_
plt.title('With perplexity = 50')
plt.show()
```



3.5 Aggregating all Features

In [95]:
```
# 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 :)
XF = hstack((categories_one_hot, sub_categories_one_hot, ss_one_hot, tp_one_hot,  price_standardi
XFS = XF.toarray()
XFS.shape
```

Out[95]: (5000, 151)

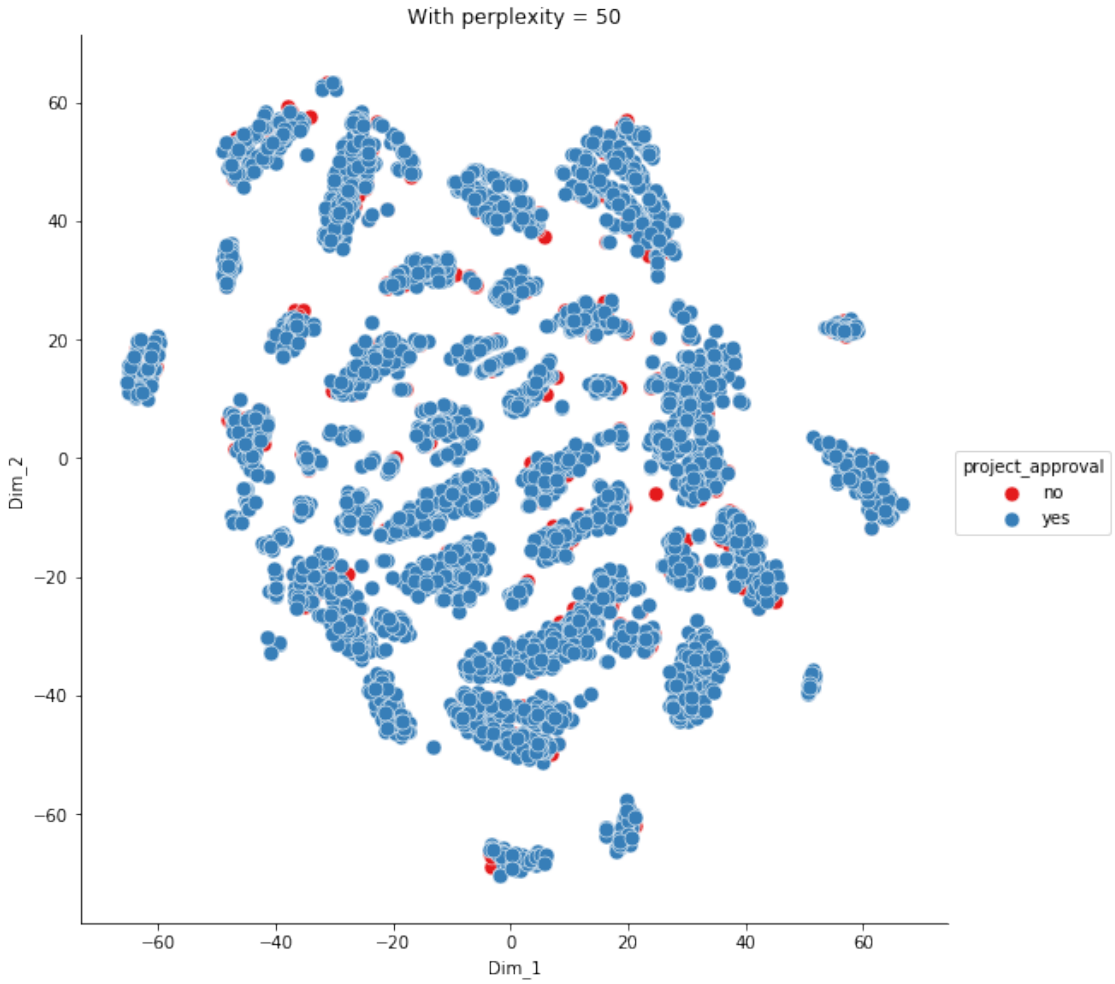
3.5.1 tSNE (final matrix aggregating all features)

In [96]:
```
# Data-preprocessing: Standardizing the data

from sklearn.preprocessing import StandardScaler
```

```
standardized_data_XFS = StandardScaler().fit_transform(XFS)
print(standardized_data_XFS.shape)
```

```
(5000, 151)
```

In [97]:
```
from sklearn.manifold import TSNE
import seaborn as sn


model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(standardized_data_XFS)

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

# Ploting the result of tsne
kws = dict(s=80, linewidth=.5, edgecolor="w")
sn.FacetGrid(tsne_df, hue="project_approval", size=8, palette="Set1").map(plt.scatter, 'Dim_1', 'Dim_
plt.title('With perplexity = 50')
plt.show()
```



2.5 Summary

- dense clusters are formed when we use tfidf_w2v & avg w2v for project title
- in comparison betn tfidf_w2v & avg w2v, more dense clusters are formed in tfidf w2v, therfore tfidf_w2v contributes more to classifying the data
- when we use tfidf or bow in vectorizing the text data, plots of tsne are fully overlapped, we cant anything from thses plots
- even though dense clusters are formed in aggregating all features and applying tsne, we cannot classify the datapoints at all, as both the polarity of data points are overlapped.
- of all the methods of vectorizing the text data, tfidf-w2v performs better in classifying the datapoints.