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`
`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."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

```
C:\Users\Rahul\Anaconda3\lib\site-packages\gensim\utils.py:1209: UserWarning: detected Windows; aliasing chunkize to chunkize_seria
l
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:

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

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

In [4]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

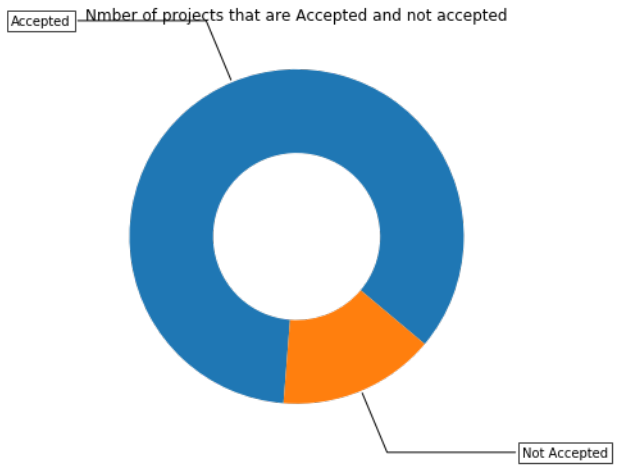
	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 Data Analysis

In [5]:

```
# PROVIDE CITATIONS TO YOUR CODE IF YOU TAKE IT FROM ANOTHER WEBSITE.
# https://matplotlib.org/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-glr-gallery-pie-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,"%)")

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

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

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

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

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

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

plt.show()
```

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



1.2.1 Univariate Analysis: School State

In [6]:

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

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

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

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

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

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

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

Out[6]:

```
'# How to plot US state heatmap: https://datascience.stackexchange.com/a/9620\n\nscl = [[0.0, \'rgb(242,240,247)\'],[0.2, \'rgb(218
,218,235)\'],[0.4, \'rgb(188,189,220)\'],            [0.6, \'rgb(158,154,200)\'],[0.8, \'rgb(117,107,177)\'],[1.0, \'rgb(84,39,143)
\']]\n\ndata = [ dict(\n        type=\'choropleth\',\n        colorscale = scl,\n        autocolorscale = False,\n        locations
= temp[\'state_code\'],\n        z = temp[\'num_proposals\'].astype(float),\n        locationmode = \'USA-states\',\n        text =
temp[\'state_code\'],\n        marker = dict(line = dict (color = \'rgb(255,255,255)\',width = 2)),\n        colorbar = dict(title
= "% of pro")\n    ) ]\n\nlayout = dict(\n        title = \'Project Proposals % of Acceptance Rate by US States\',\n        geo = d
ict(\n            scope=\'usa\',\n            projection=dict( type=\'albers usa\' ),\n            showlakes = True,\n            l
akecolor = \'rgb(255, 255, 255)\',\n        ),\n    )\n\nfig = go.Figure(data=data, layout=layout)\noffline.iplot(fig, filename=\'u
s-map-heat-map\')\n'
```

In [7]:

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

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

In [8]:

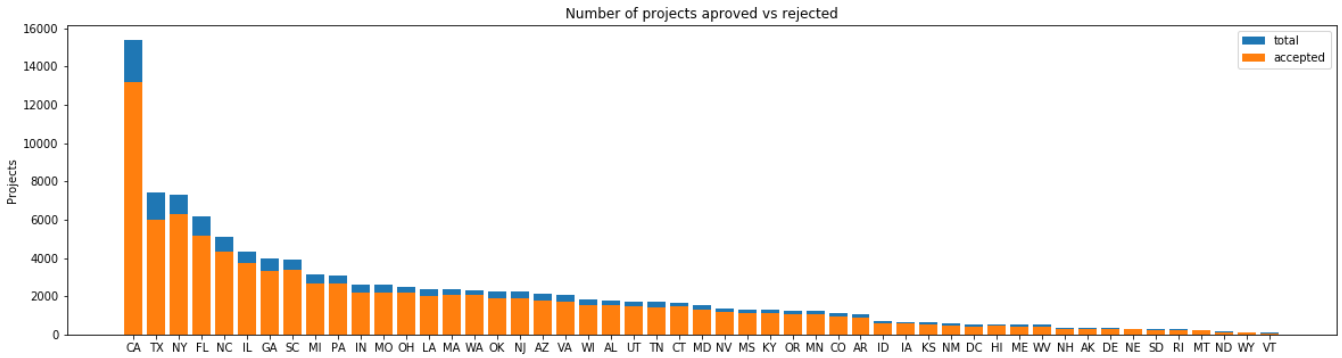
```
#stacked bar plots matplotlib: https://matplotlib.org/gallery/lines_bars_and_markers/bar_stacked.html
def stack_plot(data, xtick, col2='project_is_approved', col3='total'):
    ind = np.arange(data.shape[0])

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

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

In [9]:

```
def univariate_barplots(data, col1, col2='project_is_approved', top=False):
    # Count number of zeros in dataframe python: https://stackoverflow.com/a/51540521/4084039
    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()['total']
    temp['Avg'] = pd.DataFrame(project_data.groupby(col1)[col2].agg({'Avg':'mean'})).reset_index()['Avg']

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

    if top:
        temp = temp[0:top]

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

In [10]:

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



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

==========

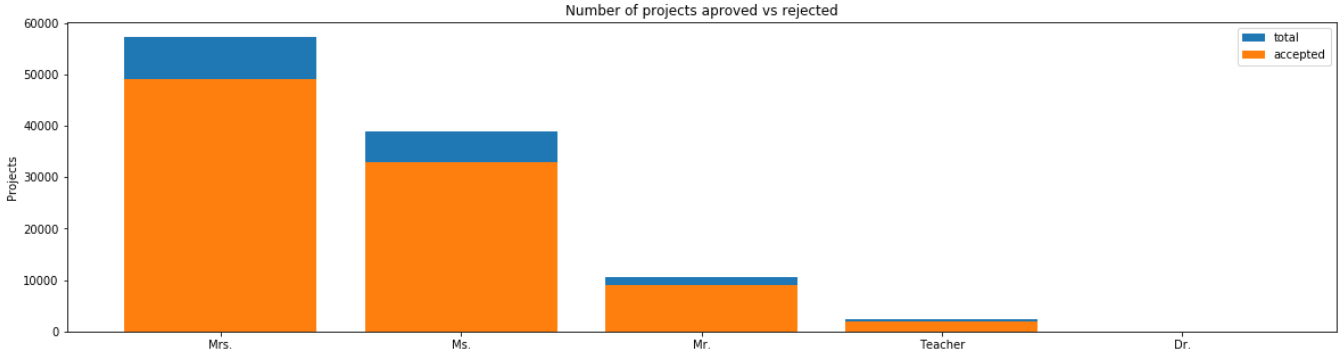
	school_state	project_is_approved	total	Avg
39	RI	243	285	0.852632
26	MT	200	245	0.816327
28	ND	127	143	0.888112
50	WY	82	98	0.836735
46	VT	64	80	0.800000

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

1.2.2 Univariate Analysis: teacher_prefix

In [11]:

```
univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=False)
```



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

==========

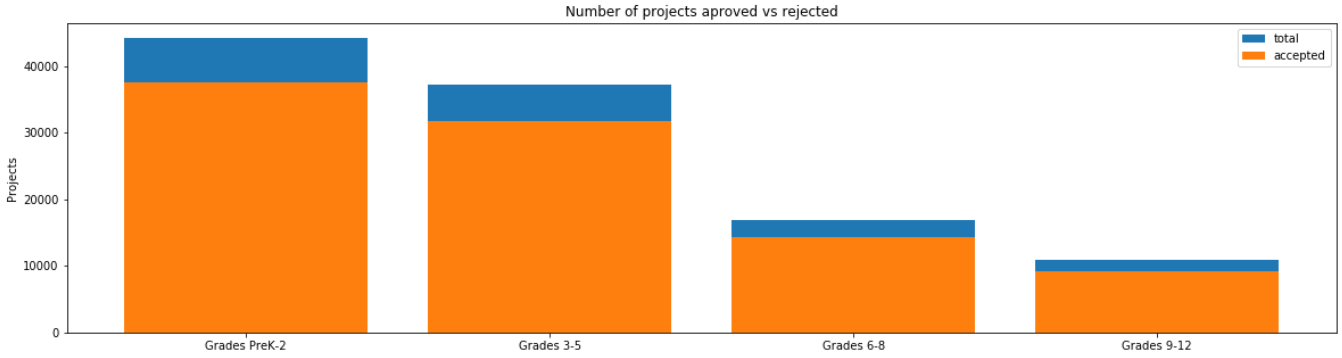
	teacher_prefix	project_is_approved	total	Avg
2	Mrs.	48997	57269	0.855559
3	Ms.	32860	38955	0.843537
1	Mr.	8960	10648	0.841473
4	Teacher	1877	2360	0.795339
0	Dr.	9	13	0.692308

SUMMARY: teachers with prefix as Mrs. have about 85% project approval rate and people with Dr, as prefix have much lower approval rates

1.2.3 Univariate Analysis: project_grade_category

In [12]:

```
univariate_barplots(project_data, 'project_grade_category', 'project_is_approved', top=False)
```




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

SUMMARY: Every grade has an approval rate higher than 80 %. Higher grades have lower approval rates

1.2.4 Univariate Analysis: project_subject_categories

In [13]:

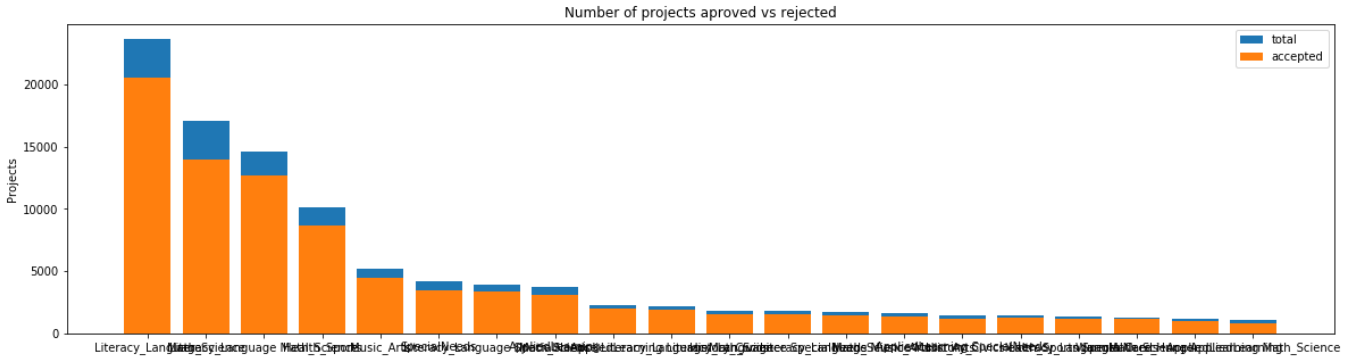
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# 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"=> "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 [14]:

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

Out[14]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_subject_sub
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	E
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Civics & Government, T



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

==========

	clean_categories	project_is_approved	total	Avg
19	History_Civics Literacy_Language	1271	1421	0.894441
14	Health_Sports SpecialNeeds	1215	1391	0.873472
50	Warmth Care_Hunger	1212	1309	0.925898
33	Math_Science AppliedLearning	1019	1220	0.835246
4	AppliedLearning Math_Science	855	1052	0.812738

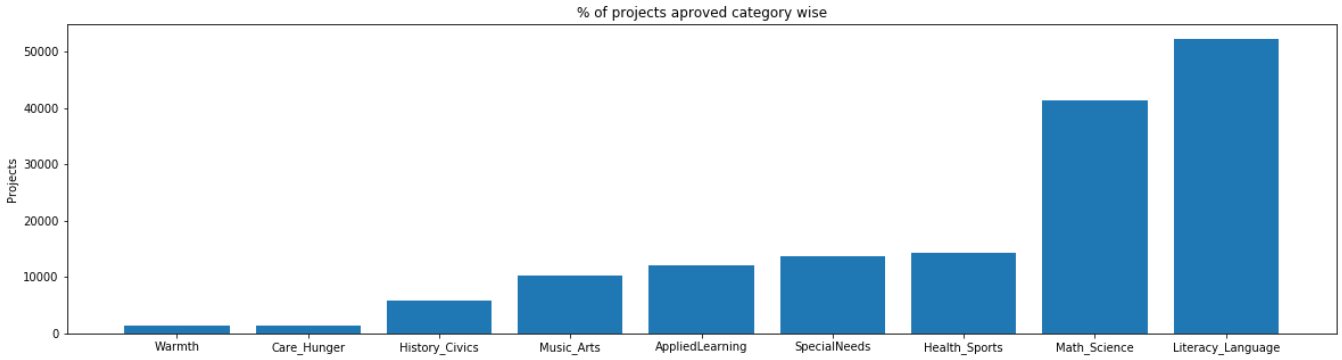
SUMMARY: Projects with warmth Care Hunger as categories have the highest approval rates

In [16]:

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

In [17]:

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


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

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



SUMMARY: projects under Literacy_language have the gihest approval rates

In [18]:

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

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

1.2.5 Univariate Analysis: project_subject_subcategories

In [19]:

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

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

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

In [20]:

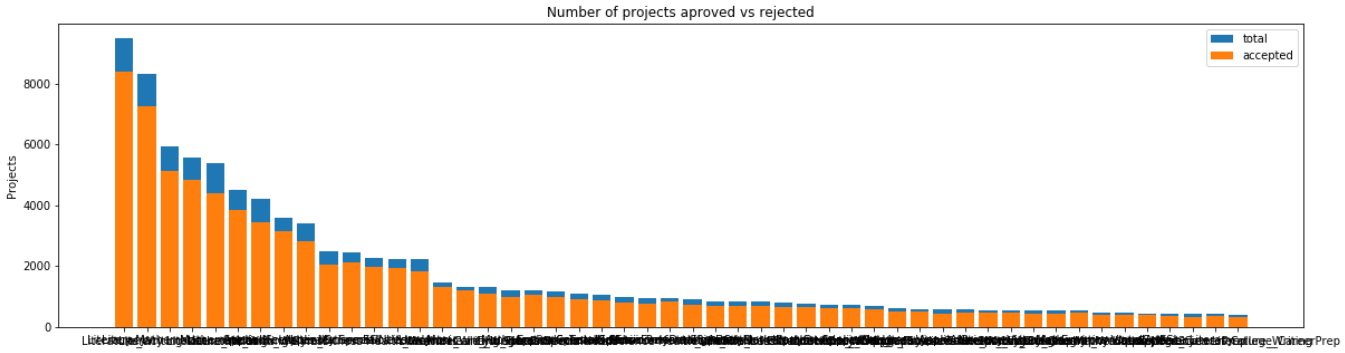
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[20]:

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

In [21]:

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




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

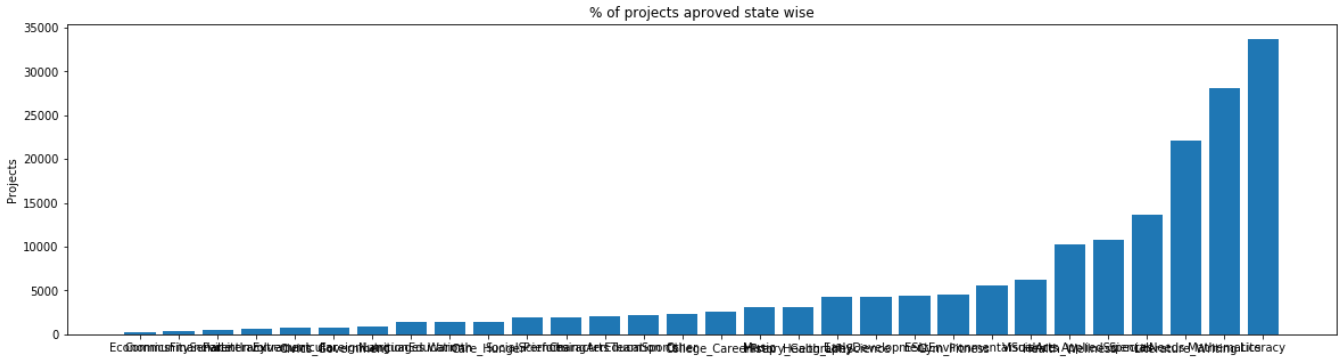
SUMMARY: All subcategories have an average approval rate higher than 80 %

In [22]:

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

In [23]:

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


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

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



In [24]:

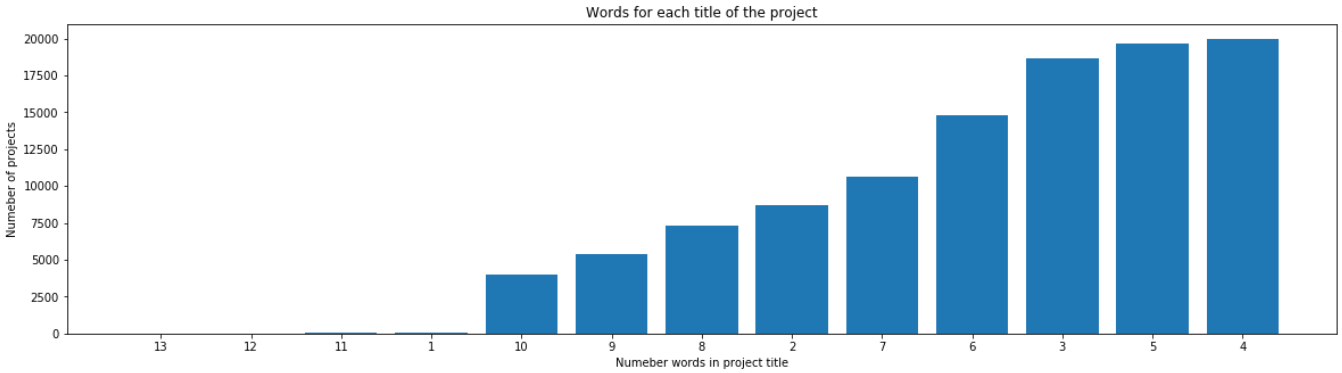
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
Economics            :       269
CommunityService     :       441
FinancialLiteracy    :       568
ParentInvolvement    :       677
Extracurricular      :       810
Civics_Government    :       815
ForeignLanguages     :       890
NutritionEducation   :      1355
Warmth               :      1388
Care_Hunger          :      1388
SocialSciences       :      1920
PerformingArts       :      1961
CharacterEducation   :      2065
TeamSports           :      2192
Other                :      2372
College_CareerPrep   :      2568
Music                :      3145
History_Geography    :      3171
Health_LifeScience   :      4235
EarlyDevelopment     :      4254
ESL                  :      4367
Gym_Fitness          :      4509
EnvironmentalScience :      5591
VisualArts           :      6278
Health_Wellness      :     10234
AppliedSciences      :     10816
SpecialNeeds         :     13642
Literature_Writing   :     22179
Mathematics          :     28074
Literacy             :     33700
```

1.2.6 Univariate Analysis: Text features (Title)

In [25]:

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.com/a/37483537/4084039
word_count = project_data['project_title'].str.split().apply(len).value_counts()
word_dict = dict(word_count)
word_dict = dict(sorted(word_dict.items(), key=lambda kv: kv[1]))


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

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



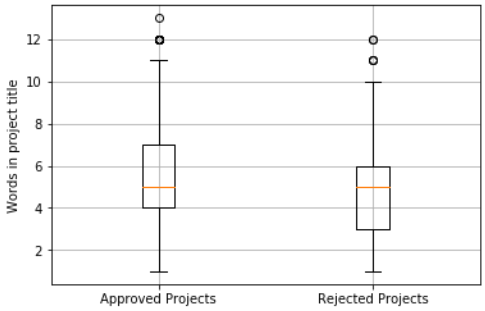
SUMMARY: Projects whose titles are concise and clear have the highest approval rates

In [26]:

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

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

In [27]:

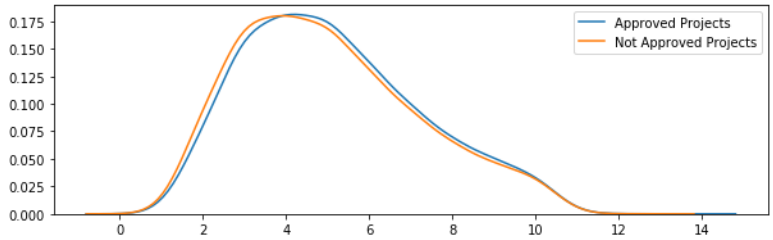
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_title_word_count, rejected_title_word_count])
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project title')
plt.grid()
plt.show()
```



SUMMARY: project_titles word count dosen't have a significant reason in the approval rates

In [28]:

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



SUMMARY: project_titles word count dosen't have a significant reason in the approval rates

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

In [29]:

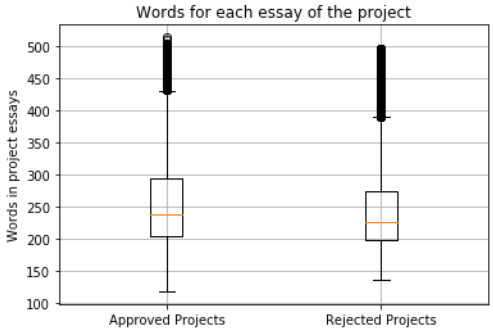
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [30]:

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

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

In [31]:

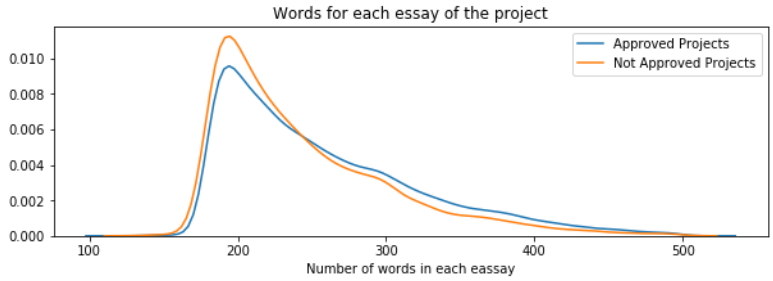
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_word_count, rejected_word_count])
plt.title('Words for each essay of the project')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Words in project essays')
plt.grid()
plt.show()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



SUMMARY: Number of word count in each essay dosen't have a significant reason in the approval rates

In [32]:

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



SUMMARY: Essays with lower word count tend to have higher approval rates

1.2.8 Univariate Analysis: Cost per project

In [33]:

```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[33]:

	id	description	quantity	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

In [34]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[34]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [35]:

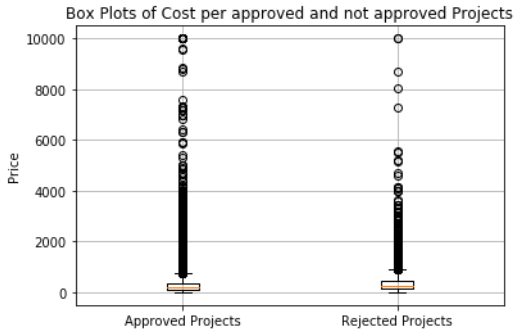
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [36]:

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

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

In [37]:

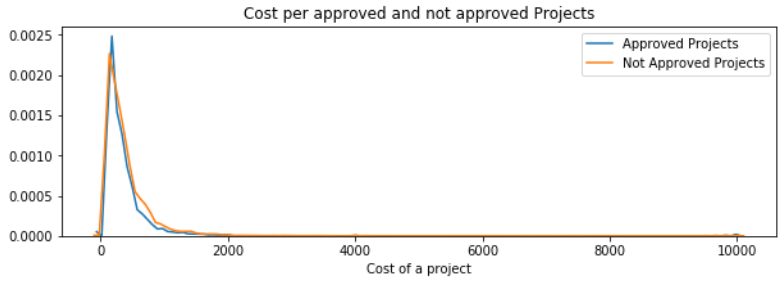
```
# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot([approved_price, rejected_price])
plt.title('Box Plots of Cost per approved and not approved Projects')
plt.xticks([1,2],('Approved Projects','Rejected Projects'))
plt.ylabel('Price')
plt.grid()
plt.show()
```



SUMMARY:Price does not play a significant role in the approval rates

In [38]:

```
plt.figure(figsize=(10,3))
sns.distplot(approved_price, hist=False, label="Approved Projects")
sns.distplot(rejected_price, hist=False, label="Not Approved Projects")
plt.title('Cost per approved and not approved Projects')
plt.xlabel('Cost of a project')
plt.legend()
plt.show()
```



SUMMARY:Price does not play a significant role in the approval rates

In [39]:

```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

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

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

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

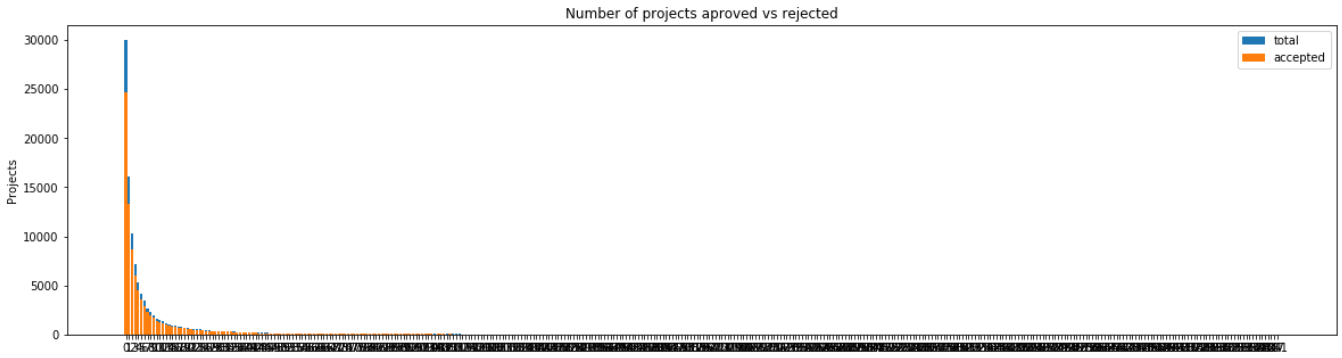
```
+------------+------------------+-----------------------+
| Percentile | Approved Projects | Not Approved Projects |
+------------+------------------+-----------------------+
|     0      |       0.66       |         1.97          |
|     5      |      13.59       |         41.9          |
|    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

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

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

In [40]:

```
univariate_barplots(project_data, 'teacher_number_of_previously_posted_projects', 'project_is_approved', False)
```




```
   teacher_number_of_previously_posted_projects  project_is_approved  total  \
0                                             0                24652  30014
1                                             1                13329  16058
2                                             2                 8705  10350
3                                             3                 5997   7110
4                                             4                 4452   5266

        Avg
0  0.821350
1  0.830054
2  0.841063
3  0.843460
4  0.845423
==================================================
     teacher_number_of_previously_posted_projects  project_is_approved  total  \
242                                           242                   1      1
268                                           270                   1      1
234                                           234                   1      1
335                                           347                   1      1
373                                           451                   1      1

     Avg
242  1.0
268  1.0
234  1.0
335  1.0
373  1.0
```

Summary : It seems like techers who are submitting projects for the first time have a higher acceptance rate

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 [41]:

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

Out[41]:

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

Function to check the presence of a number in a string

In [42]:

```
# Checking the presence of number in a string

def have_num(s):
    for i in s:
        if i.isnumeric():
            return 1
    return 0
```

In [43]:

```
have_num(' i have 10 apples ')
```

Out[43]:

```
1
```

In [44]:

```
have_num(' i have 11 apples !')
```

Out[44]:

```
1
```

In [45]:

```
have_num(' i have apples')
```

Out[45]:

```
0
```

Applying a function to all the rows of a specified column

In [46]:

```
### http://jonathansoma.com/lede/foundations/classes/pandas%20columns%20and%20functions/apply-a-function-to-every-row-in-a-pandas-dataframe/

# project_data['project_resource_summary'].apply(have_num)

presence_of_numbers = project_data['project_resource_summary'].apply(have_num)
presence_of_numbers.head(10)
```

Out[46]:

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
9    0
Name: project_resource_summary, dtype: int64
```

In [47]:

```
# project_data.loc[12, :]
 project_data.loc[12 , ['project_resource_summary']]
```

Out[47]:

```
project_resource_summary    My students need 3D and 4D life science activi...
Name: 12, dtype: object
```

Adding the new column into the original_dataframe

In [48]:

```
project_data['presence_of_numbers'] = presence_of_numbers
project_data.head(2)
```

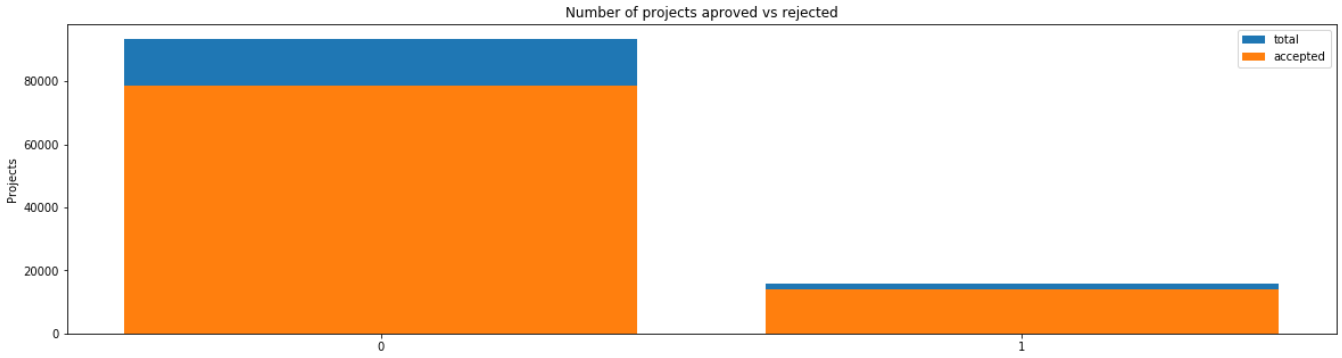
Out[48]:

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

2 rows × 21 columns

In [49]:

```
univariate_barplots(project_data, 'presence_of_numbers', 'project_is_approved', top=False)
```




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

Summary : The presence of numbers in resource summary dosent have much significance in approval rates. The rate of increase in approval is about 5%

Dealing with NAN variables in teacher_prefix column

In [50]:

```
project_data.dropna(subset = ['teacher_prefix'] , how = 'any' , inplace=True)
```

In [51]:

```
project_data.shape
```

Out[51]:

```
(109245, 21)
```

1.3 Text preprocessing

1.3.1 Essay Text

In [52]:

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

Out[52]:

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

2 rows × 21 columns

In [53]:

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

```
My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented i n our English Learner program with students at every level of mastery.  We also have over 40 countries represented with the familie s within our school.  Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home that begs for more resources.  Many times our parents are learning to read and speak English along side of t heir children.  Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English langu age even if no one at home is able to assist.  All families with students within the Level 1 proficiency status, will be a offered to be a part of this program.  These educational videos will be specially chosen by the English Learner Teacher and will be sent ho me regularly to watch.  The videos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year.  The plan is to use these videos and educationa l dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our sch ool, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school ha s a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the bea utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity.My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged c hairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be use d by the students who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhenever asked w hat the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of stu dents to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same t ime. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can 't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look fo rward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these r esources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to hel p create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very imp ortant in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade.  This kind gesture will set the tone before even the first d ay of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan
==================================================
My wonderful students are 3, 4, and 5 years old.  We are located in a small town outside of Charlotte, NC.  All of my 22 students a re children of school district employees.\r\nMy students are bright, energetic, and they love to learn!  They love hands-on activit ies that get them moving.  Like most preschoolers, they enjoy music and creating different things. \r\nAll of my students come from wonderful families that are very supportive of our classroom.  Our parents enjoy watching their children's growth as much as we do! These materials will help me teach my students all about the life cycle of a butterfly.  We will watch as the Painted Lady caterpil lars grow bigger and build their chrysalis.  After a few weeks they will emerge from the chrysalis as beautiful butterflies!  We al ready have a net for the chrysalises, but we still need the caterpillars and feeding station.\r\nThis will be an unforgettable expe rience for my students.  My student absolutely love hands-on materials.  They learn so much from getting to touch and manipulate di fferent things.  The supporting materials I have selected will help my students understand the life cycle through exploration.nanna n
==================================================
The students in my classroom are learners, readers, writers, explorers, scientists, and mathematicians! The potential in these firs t graders is endless! Each day they come in grinning from ear-to-ear and ready to learn more. \r\nI choose curriculum that is real and relevant to the students, but it will also prepare them for their futures. These kids are encouraged to investigate concepts th at are exciting for them and I hope we can keep this momentum going! These kids deserve the best, please help me give that to them! Thank you! :)These kits include a wide variety of science, technology, engineering, and mechanics for my students to dive into at t he beginning of the year. I want them to hit the ground running this upcoming year and these kits always encourage high interest.\r \nWho wouldn't want to build their own roller coaster, design a car, or even think critically to make a bean bag bounce as far as i t can go?? These kits will also shows students potential careers that they may have never heard of before!\r\nAny donations would b e greatly appreciated and my students will know exactly who to thank for them!nannan
==================================================
```

In [54]:

```
# 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 [55]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My wonderful students are 3, 4, and 5 years old.  We are located in a small town outside of Charlotte, NC.  All of my 22 students a
re children of school district employees.\r\nMy students are bright, energetic, and they love to learn!  They love hands-on activit
ies that get them moving.  Like most preschoolers, they enjoy music and creating different things. \r\nAll of my students come from
wonderful families that are very supportive of our classroom.  Our parents enjoy watching their children is growth as much as we do
!These materials will help me teach my students all about the life cycle of a butterfly.  We will watch as the Painted Lady caterpi
llars grow bigger and build their chrysalis.  After a few weeks they will emerge from the chrysalis as beautiful butterflies!  We a
lready have a net for the chrysalises, but we still need the caterpillars and feeding station.\r\nThis will be an unforgettable exp
erience for my students.  My student absolutely love hands-on materials.  They learn so much from getting to touch and manipulate d
ifferent things.  The supporting materials I have selected will help my students understand the life cycle through exploration.nann
an
==================================================
```

In [56]:

```
# \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)
```

```
My wonderful students are 3, 4, and 5 years old.  We are located in a small town outside of Charlotte, NC.  All of my 22 students a
re children of school district employees.  My students are bright, energetic, and they love to learn!  They love hands-on activitie
s that get them moving.  Like most preschoolers, they enjoy music and creating different things.  All of my students come from won
derful families that are very supportive of our classroom.  Our parents enjoy watching their children is growth as much as we do!Th
ese materials will help me teach my students all about the life cycle of a butterfly.  We will watch as the Painted Lady caterpilla
rs grow bigger and build their chrysalis.  After a few weeks they will emerge from the chrysalis as beautiful butterflies!  We alre
ady have a net for the chrysalises, but we still need the caterpillars and feeding station.  This will be an unforgettable experien
ce for my students.  My student absolutely love hands-on materials.  They learn so much from getting to touch and manipulate differ
ent things.  The supporting materials I have selected will help my students understand the life cycle through exploration.nannan
```

In [57]:

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

```
My wonderful students are 3 4 and 5 years old We are located in a small town outside of Charlotte NC All of my 22 students are chil
dren of school district employees My students are bright energetic and they love to learn They love hands on activities that get th
em moving Like most preschoolers they enjoy music and creating different things All of my students come from wonderful families tha
t are very supportive of our classroom Our parents enjoy watching their children is growth as much as we do These materials will he
lp me teach my students all about the life cycle of a butterfly We will watch as the Painted Lady caterpillars grow bigger and buil
d their chrysalis After a few weeks they will emerge from the chrysalis as beautiful butterflies We already have a net for the chry
salises but we still need the caterpillars and feeding station This will be an unforgettable experience for my students My student
absolutely love hands on materials They learn so much from getting to touch and manipulate different things The supporting material
s I have selected will help my students understand the life cycle through exploration nannan
```

In [58]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [59]:

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

```
100%|██████████| 109245/109245 [01:07<00:00, 1625.72it/s]
```

In [60]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[60]:

```
'my wonderful students 3 4 5 years old we located small town outside charlotte nc all 22 students children school district employee
s my students bright energetic love learn they love hands activities get moving like preschoolers enjoy music creating different th
ings all students come wonderful families supportive classroom our parents enjoy watching children growth much these materials help
teach students life cycle butterfly we watch painted lady caterpillars grow bigger build chrysalis after weeks emerge chrysalis bea
utiful butterflies we already net chrysalises still need caterpillars feeding station this unforgettable experience students my stu
dent absolutely love hands materials they learn much getting touch manipulate different things the supporting materials i selected
help students understand life cycle exploration nannan'
```

1.3.2 Project title Text

In [61]:

```
# similarly you can preprocess the titles also
```

In [62]:

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

Out[62]:

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

2 rows × 21 columns

In [63]:

```
# Printing some random titles

print(project_data['project_title'].values[0])
print('='*50)
print(project_data['project_title'].values[1500])
print('='*50)
print(project_data['project_title'].values[2500])
print('='*50)
print(project_data['project_title'].values[4500])
print('='*50)
print(project_data['project_title'].values[9500])
print('='*50)
print(project_data['project_title'].values[19500])
print('='*50)
print(project_data['project_title'].values[7000])
print('='*50)
```

```
Educational Support for English Learners at Home
==================================================
Listening Center
==================================================
Food for the Brain!
==================================================
Ohana Means Family...We Support Each Other!
==================================================
Opening Young Eyes to New Books!
==================================================
Growing Mathematical Thinkers
==================================================
Finding Fitness
==================================================
```

In [64]:

```
# Testing out the decontracting

sent = decontracted(project_data['project_title'].values[9500])
sent
```

Out[64]:

```
'Opening Young Eyes to New Books!'
```

In [65]:

```
# Testing out the removal of line breaks

sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
Opening Young Eyes to New Books!
```

In [66]:

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

```
100%|██████████| 109245/109245 [00:02<00:00, 38002.94it/s]
```

In [67]:

```
print('Before preprocessing  :-------------------------',project_data['project_title'].values[4500])
# After Preprocessing
print('After preprocessing -------------------------',preprocessed_titles[4500])
```

```
Before preprocessing  :------------------------- Ohana Means Family...We Support Each Other!
After preprocessing ------------------------- ohana means family we support each other
```

1. 4 Preparing data for models

In [68]:

```
project_data.columns
```

Out[68]:

```
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',
       'presence_of_numbers'],
      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
```

1.4.1 Vectorizing Categorical data

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

In [69]:

```
# 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', 'Math_Science', 'Lite
racy_Language']
Shape of matrix after one hot encodig  (109245, 9)
```

In [70]:

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


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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguag
es', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other'
, 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Environmenta
lScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109245, 30)
```

In [71]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category also
```

For School_state

In [72]:

```
# For school_state

vectorizer = CountVectorizer( lowercase=False, binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


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

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME'
, 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX
', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
Shape of matrix after one hot encodig  (109245, 51)
```

For project_grade_category

In [73]:

```
# For project grade cetegory

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


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

```
['12', 'Grades', 'PreK']
Shape of matrix after one hot encodig  (109245, 3)
```

In [74]:

```
project_data['project_grade_category'].value_counts()
```

Out[74]:

```
Grades PreK-2    44225
Grades 3-5       37135
Grades 6-8       16923
Grades 9-12      10962
Name: project_grade_category, dtype: int64
```

For teacher_prefix

- The main issue is the presence of NAN values in the teacher_prefix

how to replace nan values in a column in pandas : https://www.youtube.com/watch?v=fCMrO_VzeL8

Checking for nan/null values

In [140]:

```
project_data.isnull().sum()
```

Out[140]:

```
Unnamed: 0                                         0
id                                                 0
teacher_id                                         0
teacher_prefix                                     0
school_state                                       0
project_submitted_datetime                         0
project_grade_category                             0
project_title                                      0
project_essay_1                                    0
project_essay_2                                    0
project_essay_3                               105488
project_essay_4                               105488
project_resource_summary                           0
teacher_number_of_previously_posted_projects       0
project_is_approved                                0
clean_categories                                   0
clean_subcategories                                0
essay                                              0
price                                              0
quantity                                           0
presence_of_numbers                                0
dtype: int64
```

These are the 3 columns that have teacher prefix as NAN

- Observation : All the three have approved projects, even though their teacher prefixes are missing

In [141]:

```
project_data['teacher_prefix'].value_counts()
```

Out[141]:

```
Mrs.       57269
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
Name: teacher_prefix, dtype: int64
```

In [142]:

```
project_data[project_data.teacher_prefix.isnull()]
```

Out[142]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_essay_1	project_essay_2	...

0 rows × 21 columns

There are 2 ways as to how we proceed.

- a) We can drop the NAN values
- b) We can replace them
- Since only 3 rows have teacher_prefix as NAN values ,we can remove them without any majore change in the data

Testing out the various ways of replacing the NAN values

project_data.dropna(how = 'any').shape # Not a feasible methodproject_data.shapeproject_data.dropna(how = 'all').shape # No change and thus we use the subset methodproject_data.dropna(subset = ['teacher_prefix'] , how = 'any').shape # This works project_data.dropna(subset = ['teacher_prefix'] , how = 'all').shape # This works as well

Removing the 3 rows in teacher_prefix which have NAN values

In [ ]:

```
project_data.dropna(subset = ['teacher_prefix'] , how = 'any' , inplace=True)
```

In [ ]:

```
project_data.shape
```

Testing out if dropna worked correctly

In [143]:

```
project_data[project_data.teacher_prefix.isnull()]
```

Out[143]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_essay_1	project_essay_2	...

0 rows × 21 columns

Thus, we can proceed to apply countvectorizer on teacher_prefix without any issue now

In [75]:

```
# for teacher_prefix

vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())


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

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (109245, 5)
```

1.4.2 Vectorizing Text data

1.4.2.1 Bag of words

In [76]:

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

```
Shape of matrix after one hot encodig  (109245, 16623)
```

1.4.2.2 Bag of Words on `project_title`

In [77]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

In [78]:

```
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109245, 16623)
```

1.4.2.3 TFIDF vectorizer

In [79]:

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

```
Shape of matrix after one hot encodig  (109245, 16623)
```

1.4.2.4 TFIDF Vectorizer on `project_title`

In [80]:

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

```
Shape of matrix after one hot encodig  (109245, 16623)
```

In [81]:

```
project_data.head(1)
```

Out[81]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educational Support for English Learners at Home	My stu Englis that a

1 rows × 21 columns

1.4.2.5 Using Pretrained Models: Avg W2V

In [82]:

```
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')
```

```
Loading Glove Model
1917495it [04:49, 6616.53it/s]
Done. 1917495  words loaded!
```

In [83]:

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

for i in preprocessed_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))
```

```
all the words in the coupus 17013963
the unique words in the coupus 58966
The number of words that are present in both glove vectors and our coupus 51501 ( 87.34 %)
word 2 vec length 51501
```

In [84]:

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

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

In [ ]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/

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


'''
```

In [85]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [86]:

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

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

```
100%|██████████████████████████████████████████████████████████| 109245/109245 [00:47<00:00, 2276.56it/s]
```

```
109245
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

In [87]:

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

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

```
100%|██████████| 109245/109245 [00:02<00:00, 52145.24it/s]
```

```
109245
300
```

1.4.2.7 Using Pretrained Models: TFIDF weighted W2V

In [88]:

```
# 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 [89]:

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

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

```
100%|██████████| 109245/109245 [04:14<00:00, 429.35it/s]
```

```
109245
300
```

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

tfidf_w2v for preprocessed_titles

In [90]:

```
tfidf_w2v_vectors_titles = []; # the avg-w2v for each preporcessed_titles is stored in this list
for sentence in tqdm(preprocessed_titles): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles.append(vector)

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

```
100%|██████████| 109245/109245 [00:04<00:00, 25787.20it/s]
```

```
109245
300
```

1.4.3 Vectorizing Numerical features

a) Price Column

In [91]:

```
# 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 : 298.1152448166964, Standard deviation : 367.49642545627506
```

In [92]:

```
price_standardized
```

Out[92]:

```
array([[-0.39052147],
       [ 0.00240752],
       [ 0.5952024 ],
       ...,
       [-0.1582471 ],
       [-0.61242839],
       [-0.51215531]])
```

b) teacher_no_of_previously_posted_projects

In [93]:

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

# Now standardize the data with above mean and variance.
teacher_number_of_previously_posted_projects_standardized = no_of_projects_scalar.transform(project_data['teacher_number_of_previou
sly_posted_projects'].values.reshape(-1, 1))
```

```
C:\Users\Rahul\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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

```

```
Mean : 11.153462401025218, Standard deviation : 367.49642545627506
```

```
C:\Users\Rahul\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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

```

In [94]:

```
teacher_number_of_previously_posted_projects_standardized
```

Out[94]:

```
array([[-0.40153083],
       [-0.14952695],
       [-0.36553028],
       ...,
       [-0.29352917],
       [-0.40153083],
       [-0.40153083]])
```

1.4.4 Merging all the above features

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

```
(109245, 9)
(109245, 30)
(109245, 16623)
(109245, 1)
(109245, 1)
(109245, 5)
```

In [96]:

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

Out[96]:

```
'\n# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039\nfrom scipy.sparse import hstack\n# with the same hsta
ck function we are concatinating a sparse matrix and a dense matirx :)\nX = hstack((categories_one_hot, sub_categories_one_hot, tex
t_bow, price_standardized))\nX.shape\n'
```

In [ ]:

```
'''

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

'''
```

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.

User Defined Features list

clean_categories = categories_one_hot,

clean_subcategories = sub_categories_one_hot,

school_state = school_state_one_hot ,

techer_prefix = teacher_prefix_one_hot

project_title (BOW) = text_bow

project_title (TFIDF) = title_tfidf

project_title (Average_word2_vec) : avg_w2v_vectors_titles

project_title(TFIDF_W2V) : tfidf_w2v_vectors_titles

teacher_number_of_previously_posted_projects : teacher_number_of_previously_posted_projects_standardized
previously_posted_projects_standardized_3000

price = price_standardized

Converting the sparse (X) matrix to dense matrix

from scipy.sparse import csr_matrix

csr_matrix.todense(X)

scipy.sparse.csr_matrix.todense(X)

In [ ]:

```
X
```

For ease of calculation we are only considering 3000 datapoints

In [97]:

```
categories_one_hot_3000 = categories_one_hot[:2999]
sub_categories_one_hot_3000 = sub_categories_one_hot[:2999]
school_state_one_hot_3000 = school_state_one_hot[:2999]
teacher_prefix_one_hot_3000 = teacher_prefix_one_hot[:2999]
text_bow_3000 = text_bow[:2999]
text_tfidf_3000 = text_tfidf[:2999]
teacher_prefix_one_hot_3000 = teacher_prefix_one_hot[:2999]
avg_w2v_vectors_titles_3000 = avg_w2v_vectors_titles[:2999]
tfidf_w2v_vectors_titles_3000 = tfidf_w2v_vectors_titles[:2999]
previously_posted_projects_standardized_3000 = teacher_number_of_previously_posted_projects_standardized[:2999]
price_standardized_3000 = price_standardized[:2999]
title_bow_3000 = title_bow[:2999]
title_tfidf_3000 = title_tfidf[:2999]
avg_w2v_vectors_titles_3000 = avg_w2v_vectors_titles[:2999]
tfidf_w2v_vectors_titles_3000 = tfidf_w2v_vectors_titles[:2999]
```

In [98]:

```
print(categories_one_hot_3000.shape)
print(sub_categories_one_hot_3000.shape)
print(price_standardized_3000.shape)
print(previously_posted_projects_standardized_3000.shape)
print(teacher_prefix_one_hot_3000.shape)
print(school_state_one_hot_3000.shape)
print(title_bow_3000.shape)
print(title_tfidf_3000.shape)
print(len(avg_w2v_vectors_titles_3000))
print(len(tfidf_w2v_vectors_titles_3000))
```

```
(2999, 9)
(2999, 30)
(2999, 1)
(2999, 1)
(2999, 5)
(2999, 51)
(2999, 3329)
(2999, 3329)
2999
2999
```

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_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

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

In [ ]:

```
# 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
```

Taking project_is_approved column as the labels

- Considering the first 3000 datapoints

In [102]:

```
labels_3000 = project_data.project_is_approved[:2999]
labels_3000.value_counts()
```

Out[102]:

```
1    2546
0     453
Name: project_is_approved, dtype: int64
```

In [103]:

```
labels_3000.shape
```

Out[103]:

```
(2999,)
```

In [104]:

```
print(type(labels_3000))
```

```
<class 'pandas.core.series.Series'>
```

Steps before standardizing the labels column

a) Converting the panda series into a numpy array

In [108]:

```
# how to transform pandas series into numpy array : https://stackoverflow.com/questions/44238796/convert-panda-series-into-numpy-ar
ray

labels_array= labels_3000.as_matrix(columns=None)
print(type(labels_array))
print(labels_array.shape)
```

```
<class 'numpy.ndarray'>
(2999,)
```

b) Reshaping the labels

In [110]:

```
# Reshaping the labels

labels_array_new = labels_array.reshape(1, -1)
```

Standardizing the labels

In [111]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler(with_mean = False)
scaled_label_3000 = scaler.fit_transform(labels_array_new)
```

```
C:\Users\Rahul\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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

C:\Users\Rahul\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

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

Applying TSNE (BOW)

In [169]:

```
from sklearn.manifold import TSNE
```

In [170]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot_3000, sub_categories_one_hot_3000, school_state_one_hot_3000,teacher_prefix_one_hot_3000, title_bow_
3000, price_standardized_3000,previously_posted_projects_standardized_3000))
X.shape
```

Out[170]:

```
(2999, 3426)
```

In [171]:

```
print(type(X))
```

```
<class 'scipy.sparse.coo.coo_matrix'>
```

Converting a sparse matrix to dense matrix

how to convert a sparse matrix into a dense matrix in python : https://stackoverflow.com/questions/26576524/how-do-i-transform-a-scipy-sparse-matrix-to-a-numpy-matrix/26577144

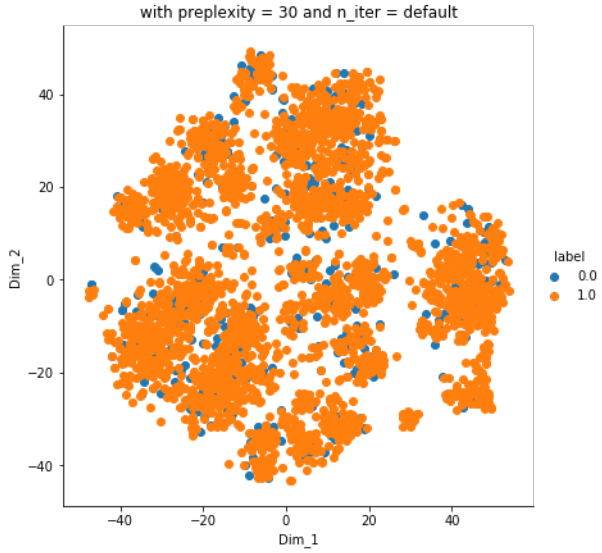
In [172]:

```
X_final = X.todense()
print(type(X_final))
```

```
<class 'numpy.matrixlib.defmatrix.matrix'>
```

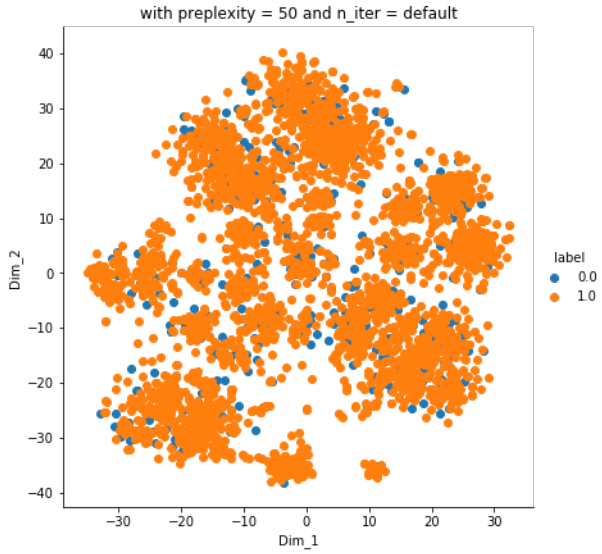
In [137]:

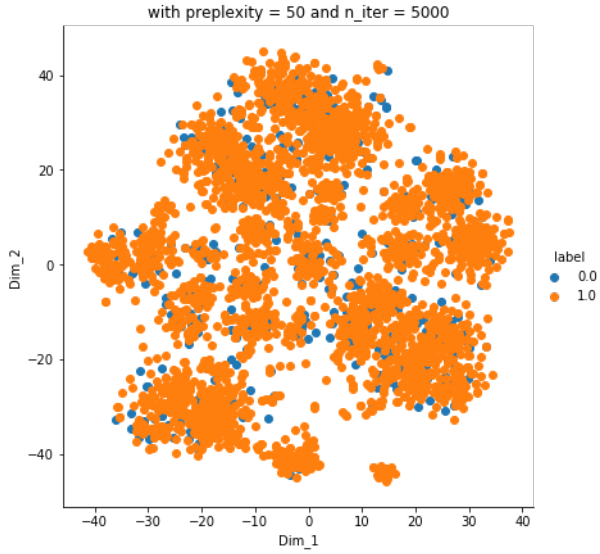
```
tsne = TSNE(n_components = 2, perplexity = 30 , random_state = 0)
tsne_bow = tsne.fit_transform(X_final)

tsne_bow = np.vstack((tsne_bow.T ,scaled_label_3000)).T
tsne_df_bow = pd.DataFrame(data = tsne_bow ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_bow , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 30 and n_iter = default')
plt.show()
```



In [118]:

```
tsne = TSNE(n_components = 2, perplexity = 50 , random_state = 0)
tsne_bow = tsne.fit_transform(X_final)

tsne_bow = np.vstack((tsne_bow.T ,scaled_label_3000)).T
tsne_df_bow = pd.DataFrame(data = tsne_bow ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_bow , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 50 and n_iter = default')
plt.show()
```





In [ ]:

```
```

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

In [ ]:

```
# 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
```

In [187]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot_3000, sub_categories_one_hot_3000, school_state_one_hot_3000,teacher_prefix_one_hot_3000, title_tfidf_3000, price_standardized_3000,previously_posted_projects_standardized_3000))
X.shape
```

Out[187]:

```
(2999, 3426)
```

In [188]:

```
print(type(X))
```

```
<class 'scipy.sparse.coo.coo_matrix'>
```

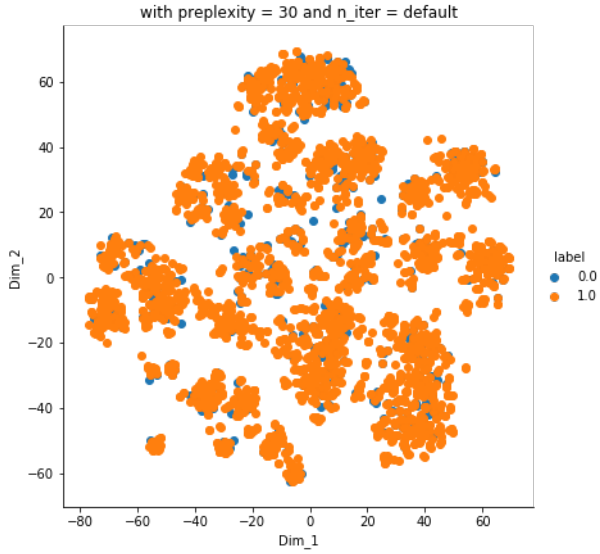
In [189]:

```
X_final = X.todense()
print(type(X_final))
```

```
<class 'numpy.matrixlib.defmatrix.matrix'>
```

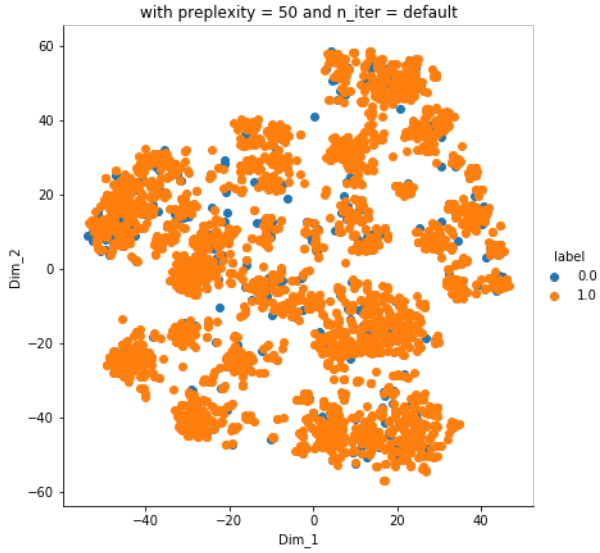
In [123]:

```
tsne = TSNE(n_components = 2, perplexity = 30 , random_state = 0)
tsne_tfidf = tsne.fit_transform(X_final)

tsne_tfidf = np.vstack((tsne_tfidf.T ,scaled_label_3000)).T
tsne_df_tfidf = pd.DataFrame(data = tsne_tfidf ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_tfidf , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 30 and n_iter = default')
plt.show()
```



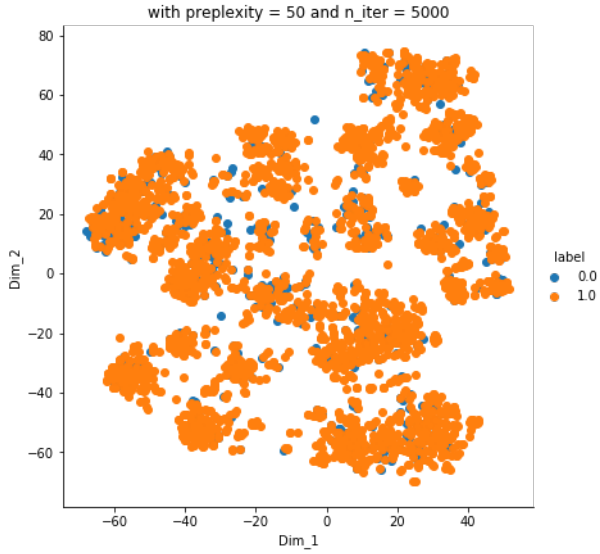
In [190]:

```
tsne = TSNE(n_components = 2, perplexity = 50 , random_state = 0)
tsne_tfidf = tsne.fit_transform(X_final)

tsne_tfidf = np.vstack((tsne_tfidf.T ,scaled_label_3000)).T
tsne_df_tfidf = pd.DataFrame(data = tsne_tfidf ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_tfidf , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 50 and n_iter = default')
plt.show()
```



In [178]:

```
# With perplexity

tsne = TSNE(n_components = 2, perplexity = 50 , n_iter = 5000,  random_state = 0)
tsne_tfidf = tsne.fit_transform(X_final)

tsne_tfidf = np.vstack((tsne_tfidf.T ,scaled_label_3000)).T
tsne_df_tfidf = pd.DataFrame(data = tsne_tfidf ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_tfidf , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 50 and n_iter = 5000')
plt.show()
```



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

In [ ]:

```
# 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
```

In [179]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot_3000, sub_categories_one_hot_3000, school_state_one_hot_3000,teacher_prefix_one_hot_3000, avg_w2v_ve
ctors_titles_3000, price_standardized_3000,previously_posted_projects_standardized_3000))
X.shape
```

Out[179]:

```
(2999, 397)
```

In [180]:

```
print(type(X))
```

```
<class 'scipy.sparse.coo.coo_matrix'>
```

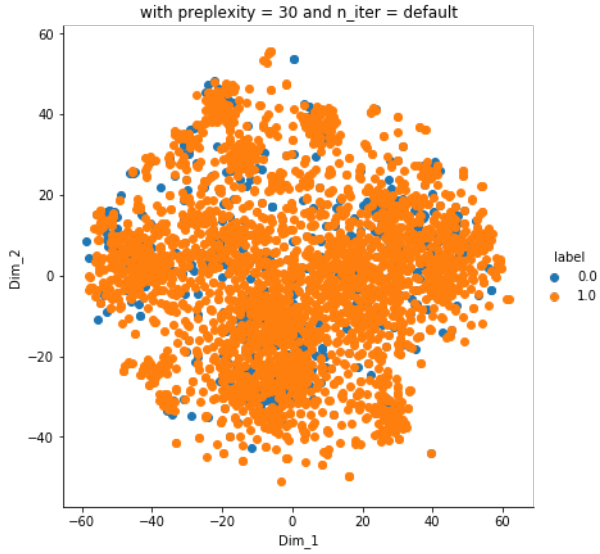
In [181]:

```
X_final = X.todense()
print(type(X_final))
```

```
<class 'numpy.matrixlib.defmatrix.matrix'>
```

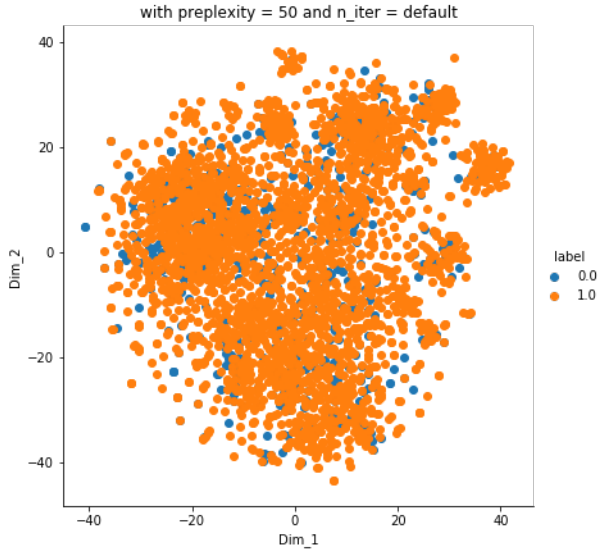
In [131]:

```
tsne = TSNE(n_components = 2, perplexity = 30 , random_state = 0)
tsne_avg_w2v = tsne.fit_transform(X_final)

tsne_avg_w2v = np.vstack((tsne_avg_w2v.T ,scaled_label_3000)).T
tsne_df_avg_w2v = pd.DataFrame(data = tsne_avg_w2v ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_avg_w2v , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 30 and n_iter = default')
plt.show()
```



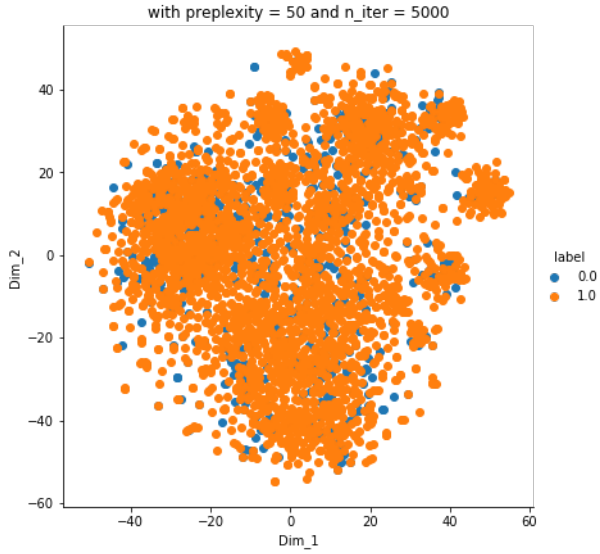
In [182]:

```
tsne = TSNE(n_components = 2, perplexity = 50 , random_state = 0)
tsne_avg_w2v = tsne.fit_transform(X_final)

tsne_avg_w2v = np.vstack((tsne_avg_w2v.T ,scaled_label_3000)).T
tsne_df_avg_w2v = pd.DataFrame(data = tsne_avg_w2v ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_avg_w2v , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 50 and n_iter = default')
plt.show()
```



In [183]:

```
tsne = TSNE(n_components = 2, perplexity = 50 ,n_iter = 5000, random_state = 0)
tsne_avg_w2v = tsne.fit_transform(X_final)

tsne_avg_w2v = np.vstack((tsne_avg_w2v.T ,scaled_label_3000)).T
tsne_df_avg_w2v = pd.DataFrame(data = tsne_avg_w2v ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_avg_w2v , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 50 and n_iter = 5000')
plt.show()
```



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

In [ ]:

```
# 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
```

In [132]:

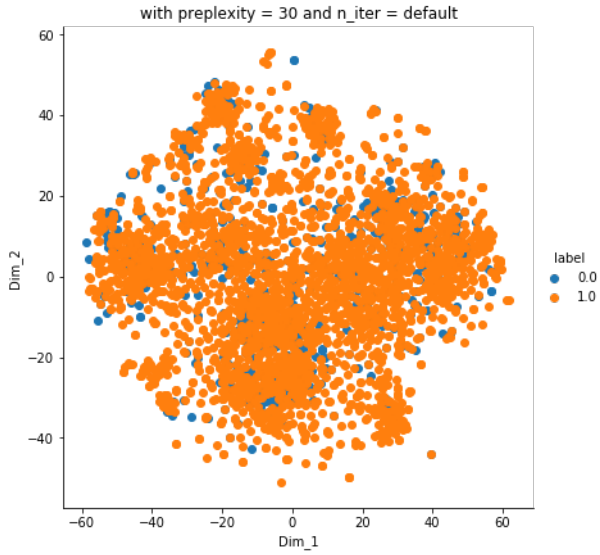
```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot_3000, sub_categories_one_hot_3000, school_state_one_hot_3000,teacher_prefix_one_hot_3000, tfidf_w2v_
vectors_titles_3000, price_standardized_3000,previously_posted_projects_standardized_3000))
X.shape
```

Out[132]:

```
(2999, 397)
```

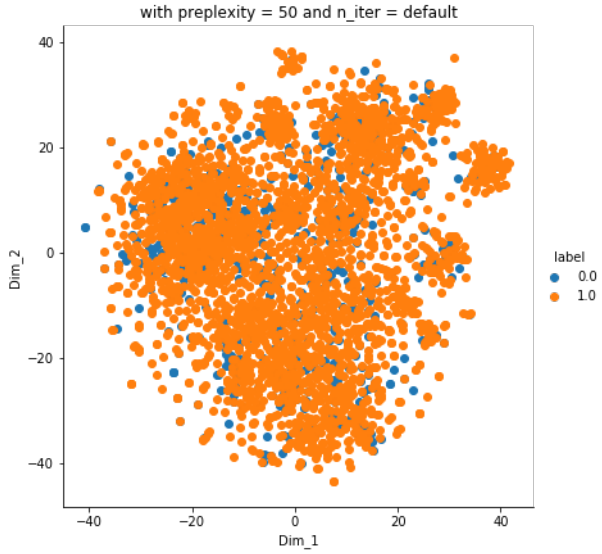
In [133]:

```
tsne = TSNE(n_components = 2, perplexity = 30 , random_state = 0)
tsne_tfidf_w2v = tsne.fit_transform(X_final)

tsne_tfidf_w2v = np.vstack((tsne_tfidf_w2v.T ,scaled_label_3000)).T
tsne_df_tfidf_w2v = pd.DataFrame(data = tsne_tfidf_w2v ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_tfidf_w2v , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 30 and n_iter = default')
plt.show()
```



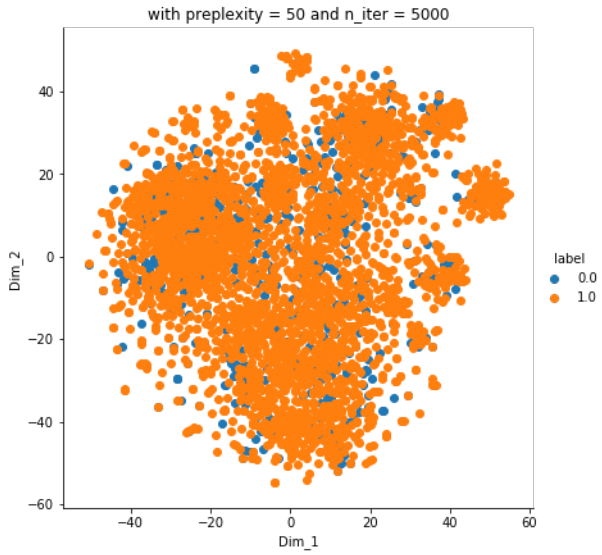
In [184]:

```
tsne = TSNE(n_components = 2, perplexity = 50 , random_state = 0)
tsne_tfidf_w2v = tsne.fit_transform(X_final)

tsne_tfidf_w2v = np.vstack((tsne_tfidf_w2v.T ,scaled_label_3000)).T
tsne_df_tfidf_w2v = pd.DataFrame(data = tsne_tfidf_w2v ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_tfidf_w2v , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 50 and n_iter = default')
plt.show()
```



In [185]:

```
tsne = TSNE(n_components = 2, perplexity = 50 ,n_iter = 5000, random_state = 0)
tsne_tfidf_w2v = tsne.fit_transform(X_final)

tsne_tfidf_w2v = np.vstack((tsne_tfidf_w2v.T ,scaled_label_3000)).T
tsne_df_tfidf_w2v = pd.DataFrame(data = tsne_tfidf_w2v ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_tfidf_w2v , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 50 and n_iter = 5000')
plt.show()
```



Combining all the features

- (BOW_titles, tfidf_titles, avg_w2v_titles and tfid_avg_w2c)

In [192]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot_3000, sub_categories_one_hot_3000, school_state_one_hot_3000,teacher_prefix_one_hot_3000, tfidf_w2v_
vectors_titles_3000, avg_w2v_vectors_titles_3000, title_tfidf_3000,title_bow_3000,price_standardized_3000,previously_posted_project
s_standardized_3000))
X.shape
```

Out[192]:

```
(2999, 7355)
```

In [193]:

```
print(type(X))
```

```
<class 'scipy.sparse.coo.coo_matrix'>
```

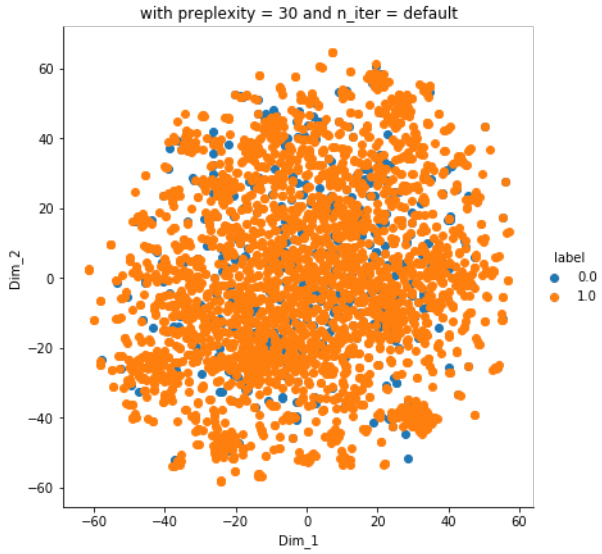
In [194]:

```
X_final = X.todense()
print(type(X_final))
```

```
<class 'numpy.matrixlib.defmatrix.matrix'>
```

In [195]:

```
tsne = TSNE(n_components = 2, perplexity = 30 , random_state = 0)
tsne_summary = tsne.fit_transform(X_final)

tsne_summary = np.vstack((tsne_summary.T ,scaled_label_3000)).T
tsne_df_summary = pd.DataFrame(data = tsne_summary ,columns = ('Dim_1', 'Dim_2','label'))

sns.FacetGrid(tsne_df_summary , hue = 'label' , height = 6).map(plt.scatter,'Dim_1', 'Dim_2').add_legend()
plt.title('with preplexity = 30 and n_iter = default')
plt.show()
```



2.5 Summary

TFIDF seems to have the best saperation for saperating the approved projects and unapproved projects

Since , the number of points selected is too small, that maybe a reason as to why the saperation is not distinct

In [ ]:

```
```

In [ ]:

```
```