DonorsChoose

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

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- 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.

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

Feature	Description
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
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.

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."

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 [397]:

```
%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

print('all done')
```

```
all done
```

1.1 Reading Data

In [398]:

```
project_data = pd.read_csv('C:/Users/Harry Singh/Desktop/revesion Applied AI course/Assinments/2nd
assinment/train_data.csv')
resource_data= pd.read_csv('C:/Users/Harry Singh/Desktop/revesion Applied AI course/Assinments/2nd
assinment/resources.csv')
print(type(project_data))# this is the pandas data frame
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [399]:

```
# take half approved and half rejected projects  then do apply tsne on this dataset
print(project_data.shape)
```

```
(109248, 17)
```

```
(109248, 17)
```

In [400]:

```
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 [401]:

```
print("Number of data points in resouce data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
print(project_data.head(2))
# For a single project we can have many resources so that these rows are very large in number as c
ompared to projects data
```

```
Number of data points in resouce data (1541272, 4)
['id' 'description' 'quantity' 'price']
   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_categories     project_subject_subcategories  \
0                Literacy & Language                     ESL, Literacy   
1  History & Civics, Health & Sports  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  
```

Note

--I will take only first 3000 rows for analysis beacuse my laptop is very slow it can't handls the data it takes loading again

and again and sometimes it hang then it restart just because of this large data so thats why i am using less data points

So i'll do my all analysis on first 3000 datapoints.

---The big reason of this is i just completed this assinment in one day, but for apppying tsne it took 3 days just loading again and again so that''s why i have to do this. and my jupyter notebook hangs as well with this.

---Mullticore tsne can't downloading,and i have to use sklearn tsne.

In [402]:

```
resource_data=resource_data[:3000]
project_data=project_data[:3000]
print(project_data.shape)
print(resource_data.shape)
```

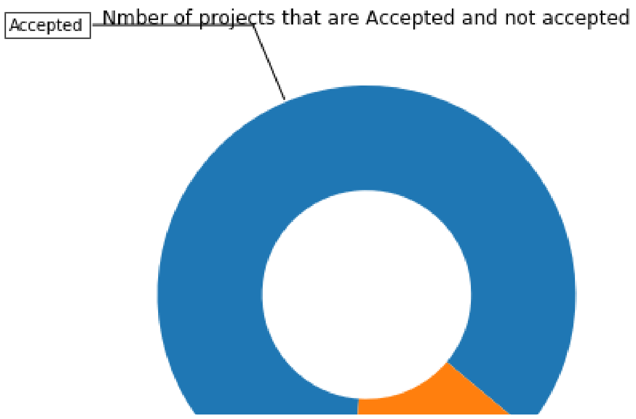
```
(3000, 17)
(3000, 4)
```

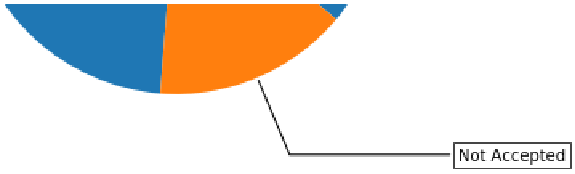
1.2 Data Analysis

In [403]:

```
# 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-p
ie-and-polar-charts-pie-and-donut-labels-py


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


# Now look at the same thing in a interactive way Pie plot
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  2547 , ( 84.89999999999999 %)
Number of projects thar are not approved for funding  453 , ( 15.1 %)
```





Summary

As we saw,this is the imbalanced dataset,Number of approved projects is greater than not approved projects.

So 84 % projects are approved and only 15 % are not approved.

1.2.1 Univariate Analysis: School State

In [404]:

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

temp = pd.DataFrame(project_data.groupby("school_state")
["project_is_approved"].apply(np.mean)).reset_index()
#When we reset the index, the old index is added as a column, and a new sequential index is used:

# if you have data which contain only 0 and 1, then the mean = percentage
temp.columns = ['state_code', 'num_proposals']
```

In [405]:

```
# 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        0.50000
7          DC        0.62500
0          AK        0.75000
21         ME        0.75000
42         TN        0.77551
==================================================
States with highest % approvals
   state_code  num_proposals
41         SD            1.0
26         MT            1.0
28         ND            1.0
11         HI            1.0
50         WY            1.0
```

__Summary:

__1.Their is a great variability here in the number of projects approved, according to states.

__2.As we saw the state sd,mt,nd,hi,wh has all the project proposals approved but we can't conclude because who knows these states has just got less project proposals like 2 or 4.so next we will see how many projects hs gotten by particulat state and how much apporved

In [406]:

```
#stacked bar plots matplotlib:
https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
# IF you want to make the interactive plots u also have to change the plots accoring to your data
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))# all the states in the axis
    plt.legend((p1[0], p2[0]), ('total', 'accepted'))
    plt.show()

print(project_data.shape[0])
```

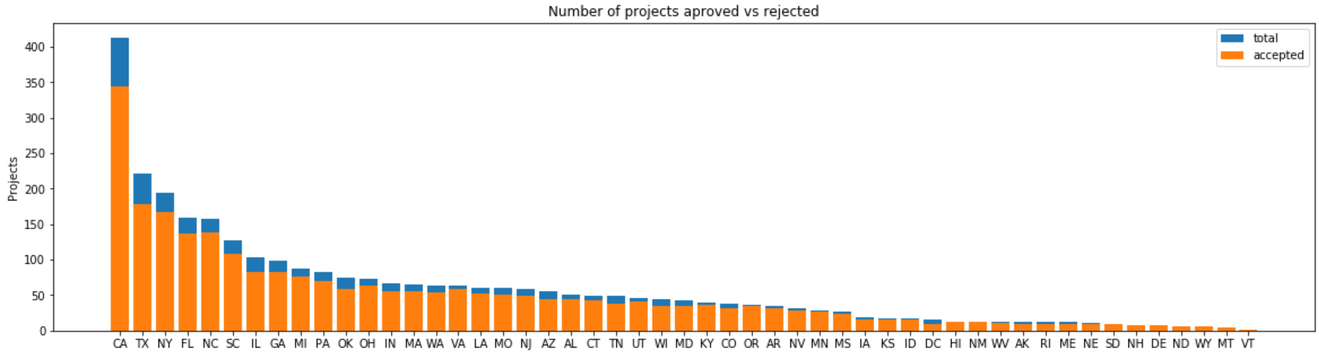
```
3000
```

In [407]:

```
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(
)

    # second line means where  project approved (ther is one) so sum of all


    # Pandas dataframe grouby count: https://stackoverflow.com/a/19385591/4084039
    temp['total'] = pd.DataFrame(project_data.groupby(col1)
[col2].agg({'total':'count'})).reset_index()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':'mean'})).reset_index()[
'Avg']
    # we hve to write reset_index['avg'] otherwise it not reset the index accr it just shows o
    if top:
        temp=temp[0:top]
    temp.sort_values(by=['total'],inplace=True, ascending=False)

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

In [408]:

```
univariate_barplots(project_data, 'school_state', 'project_is_approved')
```



```
   school_state  project_is_approved  total       Avg
4            CA                  345    413  0.835351
43           TX                  179    221  0.809955
34           NY                  167    194  0.860825
9            FL                  137    159  0.861635
27           NC                  139    157  0.885350
==================================================
   school_state  project_is_approved  total  Avg
8            DE                    7      7  1.0
28           ND                    6      6  1.0
50           WY                    6      6  1.0
26           MT                    4      4  1.0
46           VT                    1      2  0.5
```

SUMMARY: 1.Every state has greater than 80% success rate in approval instead of vt

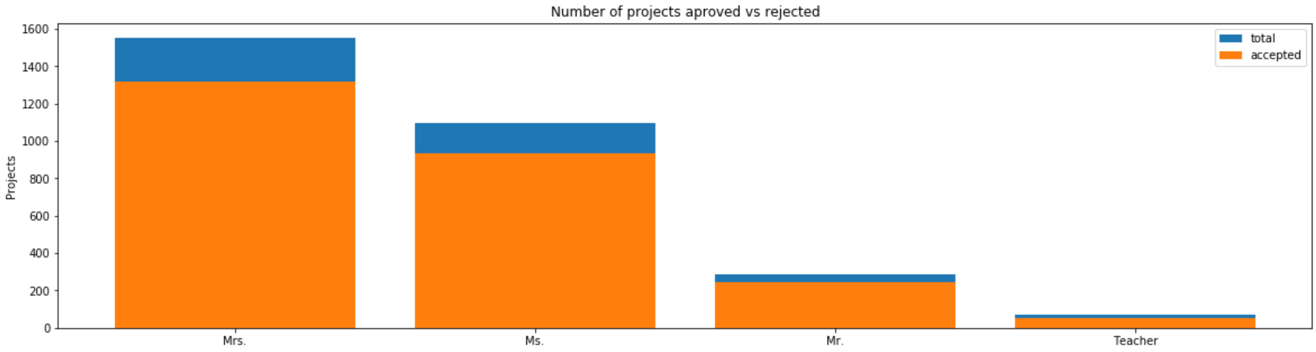
__2. WE can see in number of the projects approved in the different states has a very great spread, some states has a projets approved as low as 1 and also got less projects , and some states approval_rate has as high as 413 ,you can saw from this plot.

__3 ca state got higher projects as compared to all other states

1.2.2 Univariate Analysis: teacher_prefix

In [409]:

```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved')
```



```
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 1317   1553  0.848036
2            Ms.                  933   1095  0.852055
0            Mr.                  246    284  0.866197
3        Teacher                   51     68  0.750000
==================================================
  teacher_prefix  project_is_approved  total       Avg
1           Mrs.                 1317   1553  0.848036
2            Ms.                  933   1095  0.852055
0            Mr.                  246    284  0.866197
3        Teacher                   51     68  0.750000
```

__Summary:

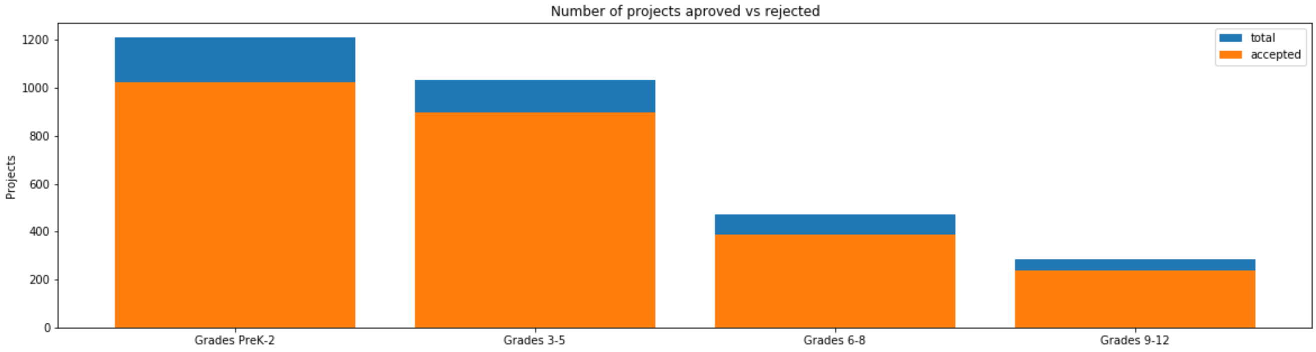
__1.You can see teachers prefixes MRS,MR ans MS has the higher number of projects and approval rate is also high above 80%.

__2. teachers got less projects as compated to all so we can see the variability is high.

1.2.3 Univariate Analysis: project_grade_category

In [410]:

```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved')
```



```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 1025   1211  0.846408
0             Grades 3-5                  897   1032  0.869186
1             Grades 6-8                  388    472  0.822034
2            Grades 9-12                  237    285  0.831579
```

```
2            Grades 9-12                   237    285  0.831579
==================================================
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                 1025   1211  0.846408
0             Grades 3-5                  897   1032  0.869186
1             Grades 6-8                  388    472  0.822034
2            Grades 9-12                  237    285  0.831579
```

__Summary:

__1.Just looking at we can say high variability of projects approved in the grades ranges.

__2.Grades prek_2 and 3-5 has higher number of project proposals and acceptance range of all is 85%

__3. All the grades has greater than 80% projects proposals are accepted.

1.2.4 Univariate Analysis: project_subject_categories

In [411]:

```
# we have to remove the commas in the project categories so that we can plot it by using the  univ
ariate funciotn



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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [412]:

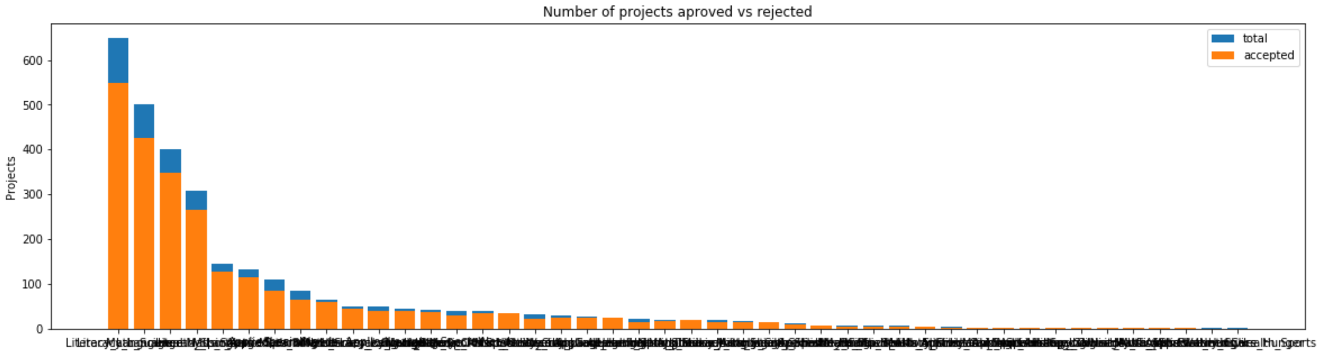
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[412]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [413]:

```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved',top=50)
```



```
                  clean_categories  project_is_approved  total       Avg
21               Literacy_Language                  548    649  0.844376
28                    Math_Science                  425    502  0.846614
24  Literacy_Language Math_Science                  348    401  0.867830
7                    Health_Sports                  266    308  0.863636
35                      Music_Arts                  128    146  0.876712
==================================================
                        clean_categories  project_is_approved  total  Avg
27  Literacy_Language Warmth Care_Hunger                    1      1  1.0
36            Music_Arts AppliedLearning                    1      1  1.0
37             Music_Arts History_Civics                    1      1  1.0
39         Music_Arts Warmth Care_Hunger                    0      1  0.0
41            SpecialNeeds Health_Sports                    0      1  0.0
```

__Summary:

__1.We can see that their is a great amount of spread in the projects proposals acccoring to subjects. We can see from the info that Literacy_Language subject has proposals as high as 649 and Music_Arts Warmth Care_Hunger has as low as 0 means no porposal from this subject.

__2. SO we can conclude in the Literacy_Language ,Math_Science and combined(Literacy_Language and Math_Science) domain their are lots of project proposols as compared to others.So these are the subjects with great project proposols and high acceptance >80%.

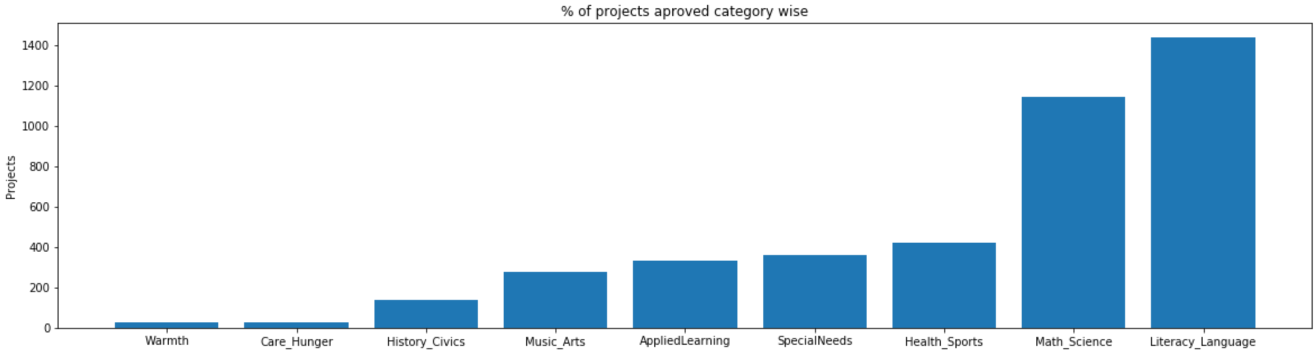
In [414]:

```
# 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())

print(my_counter)
# this tells that count of a particular word in our corpus means dataset
```

```
Counter({'Literacy_Language': 1439, 'Math_Science': 1143, 'Health_Sports': 420, 'SpecialNeeds':
362, 'AppliedLearning': 332, 'Music_Arts': 279, 'History_Civics': 137, 'Warmth': 29,
'Care_Hunger': 29})
```

In [415]:

```
# 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.title('% of projects aproved category wise')
plt.xticks(ind, list(sorted_cat_dict.keys()))
plt.show()
```



In [416]:

```
for i, j in sorted_cat_dict.items():
    print("{:30} :{:30}".format(i,j))
```

```
Warmth                         :                             29
Care_Hunger                    :                             29
History_Civics                 :                            137
Music_Arts                     :                            279
AppliedLearning                :                            332
SpecialNeeds                   :                            362
Health_Sports                  :                            420
Math_Science                   :                           1143
Literacy_Language              :                           1439
```

__Summary:

__1.Now we can see form this bar plot that literacy_language has the higher count in the corpus means we can say that this subject has the higher number of project approvals from all.

__2.And their is a great spread in terms of projects approvals accoring to subjects.

1.2.5 Univariate Analysis: project_subject_subcategories

In [417]:

```
# Do the samet thing which we did with project_catogoreis like remove all commas, and unecessary ke
ywords.

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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [418]:

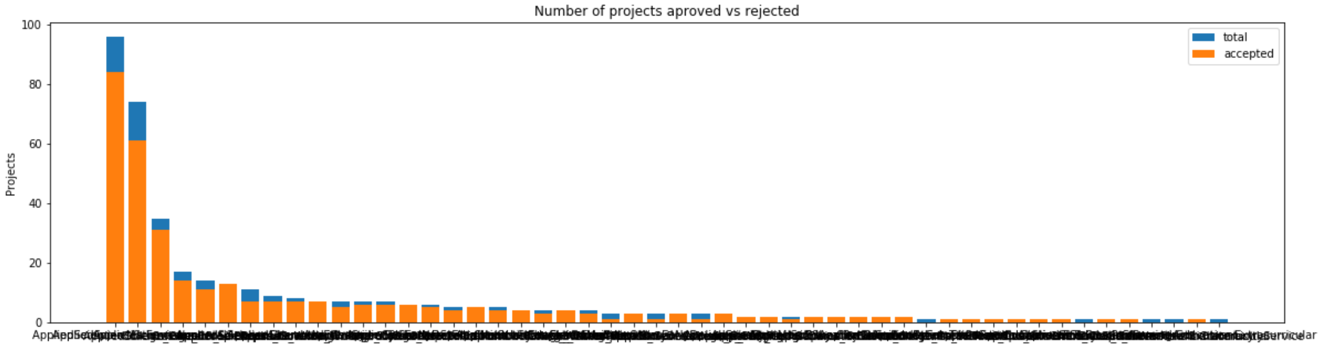
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[418]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [419]:

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



```
                         clean_subcategories  project_is_approved  total       Avg
12          AppliedSciences Mathematics                        84     96  0.875000
0                       AppliedSciences                        61     74  0.824324
6   AppliedSciences EnvironmentalScience                       31     35  0.885714
8     AppliedSciences Health_LifeScience                       14     17  0.823529
44                     College_CareerPrep                       11     14  0.785714
==================================================
                         clean_subcategories  project_is_approved  total  Avg
31   CharacterEducation ParentInvolvement                        1      1  1.0
37    Civics_Government FinancialLiteracy                        0      1  0.0
36      Civics_Government Extracurricular                        0      1  0.0
35     Civics_Government CommunityService                        1      1  1.0
25     CharacterEducation Extracurricular                        0      1  0.0
```

__Summary:

__1.We can see that their is a great amount of spread in the projects proposals acccoring to project_subject_subcategories. We can see from the plot AppliedSciences Mathematics subject_subcategories has proposals as high as 84 and CharacterEducation ParentInvolvement has as low as 1 porposal from this subject.

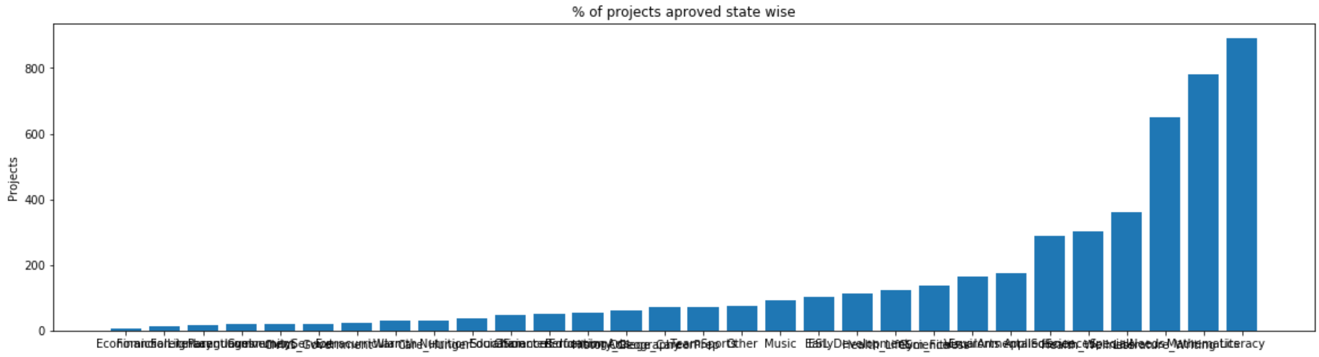
__2. SO we can say that in the AppliedSciences Mathematics and AppliedSciences their are great number of project proposals and average acceptance rate is greater than 80%.

In [420]:

```
# 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 [421]:

```
# 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()
```



In [422]:

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

```
Economics            :         6
FinancialLiteracy    :        12
ForeignLanguages     :        16
ParentInvolvement    :        19
CommunityService     :        19
Civics_Government    :        21
Extracurricular      :        24
Warmth               :        29
Care_Hunger          :        29
NutritionEducation   :        36
SocialSciences       :        49
CharacterEducation   :        51
PerformingArts       :        56
History_Geography    :        61
College_CareerPrep   :        71
TeamSports           :        73
Other                :        75
Music                :        91
ESL                  :       104
EarlyDevelopment     :       113
Health_LifeScience   :       125
Gym_Fitness          :       139
VisualArts           :       166
EnvironmentalScience :       174
AppliedSciences      :       289
Health_Wellness      :       301
SpecialNeeds         :       362
Literature_Writing   :       650
Mathematics          :       780
Literacy             :       891
```

__Summary:

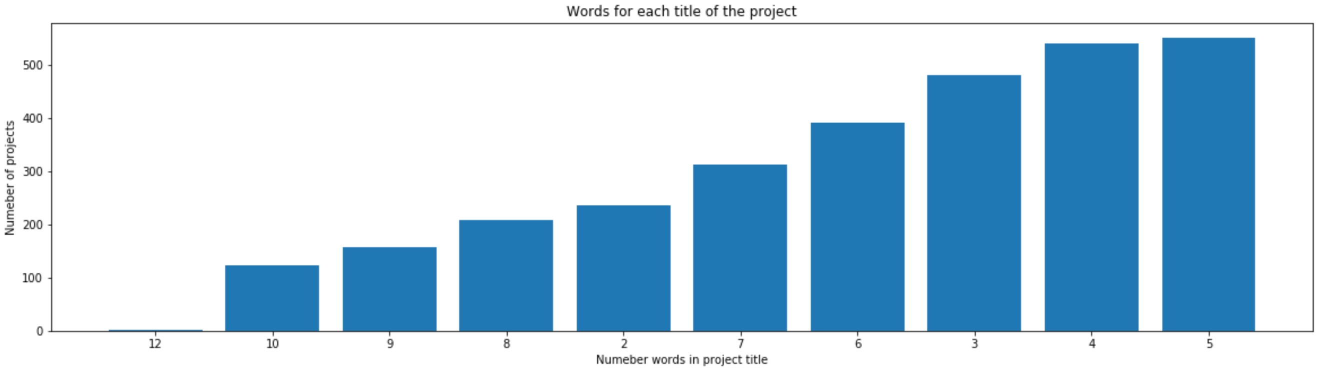
__1.Now we can see form this bar plot that literacy has the higher count in the corpus means we can say that this subject has the higher number of project approvals from all.

__2.And their is a great spread in terms of projects approvals accoring to subjects subcategories.

1.2.6 Univariate Analysis: Text features (Title)

In [423]:

```
#How to calculate number of words in a string in DataFrame:
https://stackoverflow.com/a/37483537/4084039

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


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

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



__Summary:

__1.We see from the plot that the title which has 3 or 4 or 5 words has the the highest project proposals.

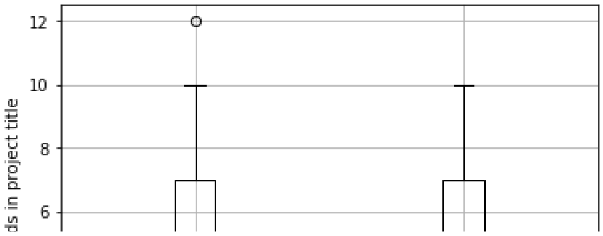
__2.We can see the spread and Also we can see that the title which has (12) words has lower than 10 proposals.

In [424]:

```
# See the amount of acceptance and rejectance of the project proposals.

approved_title_word_count = project_data[project_data['project_is_approved']==1]['project_title'].
str.split().apply(len).values
rejected_title_word_count = project_data[project_data['project_is_approved']==0]['project_title'].
str.split().apply(len).values
```

In [425]:

```
# 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()
```



Wor
4
2
Approved Projects
Rejected Projects

__Summary:

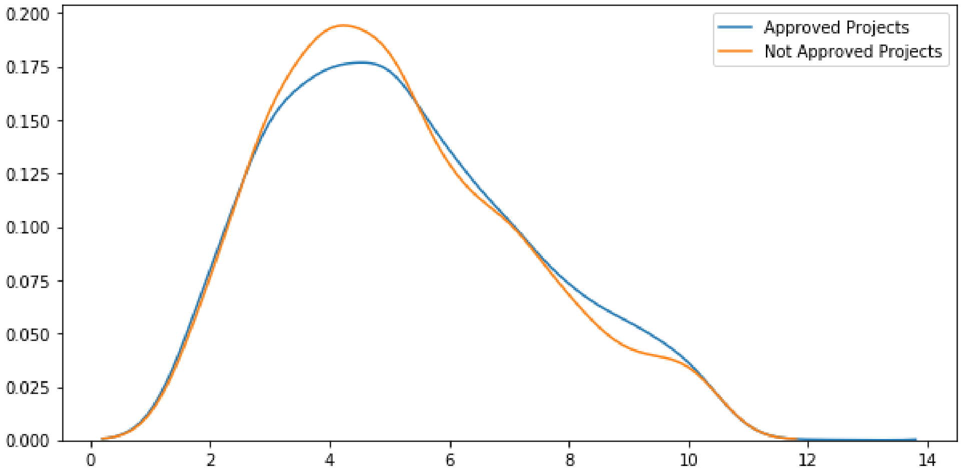
__1.Actually the bar plots of approved and rejected projects (based on the number of words in his title) looks like similar

__2.Spread is basically high in both the bar plots.

__3. Their is also one outlier in the approved projects.

In [426]:

```
plt.figure(figsize=(10,5))
sns.kdeplot(approved_title_word_count,label="Approved Projects", bw=0.6)# pdf of the Approved project proposals
sns.kdeplot(rejected_title_word_count,label="Not Approved Projects", bw=0.6)# pdf of the Not Approved project proposals
plt.legend()
plt.show()
```



__Summary:

__1. Pdf of the both approved projects proposals title words and the not approved projects proposals's (title words range) is basically the same and strictly overlapping but we can say that 3 to 6 title words's projects are more in the dataset.

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

In [427]:

```
# 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)

project_data.head(2)
```

Out[427]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra

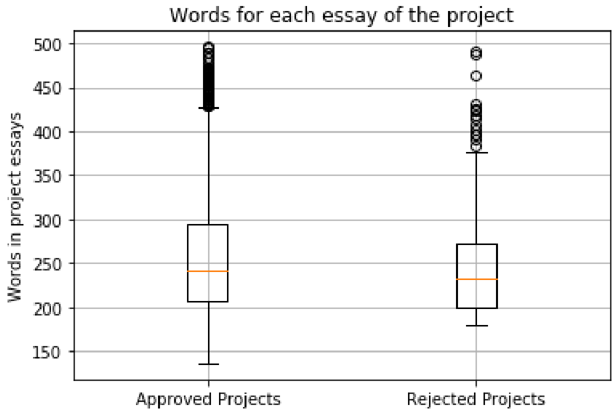
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [428]:

```
approved_word_count = project_data[project_data['project_is_approved']==1]['essay'].str.split().app
ly(len).values

rejected_word_count = project_data[project_data['project_is_approved']==0]['essay'].str.split().app
ly(len).values
```

In [429]:

```
# 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()
```



__Summary:

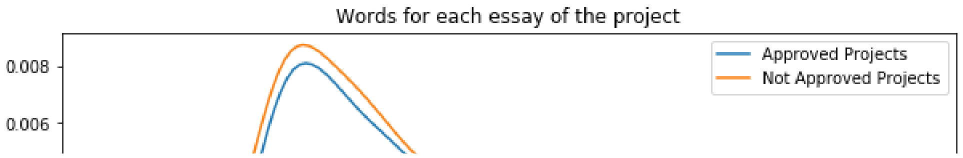
__1. Mean of both the bar plots look like same, so can't say much about this.

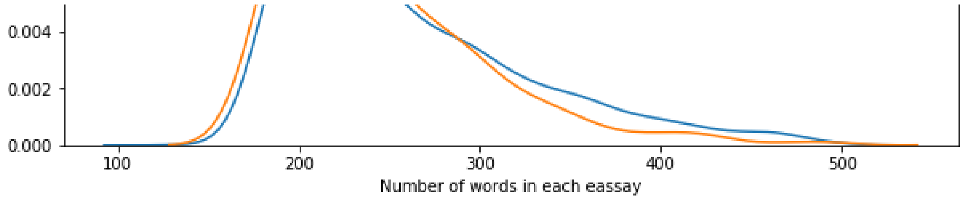
__3.As compared the project which are not appproved essay words,Approved projects essay words has the higher range or spread.

__4.Also their are many outliers in the essay words in the Approved and rejected projects proposals

In [430]:

```
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()
```





__Summary:

__1. Pdf of the both approved projects proposals and the not approved projects proposals's (essay words range) is basically the same and strictly overlapping but we can say that 190 to 250 Essay words's projects are more in the dataset.

__2. Also the peak of the Not Approved projects proposals is higher means the essay which contains the close to 200 words has the higher chances of rejected as compared to approved.

1.2.8 Univariate Analysis: Cost per project

In [431]:

```
# we get the cost of the project using resource.csv file
resource_data.head(2)
resource_data['price']=resource_data['price'].astype(int)
print(resource_data.shape)
print(resource_data['price'])
```

```
(3000, 4)
0       149
1        14
2         8
3        13
4        24
5        16
6         9
7        10
8         9
9         9
10        5
11        7
12        5
13        0
14      149
15      129
16      129
17      129
18      129
19        4
20        4
21        4
22        4
23        4
24        4
25      149
26        8
27        8
28        8
29        8
       ...
2970      3
2971      3
2972     59
2973    149
2974    299
2975     33
2976     58
2977      9
2978      8
2979      7
2980     18
2981      9
2982      9
2983      7
2984      9
```

```
2985       9
2986       8
2987      20
2988       6
2989      15
2990      14
2991      15
2992      18
2993      15
2994       8
2995       5
2996       5
2997       5
2998       5
2999       5
Name: price, Length: 3000, dtype: int32
```

In [432]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in
-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'count'}).reset_index()
price_data.head(10)

# It means In a particular project how much resouces we need its total price and its quantity  (--
Id-> here is the project id)
```

Out[432]:

	id	price	quantity
0	p000502	448	1
1	p002896	19	1
2	p003401	55	5
3	p003483	77	7
4	p006068	546	16
5	p007221	42	4
6	p009240	158	11
7	p010784	935	31
8	p012075	107	1
9	p012462	110	1

In [433]:

```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')# we are using  the left join
here
# the left join return all teh records in the project_data and return the matches records from the
resources data.
```

In [434]:

```
# convert to nan
project_data=project_data.fillna(0)
```

In [435]:

```
import math
approved_price = project_data[project_data['project_is_approved']==1]['price'].values
rejected_price = project_data[project_data['project_is_approved']==0]['price'].values

# convert nan to zero
newlist=[]
for item in approved_price:
    x = float(str(item))

    if not math.isnan(x):
        newlist.append(int(x))
    else:
        newlist.append(0)


newlist2=[]
for item in rejected_price:

    x = float(str(item))

    if not math.isnan(x):
        newlist2.append(int(x))
    else:
        newlist2.append(0)
print(len(newlist))
print(len(newlist2))
```

```
2547
453
```

In [436]:

```
# box plot of this can't make any sence
'''-> in rejected_price all the values are zero;
   -> in approved_price just 2 values are non zero ,other wise all zeros'''
# we have lots of zeros because there are many prices and quanties of products in resource_data which products are not in project_data.
#---------------------------------------------------------------------------------------









# # 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()
```

Out[436]:

```
'-> in rejected_price all the values are zero;\n   -> in approved_price just 2 values are non zero ,other wise all zeros'
```

In [437]:

```
# pdf of this not make any sence reason  told  in upper cell




# 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()
```

In [438]:

```
# percentiles of prices also don't make any  sence because of these all nan values



#----------------------------------------------------------------------
```

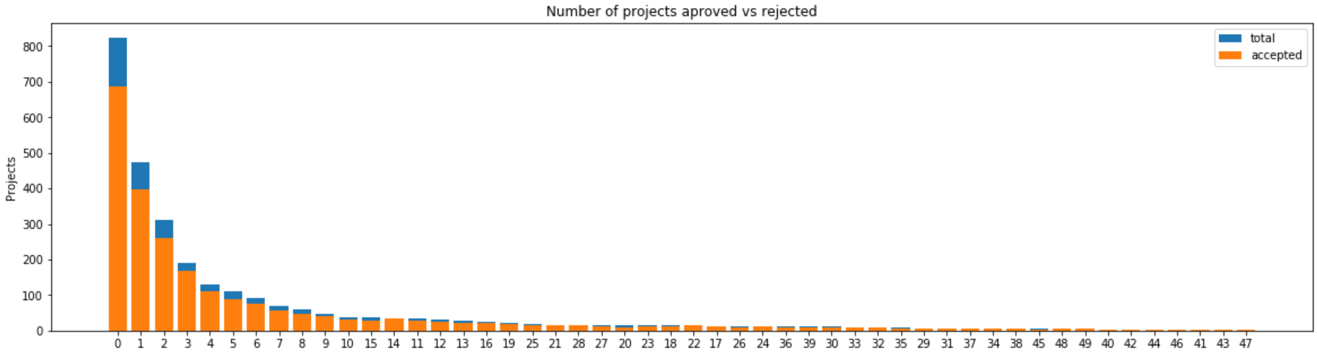
```
# # http://zetcode.com/python/prettytable/
# from prettytable import PrettyTable
# x = PrettyTable()
# x.field_names = ["Percentile", "Approved Projects", "Not Approved Projects"]

# 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)
```

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

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

In [439]:

```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects',
'project_is_approved',top=50)
```



```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                  687    824   
1                                             1                  397    473   
2                                             2                  261    312   
3                                             3                  168    191   
4                                             4                  111    130   

        Avg  
0  0.833738  
1  0.839323  
2  0.836538  
3  0.879581  
4  0.853846  
==================================================
    teacher_number_of_previously_posted_projects  project_is_approved  total  \
44                                            44                    4      4   
46                                            46                    4      4   
41                                            41                    3      3   
43                                            43                    3      3   
47                                            47                    3      3   

    Avg  
44  1.0  
46  1.0  
41  1.0  
43  1.0  
47  1.0  
```

__Summary:

__1.We clearly see from the plot, that the teacher whose previiuly posted projects rate is 0-1 has the more project Approved proposals ,they also has more project proposals as comparted to the teachers who has previouly posted 43-47 projects.

__2. Their is a great spread in the teacher_number_of_previously_posted_projects feature.

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 [440]:

```
#https://stackoverflow.com/questions/19859282/check-if-a-string-contains-a-number

def hasNumbers(inputString):
    return bool(re.search(r'\d', inputString))


approved_word_count = project_data[project_data['project_is_approved']==1]
['project_resource_summary'].values
not_approved_word_count = project_data[project_data['project_is_approved']==0]
['project_resource_summary'].values
find=0

for inputString in approved_word_count:
    if(hasNumbers(inputString))==True:
        find=find+1
b=find

print('Approved ->',approved_word_count.size,' -> Presence of the numerical digits in the project_
resource_summary of appproved projects is', find)
print('Approved ->',approved_word_count.size,' -> Presence of only characters in the
project_resource_summary of appproved projects is', approved_word_count.size-find)

find=0

for inputString in not_approved_word_count:
    if(hasNumbers(inputString))==True:
        find=find+1

print('Not_Approved ->',not_approved_word_count.size,' -> Presence of the numerical digits in the
project_resource_summary of not_appproved projects is', find)
print('Not_Approved ->',not_approved_word_count.size,' -> Presence of only characters in the proje
ct_resource_summary of not_appproved projects is', not_approved_word_count.size-find)


print('\n','Number of project_resource_summary has numberical digits ',find+b)
```

```
Approved -> 2547  -> Presence of the numerical digits in the project_resource_summary of appproved
projects is 373
Approved -> 2547  -> Presence of only characters in the project_resource_summary of appproved
projects is 2174
Not_Approved -> 453  -> Presence of the numerical digits in the project_resource_summary of not_ap
pproved projects is 47
Not_Approved -> 453  -> Presence of only characters in the project_resource_summary of
not_appproved projects is 406

 Number of project_resource_summary has numberical digits  420
```

__Summary:

__1.From this we clearly see that total numberical_digtis present resource summary is 420 and from this:

__ -> Their are 373 (numberical_digtis present in resource summaries) has Approved projects.

__ -> Their are 47 (numberical_digtis present in resource summaries) has Not_Approved projects

__2. So when the numerical digit present in the resouce summary the probability of approval is higher than the rejected.

1.3 Text preprocessing

1.3.1 Essay Text

In [441]:

In [441]:

```
project_data.head(2)
```

Out[441]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [ ]:

```
# 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[200])
print("="*50)
print(project_data['essay'].values[999])
print("="*50)

# As you can see we have lots of punctuation signs ,special characters andblack slashes
#so we need to remove this ,before applying the featuring teachiqnies ike baw,tf_idf etc.
```

In [ ]:

```
# 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 [ ]:

```
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
# 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', 'e
ach', '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', "d
esn't", 'hadn',\
             "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
             "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
             'won', "won't", 'wouldn', "wouldn't"]
```

In [ ]:

```
# 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())
```

In [ ]:

```
preprocessed_essays[4]
```

1.3.2 Project title Text

In [ ]:

```
# similarly you can preprocess the titles also
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_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_title.append(sent.lower().strip())
```

In [ ]:

```
preprocessed_title[0]
```

1. 4 Preparing data for models

In [ ]:

```
project_data.columns
```

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
```

Numerical data -> apply standization

Text data -> apply baw,tf-idf similar algo for converting to vectors

categorical data-> apply one hot encoding ,its actually binary encoding

1.4.1 Vectorizing Categorical data

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

In [ ]:

```
# we use count vectorizer to convert the values into one hot encoded features
# read the documentation for the CountVectorizer from sklearn
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)
print(categories_one_hot[0:10])
```

In [ ]:

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


sub_categories_one_hot = vectorizer.transform(project_data['clean_subcategories'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
print(sub_categories_one_hot[0:10])
print(sub_categories_one_hot.shape)
```

In [ ]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category als
o
# fist of all convert state, teacher_prefix and project_grade_category also into dictonaries
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039



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


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


print(sorted_state_dict)
# this tells that count of a particular word in our corpus means dataset



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


sch_one_hot = vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encodig ",sch_one_hot.shape)
print(sch_one_hot[0:10])
```

In [ ]:

```
for d in project_data['teacher_prefix'].values:
    f=str(re.sub(r'\.', ' ',str(d)))
    project_data['teacher_prefix'][d]=f
```

In [ ]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category als
o
# fist of all convert state, teacher_prefix and project_grade_category also into dictonaries
# 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['teacher_prefix'].values:
    my_counter.update(word.split())


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


print(sorted_teaher_prefix_dict)
# this tells that count of a particular word in our corpus means dataset
# apply countvecorizer
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_teaher_prefix_dict.keys()), lowercase=False, b
inary=True)
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())


tf_one_hot = vectorizer.transform(project_data['teacher_prefix'].values)
print("Shape of matrix after one hot encodig ",tf_one_hot .shape)
print(tf_one_hot [0:10])
tf_one_hot=tf_one_hot[:3000]
print(tf_one_hot.shape)
```

In [ ]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category als
o
# fist of all convert state, teacher_prefix and project_grade_category also into dictonaries
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039


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


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


print(sorted_project_grade_category_dict )
# this tells that count of a particular word in our corpus means dataset



vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys()), lowercase
=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())


grade_one_hot = vectorizer.transform(project_data['project_grade_category'].values)
print("Shape of matrix after one hot encodig ",grade_one_hot.shape)
print(grade_one_hot[0:10])
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [ ]:

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

1.4.2.2 Bag of Words on `project_title`

In [ ]:

```
# you can vectorize the title also
# 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_title = vectorizer.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

In [ ]:

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

1.4.2.3 TFIDF vectorizer

In [ ]:

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

1.4.2.4 TFIDF Vectorizer on `project_title`

In [ ]:

```
# 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_title)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

1.4.2.5 Using Pretrained Models: Avg W2V

In [ ]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/
# 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 [ ]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

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

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [ ]:

```
# Similarly you can vectorize for title also
# average Word2Vec
# compute average word2vec for each review.
```

```
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_title): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

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

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [443]:

```
# 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 [444]:

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

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

```
100%|██████████████████████████████████████████████████████████████████████████████| 3000/3000 [01
:17<00:00, 38.64it/s]
```

```
3000
300
```

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

In [445]:

```
# 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_title)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [446]:

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

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

```
100%|██████████████████████████████████████████████████████████████████████████████| 3000/3000
[00:00<00:00, 5928.80it/s]
```

```
3000
300
```

1.4.3 Vectorizing Numerical features

In [447]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

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

```
Mean : 0.101, Standard deviation : 4.1303106017183095
```

In [448]:

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

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

```
Mean : 10.598, Standard deviation : 25.62666051855632
```

In [449]:

```
print(price_standardized)
print('\n\n\n',tfp_standardized)
```

```
[[-0.02445337]
 [-0.02445337]
 [-0.02445337]
 ...
 [-0.02445337]
 [-0.02445337]
 [-0.02445337]]



 [[-0.41355369]
 [-0.14040066]
 [-0.37453183]
 ...
 [ 0.4059054 ]
 [ 0.01568679]
 [-0.33550997]]
```

1.4.4 Merging all the above features

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

In [450]:

```
print(categories_one_hot.shape)# clearn categories
print(sub_categories_one_hot.shape) # clean subcategories
print(sch_one_hot.shape)       # state
print(grade_one_hot.shape)#grade categories
print(tf_one_hot.shape)   # teacher prefixes
print(text_bow_title.shape)# title
print(price_standardized.shape)#price
print(tfp_standardized.shape)#number_of_teacher_previous_posted_projecrs
```

```
(3000, 9)
(3000, 30)
(3000, 51)
(3000, 5)
(3000, 7)
(3000, 234)
(3000, 1)
(3000, 1)
```

In [453]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
```

Out[453]:

```
(3000, 338)
```

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.

1. In the above cells we have plotted and analyzed many features. Please observe the plots and write the observations in markdown cells below every plot.
2. EDA: Please complete the analysis of the feature: teacher_number_of_previously_posted_projects
3. 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_grade_category : categorical data (one hot encoding)

- project_title : text data (BOW, TFIDF, AVG W2V, TFIDF W2V)
- price : numerical
- teacher_number_of_previously_posted_projects : numerical

4. Now, plot FOUR t-SNE plots with each of these feature sets.
   A. categorical, numerical features + project_title(BOW)
   B. categorical, numerical features + project_title(TFIDF)
   C. categorical, numerical features + project_title(AVG W2V)
   D. categorical, numerical features + project_title(TFIDF W2V)
5. Concatenate all the features and Apply TNSE on the final data matrix
6. Note 1: The TSNE accepts only dense matrices
7. Note 2: Consider only 5k to 6k data points to avoid memory issues. If you run into memory error issues, reduce the number of data points but clearly state the number of datat-poins you are using

__Note: In all the tsne plots i take numberr of iterations not more than 400 because in take lots of time,full 2 days to execute in my laptop,i am just waiting to for running these so by using this plots may be i can't make some accurate decision.

In [462]:

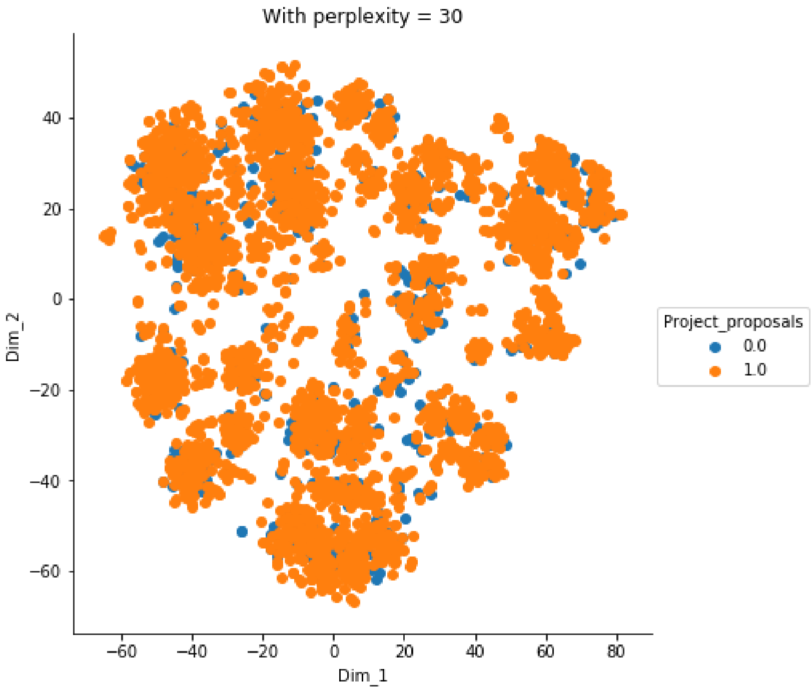
```
from scipy.sparse import hstack
from sklearn.manifold import TSNE
# 1st tsne with baw(title) and other all features

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
first= hstack((categories_one_hot, sub_categories_one_hot,sch_one_hot,grade_one_hot,tf_one_hot,
text_bow_title, price_standardized,tfp_standardized))
first=first.toarray()# dense matrix
print(first)
```

```
[[ 0.          0.          0.         ...  0.         -0.02445337
  -0.41355369]
 [ 0.          0.          1.         ...  0.         -0.02445337
  -0.14040066]
 [ 0.          0.          0.         ...  0.         -0.02445337
  -0.37453183]
 ...
 [ 0.          0.          0.         ...  0.         -0.02445337
   0.4059054 ]
 [ 0.          0.          0.         ...  0.         -0.02445337
   0.01568679]
 [ 0.          0.          0.         ...  0.         -0.02445337
  -0.33550997]]
```

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

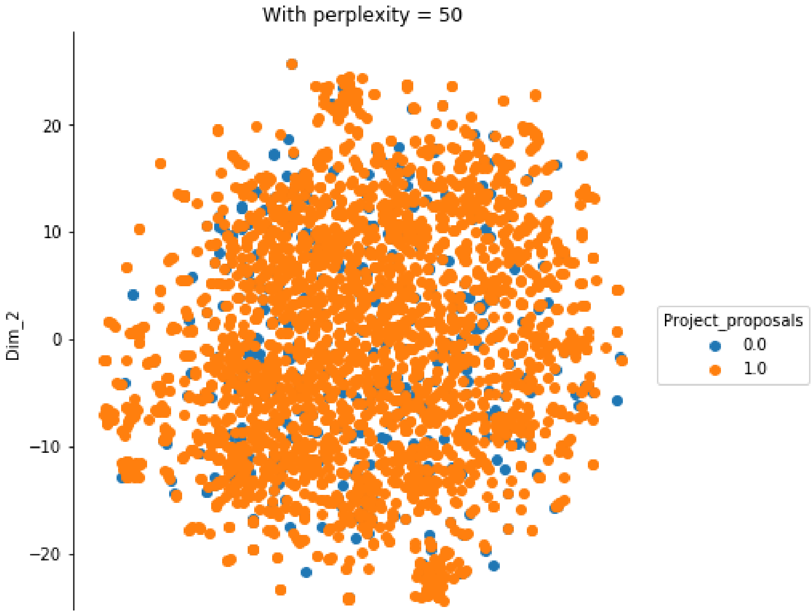
In [466]:

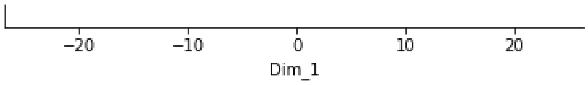
```
# please write all of the code with proper documentation and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label




y=project_data['project_is_approved']
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200,n_iter=500)

X_embedding = tsne.fit_transform(first)
#for_tsne = np.hstack((X,y))   # <- error on this line  (all the input arrays must have same numbe
r of dimensions)
for_tsne=np.column_stack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_1','Dim_2','Project_proposals'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_
legend()
plt.title('With perplexity = 30')
plt.show()
```



In [475]:

```
# 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



y=project_data['project_is_approved']
tsne = TSNE(n_components=2, perplexity=50, learning_rate=200,n_iter=400)

X_embedding = tsne.fit_transform(first)
#for_tsne = np.hstack((X,y))   # <- error on this line  (all the input arrays must have same numbe
r of dimensions)
for_tsne=np.column_stack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_1','Dim_2','Project_proposals'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_
legend()
plt.title('With perplexity = 50')
plt.show()
```





__Summary:

__1.We can see that from the 1st plot of Bow we achieve global structure,but different grouping of elements , But all are strongly overlapping in this,we can't conclude anything from this .

__2.We can see that from the wst plot of Bow we achieve global structure,bu this it not make any groups because of high prepelixity value , But all are strongly overlapping in this,we can't conclude anything from this .

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

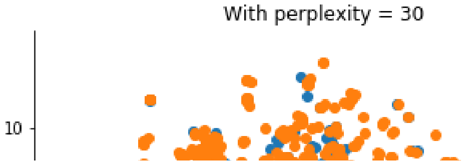
In [469]:

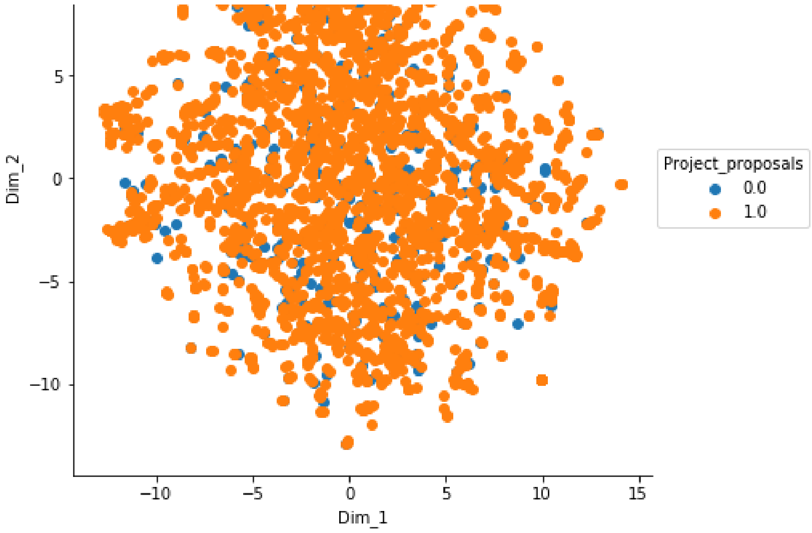
```
from scipy.sparse import hstack
from sklearn.manifold import TSNE
# 1st tsne with baw(title) and other all features

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
first= hstack((categories_one_hot, sub_categories_one_hot,sch_one_hot,grade_one_hot,tf_one_hot,
text_tfidf, price_standardized,tfp_standardized))
first=first.toarray()# dense matrix
print(first)
```

```
[[ 0.          0.          0.         ...  0.          -0.02445337
  -0.41355369]
 [ 0.          0.          1.         ...  0.          -0.02445337
  -0.14040066]
 [ 0.          0.          0.         ...  0.          -0.02445337
  -0.37453183]
 ...
 [ 0.          0.          0.         ...  0.          -0.02445337
   0.4059054 ]
 [ 0.          0.          0.         ...  0.          -0.02445337
   0.01568679]
 [ 0.          0.          0.         ...  0.          -0.02445337
  -0.33550997]]
```

In [476]:

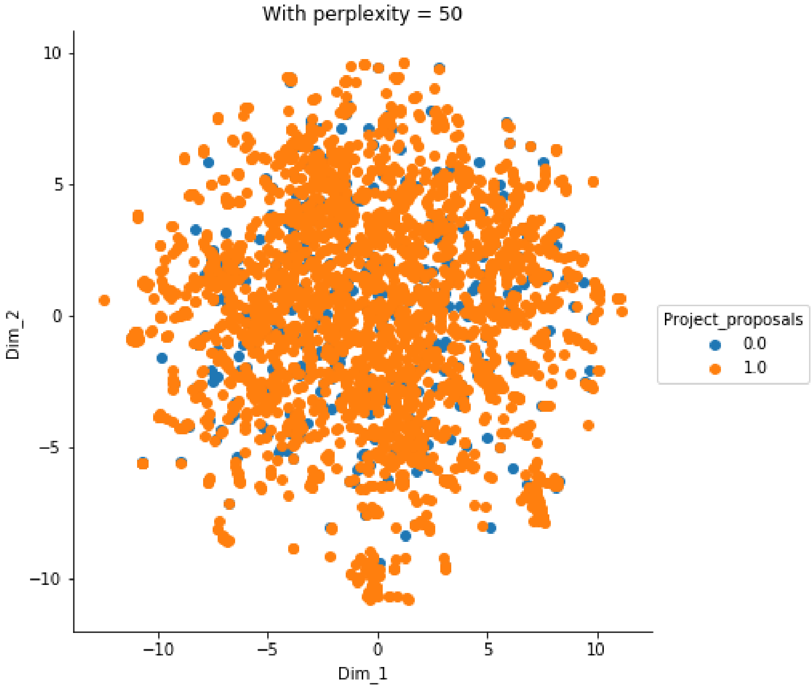
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label

y=project_data['project_is_approved']
tsne = TSNE(n_components=2, perplexity=30, learning_rate=200,n_iter=300)

X_embedding = tsne.fit_transform(first)
#for_tsne = np.hstack((X,y))    # <- error on this line  (all the input arrays must have same numbe
r of dimensions)
for_tsne=np.column_stack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_1','Dim_2','Project_proposals'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_
legend()
plt.title('With perplexity = 30')
plt.show()
```





In [477]:

```
# 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

y=project_data['project_is_approved']
tsne = TSNE(n_components=2, perplexity=50, learning_rate=200,n_iter=300)

X_embedding = tsne.fit_transform(first)
#for_tsne = np.hstack((X,y))    # <- error on this line  (all the input arrays must have same numbe
r of dimensions)
for_tsne=np.column_stack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_1','Dim_2','Project_proposals'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_
legend()
plt.title('With perplexity = 50')
plt.show()
```



__Summary:

__1.May be because of low iterations we can't see any different between these bow and tf-idf plots,all are strongly overlapping.

__2.But as we saw the variance of all the points also in 2 dimensional be high as we see from all the plots,becase we have all ranges of values.

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

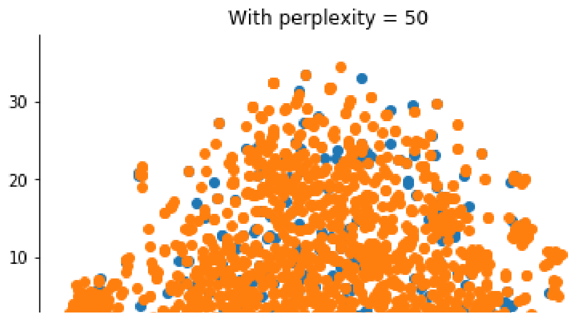
In [470]:

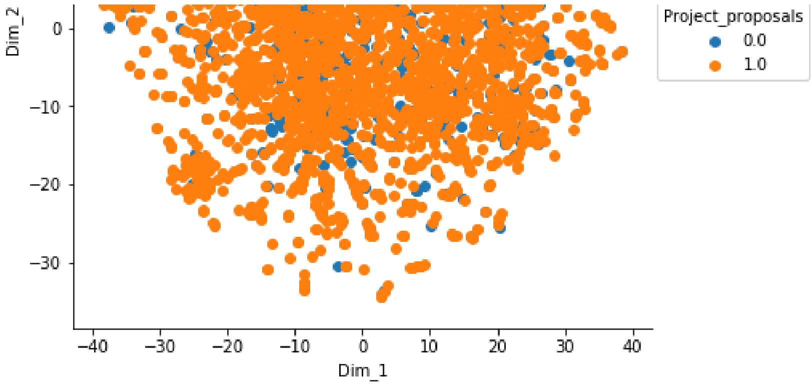
```
from scipy.sparse import hstack
from sklearn.manifold import TSNE
# 1st tsne with baw(title) and other all features

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
first= hstack((categories_one_hot,
sub_categories_one_hot,sch_one_hot,grade_one_hot,tf_one_hot,avg_w2v_vectors, price_standardized,tf
p_standardized))
first=first.toarray()# dense matrix
print(first)
```

```
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  8.56000000e-05
  -2.44533668e-02 -4.13553689e-01]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00 ... -3.51785000e-01
  -2.44533668e-02 -1.40400658e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  8.36399500e-02
  -2.44533668e-02 -3.74531828e-01]
 ...
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  1.10000333e-01
  -2.44533668e-02  4.05905404e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  3.70000000e-02
  -2.44533668e-02  1.56867884e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  4.00000000e-02
  -2.44533668e-02 -3.35509966e-01]]
```

In [479]:

```
# 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

y=project_data['project_is_approved']
tsne = TSNE(n_components=2, perplexity=50, learning_rate=200,n_iter=500)

X_embedding = tsne.fit_transform(first)
#for_tsne = np.hstack((X,y))   # <- error on this line  (all the input arrays must have same numbe
r of dimensions)
for_tsne=np.column_stack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_1','Dim_2','Project_proposals'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_
legend()
plt.title('With perplexity = 50')
plt.show()
```





__Summary:

__1. This also seems a global structure with strongly overlapping there are reasons like iteratios and as i took less data.But if we deeply see in this plots, it is non linear structure like a 2 circles(one is inside of other). If unseen datapoint lies in the outside circle then high probability that, this project 'll approve because lots of data points in the outside circle are accepted proposals.But if we use this plots our model accuracy 'll be low, so we have to make some features so that we seprate them.

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

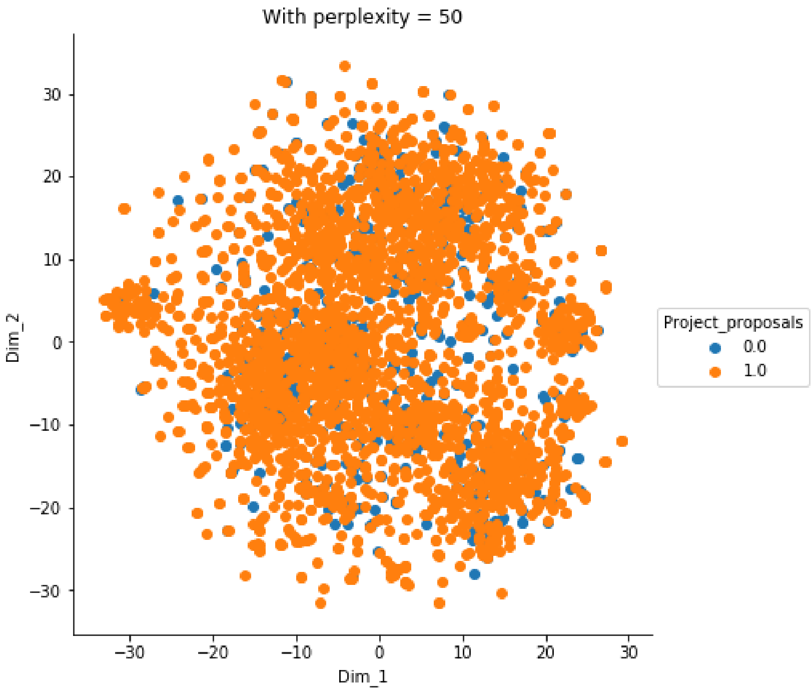
In [480]:

```
from scipy.sparse import hstack
from sklearn.manifold import TSNE
# 1st tsne with baw(title) and other all features

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
first= hstack((categories_one_hot,
sub_categories_one_hot,sch_one_hot,grade_one_hot,tf_one_hot,tfidf_w2v_vectors, price_standardized,
tfp_standardized))
first=first.toarray()# dense matrix
print(first)
```

```
[[ 0.          0.          0.         ...  0.03806801 -0.02445337
  -0.41355369]
 [ 0.          0.          1.         ... -0.33687315 -0.02445337
  -0.14040066]
 [ 0.          0.          0.         ...  0.09534681 -0.02445337
  -0.37453183]
 ...
 [ 0.          0.          0.         ...  0.1081979  -0.02445337
   0.4059054 ]
 [ 0.          0.          0.         ...  0.02312268 -0.02445337
   0.01568679]
 [ 0.          0.          0.         ...  0.0341102  -0.02445337
  -0.33550997]]
```

In [481]:

```
# 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
    # please write all the code with proper documentation, and proper titles for each subsection

y=project_data['project_is_approved']
tsne = TSNE(n_components=2, perplexity=50, learning_rate=200,n_iter=500)

X_embedding = tsne.fit_transform(first)
#for_tsne = np.hstack((X,y))    # <- error on this line  (all the input arrays must have same numbe
r of dimensions)
for_tsne=np.column_stack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_1','Dim_2','Project_proposals'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_
legend()
plt.title('With perplexity = 50')
plt.show()
```



__Summary:

__This plot also strongly overlapping,but accepted proposals are like more spread in the outside circle,as i said earlie.

Cobined tsne

In [482]:

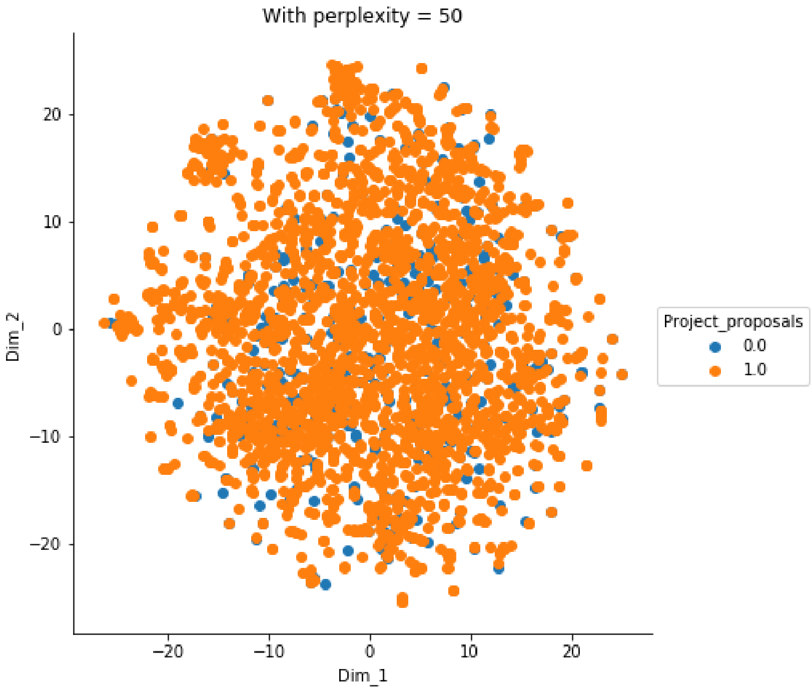
```
# combine all features and all baw,tf-idf ,avg-wortovec etc
from scipy.sparse import hstack
from sklearn.manifold import TSNE
# 1st tsne with baw(title) and other all features

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
first= hstack((categories_one_hot,
sub_categories_one_hot,sch_one_hot,grade_one_hot,tf_one_hot,tfidf_w2v_vectors,text_bow_title,text_t
fidf,avg_w2v_vectors, price_standardized,tfp_standardized))
first=first.toarray()# dense matrix
print(first.shape)
```

```
(3000, 4265)
```

In [483]:

```
y=project_data['project_is_approved']
tsne = TSNE(n_components=2, perplexity=50, learning_rate=200,n_iter=400)

X_embedding = tsne.fit_transform(first)
#for_tsne = np.hstack((X,y))   # <- error on this line  (all the input arrays must have same numbe
r of dimensions)
for_tsne=np.column_stack((X_embedding,y))
for_tsne_df = pd.DataFrame(data=for_tsne, columns=['Dim_1','Dim_2','Project_proposals'])

# Ploting the result of tsne
sns.FacetGrid(for_tsne_df, hue="Project_proposals", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_
legend()
```

```
plt.title('With perplexity = 50')
plt.show()
```



__Summary:

__1.Yes , as i said earliear it has the non linear shape ,but not 2 circles as i said before , in this its like 2 ellipses(one is inside of other). It is the plot of comnined all features like acceped proposals has more spread as compared to rejected proposals.But all are strongly overlapping.

2.5 Summary

As i said before in all the tsne plots with different techniques, All the accepted proposals and non_accepted proposals are strongly overlapping,may be this is because of less data points and (less number of iterations).But i can't use more iterations. So ,from this plots making some accurate conclusion is difficult,because all are overlapping strongly.But the main thing in the tsne plots i like if we use the preplexity value less then i'll take like local structure, but as prepleity values increaess,then the structure like the global structure.