In [0]:

```

```

In [1]:

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

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 (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **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.`

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_4:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [3]:

```
%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

!pip install chart_studio
import chart_studio.plotly as py
import plotly.graph_objs as go
import plotly.offline as offline
offline.init_notebook_mode()
from collections import Counter
```

```
Output hidden; open in https://colab.research.google.com to view.
```

1.1 Reading Data

In [0]:

```
project_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/train_data.csv
')
resource_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/resources.csv
')
```

In [5]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'schoo
l_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 [6]:
```
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[6]:

	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 [0]:
```
project_data=project_data.dropna(subset=['teacher_prefix'])
```

In [8]:
```
print("Number of data points in train data", project_data.shape)
print(project_data.columns.values)
project_data.head(2)
```

```
Number of data points in train data (109245, 17)
['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

Out[8]:

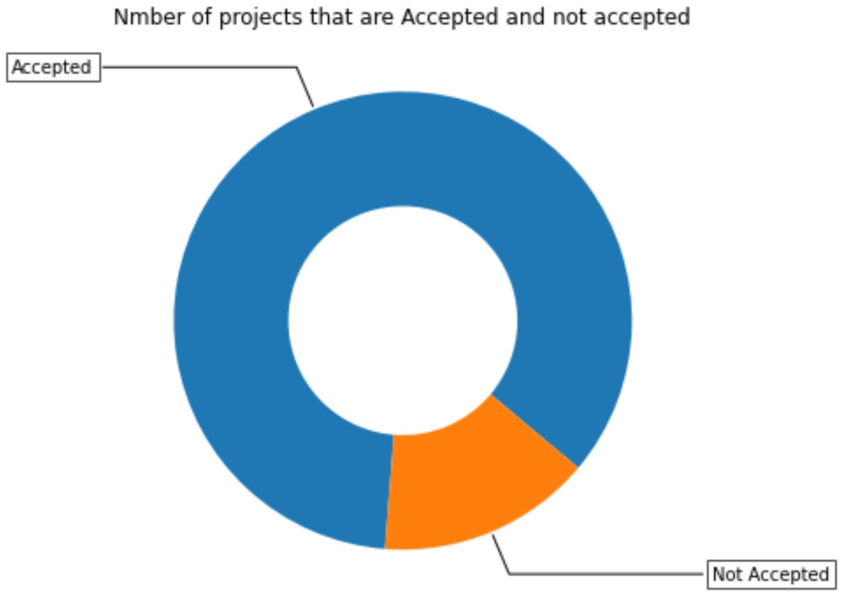
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:2

Data Analysis

In [9]:

```
# 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)#,labels=["Accepted","Not Accepted"])

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), 1.2*y),
                 horizontalalignment=horizontalalignment, **kw)

ax.set_title("Nmber of projects that are Accepted and not accepted")
#plt.legend()
plt.show()
```

```
Number of projects thar are approved for funding  92703 , ( 84.85788823287108 %)
Number of projects thar are not approved for funding  16542 , ( 15.14211176712893 %)
```



1 Univariate Analysis: School State

In [13]:
```
# 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')
```

In [14]:

```
# 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 [0]:

```
#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 [0]:

```
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))
```

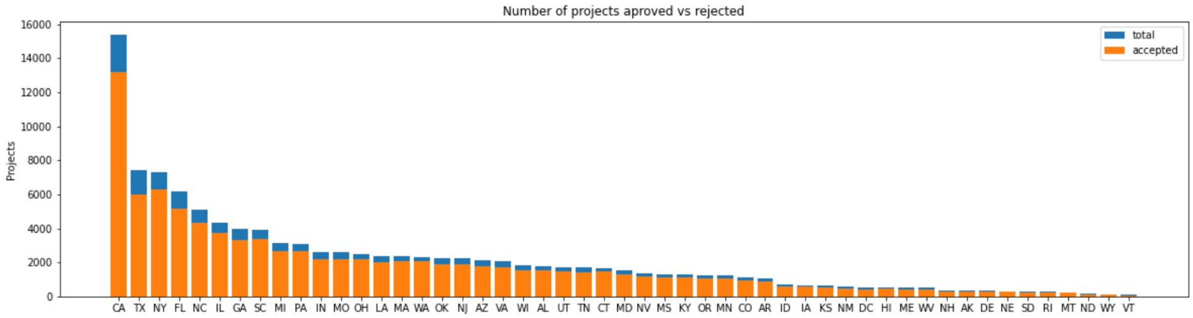
In [17]:

```
temp.head()
```

Out[17]:

	state_code	num_proposals
46	VT	0.800000
7	DC	0.802326
43	TX	0.813142
26	MT	0.816327
18	LA	0.831245

```
In [18]: univariate_barplots(project_data, 'school_state')
```

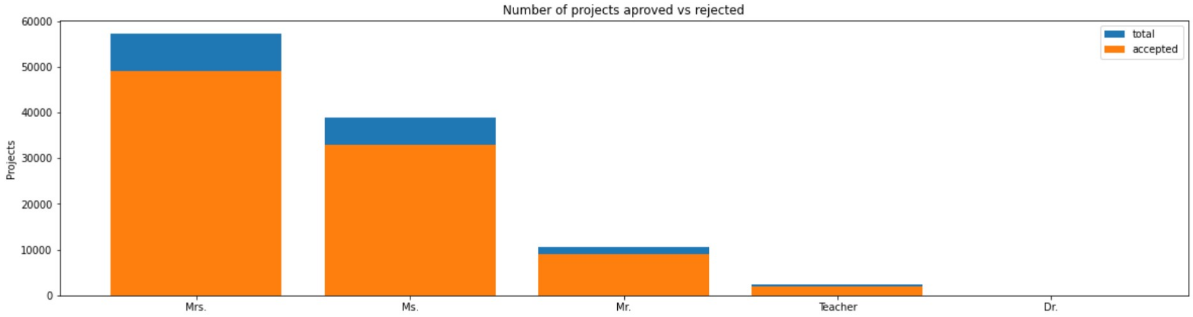


```
   school_state  project_is_approved  total       Avg
4            CA                13204  15387  0.858127
43           TX                 6014   7396  0.813142
34           NY                 6291   7318  0.859661
9            FL                 5144   6185  0.831690
27           NC                 4353   5091  0.855038
==================================================
   school_state  project_is_approved  total       Avg
39           RI                  243    285  0.852632
26           MT                  200    245  0.816327
28           ND                  127    143  0.888112
50           WY                   82     98  0.836735
46           VT                   64     80  0.800000
```

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

2 Univariate Analysis: teacher_prefix

```
In [19]: univariate_barplots(project_data, 'teacher_prefix', 'project_is_approved' , top=Fal
         se)
```

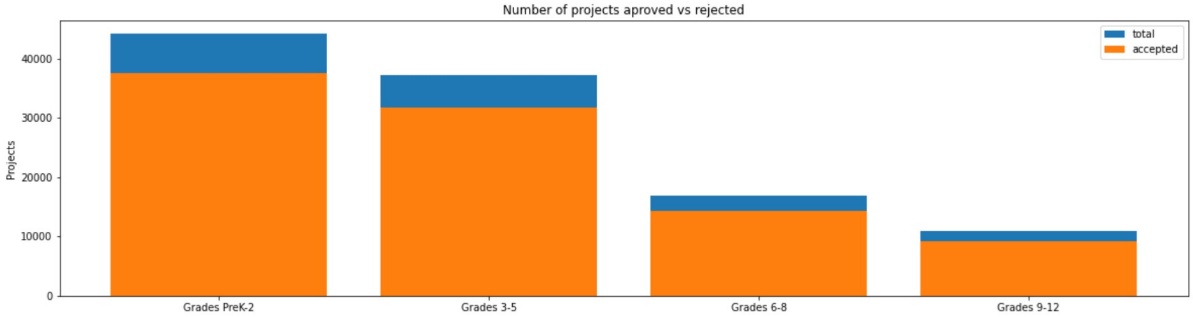


```
  teacher_prefix  project_is_approved  total       Avg
2           Mrs.                48997  57269  0.855559
3            Ms.                32860  38955  0.843537
1            Mr.                 8960  10648  0.841473
4        Teacher                 1877   2360  0.795339
0            Dr.     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

3 Univariate Analysis: project_grade_category

In [20]:

```
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                31727  37135  0.854369
1             Grades 6-8                14258  16923  0.842522
2            Grades 9-12                 9182  10962  0.837621
==================================================
```

```
  project_grade_category  project_is_approved  total       Avg
3          Grades PreK-2                37536  44225  0.848751
0             Grades 3-5                31727  37135  0.854369
1             Grades 6-8                14258  16923  0.842522
2            Grades 9-12                 9182  10962  0.837621
```

4 Univariate Analysis: project_subject_categories

In [0]:

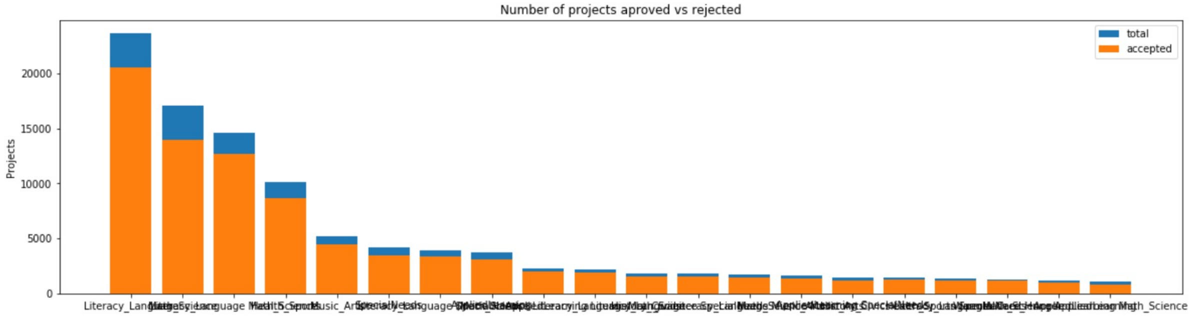
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# 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 [22]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)
```

Out[22]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:2

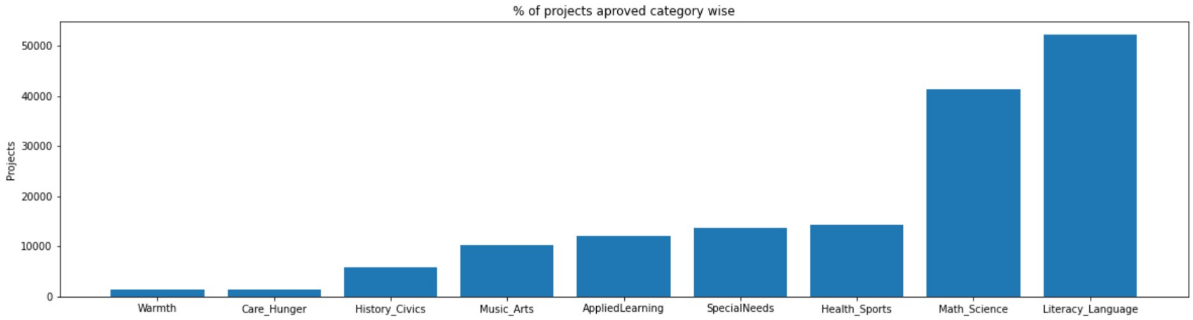
In [0]:
```
univariate_barplots(project_data, 'clean_categories', 'project_is_approved', top=2
0)
```



```
                    clean_categories  project_is_approved  total       Avg
24                  Literacy_Language                20519  23654  0.867464
32                       Math_Science                13991  17072  0.819529
28  Literacy_Language Math_Science                12723  14634  0.869414
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
```

In [0]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/408
4039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
```

In [24]:
```
# 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()
```



In [25]:
```
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         :     41419
Literacy_Language    :     52236
```

1.2.5 Univariate Analysis: project_subject_subcategories

In [0]:

```python
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())
```

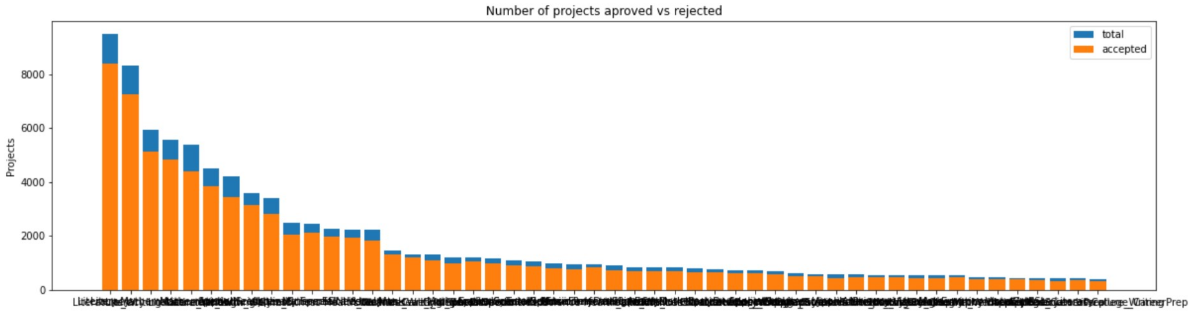
In [27]:

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

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:2

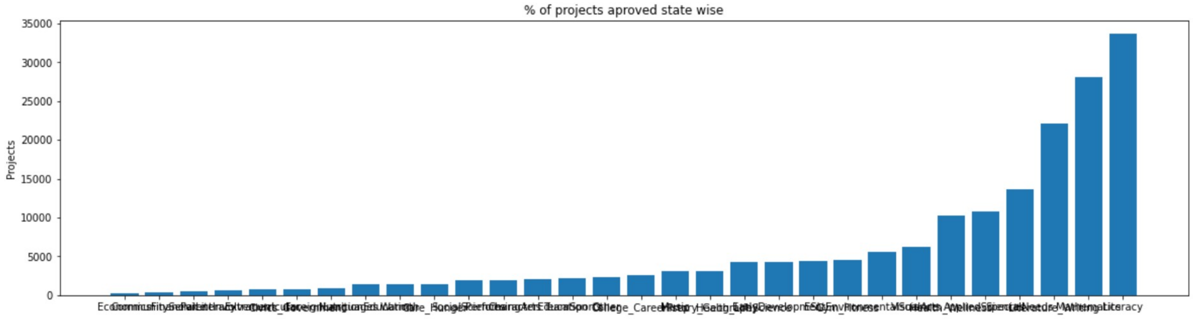
In [28]:
```
univariate_barplots(project_data, 'clean_subcategories', 'project_is_approved', top
=50)
```



```
                    clean_subcategories  project_is_approved  total       Avg
317                            Literacy                 8371   9486  0.882458
319                Literacy Mathematics                 7259   8324  0.872057
331  Literature_Writing Mathematics                     5139   5922  0.867781
318     Literacy Literature_Writing                     4823   5571  0.865733
342                         Mathematics                 4385   5379  0.815207
==================================================
                        clean_subcategories  project_is_approved  total       Avg
196        EnvironmentalScience Literacy                      389    444  0.876126
127                                    ESL                    349    421  0.828979
79                      College_CareerPrep                    343    421  0.814727
17   AppliedSciences Literature_Writing                       361    420  0.859524
3    AppliedSciences College_CareerPrep                       330    405  0.814815
```

In [0]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/408
4039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [30]:
```
# 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 [31]:

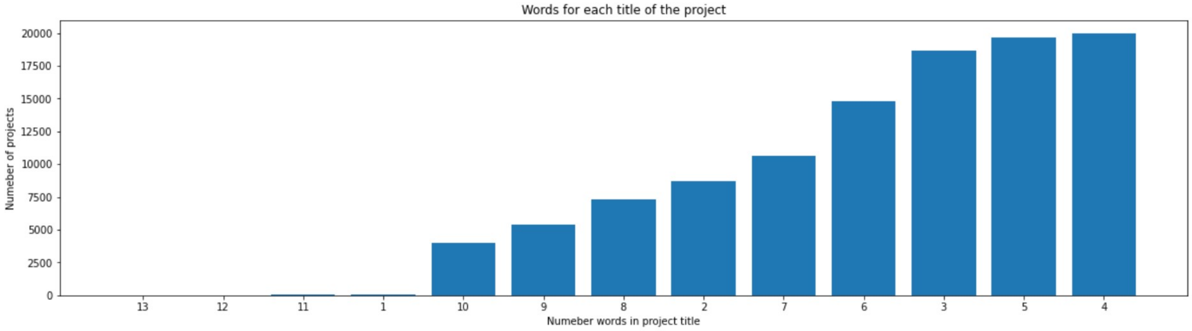
```
for i, j in sorted_sub_cat_dict.items():
    print("{:20} :{:10}".format(i,j))
```

```
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   :     22177
Mathematics          :     28072
Literacy             :     33699
```

1.2.6 Univariate Analysis: Text features (Title)

In [32]:

```
#How to calculate number of words in a string in DataFrame: https://stackoverflow.c
om/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()
```

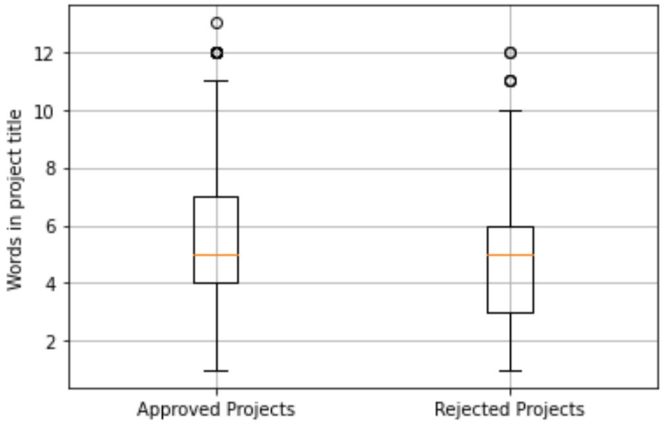


In [0]:

```
approved_title_word_count = project_data[project_data['project_is_approved']==1]['p
roject_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]['p
roject_title'].str.split().apply(len)
rejected_title_word_count = rejected_title_word_count.values
```

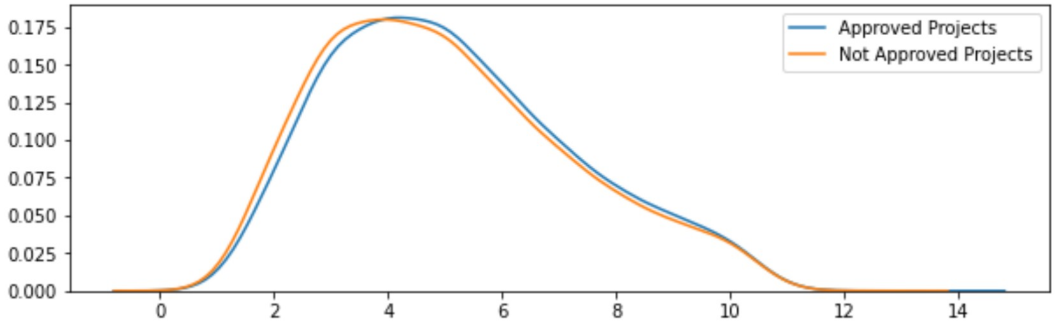
In [34]:

```
# 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()
```



In [35]:

```
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()
```



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

In [0]:

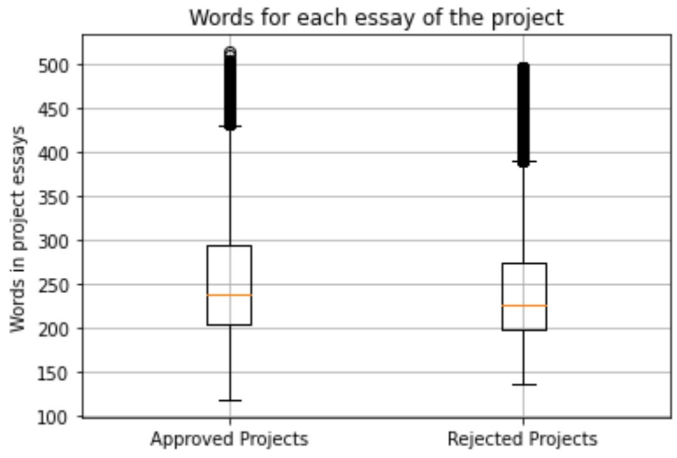
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)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

In [0]:

```
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
```

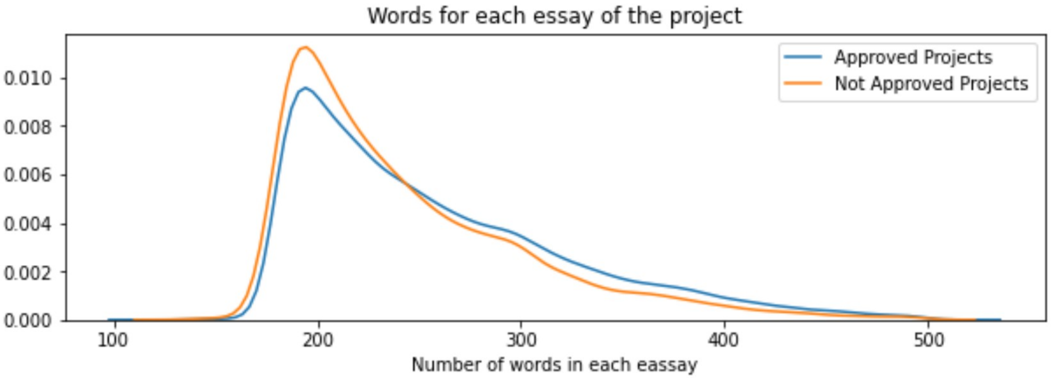
In [38]:

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



In [39]:

```
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()
```



1.2.8 Univariate Analysis: Cost per project

In [40]:

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

Out[40]:

	id	description	quantity	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 [41]:

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

Out[41]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [42]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
project_data.head(5)
```

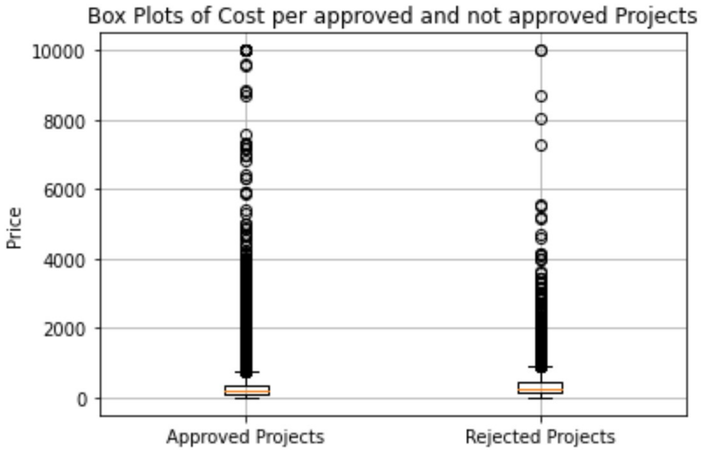
Out[42]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:2
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:0
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:1
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:1

In [0]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].valu
es

rejected_price = project_data[project_data['project_is_approved']==0]['price'].valu
es
```

In [44]:

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



In [45]:

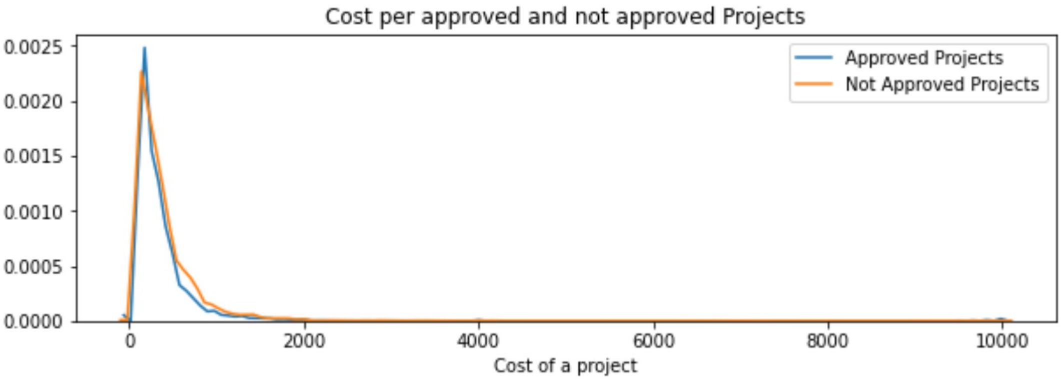
```
approved_price
```

Out[45]:

```
array([299.  , 232.9 ,  67.98, ..., 239.96,  73.05, 109.9 ])
```

In [46]:

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



In [47]:

```
# 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     |       77.374      |         118.56        |
|     25     |       99.95       |        140.892        |
|     30     |      116.672      |         162.23        |
|     35     |      137.207      |        184.014        |
|     40     |       157.0       |        208.632        |
|     45     |      178.259      |        235.106        |
|     50     |       198.99      |        263.145        |
|     55     |       223.99      |         292.61        |
|     60     |      255.598      |        325.144        |
|     65     |       285.41      |         362.39        |
|     70     |      321.222      |         399.99        |
|     75     |       366.07      |        449.945        |
|     80     |      411.666      |        519.282        |
|     85     |       479.0       |        618.276        |
|     90     |      593.082      |        739.356        |
|     95     |      801.494      |        992.486        |
|    100     |       9999.0      |         9999.0        |
+------------+-------------------+-----------------------+
```

In [48]:

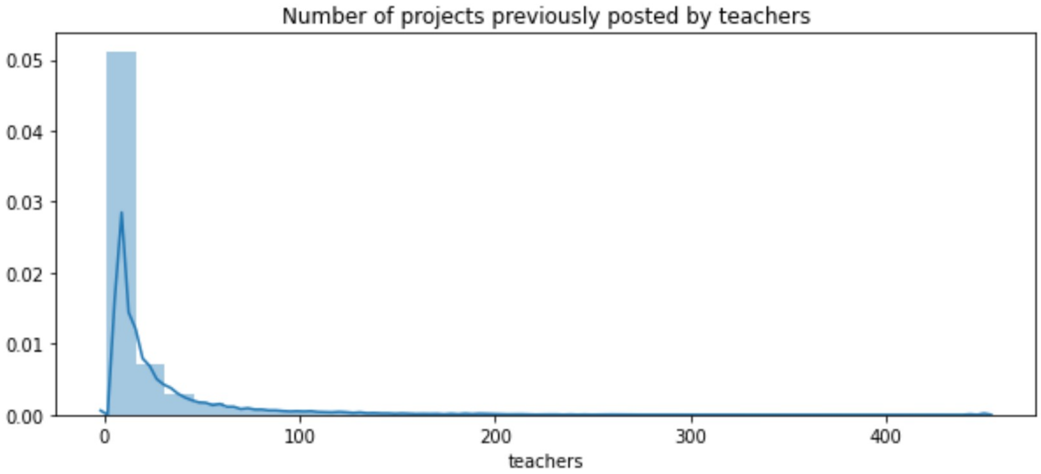
```
print(project_data.project_is_approved.value_counts())
```

```
1    92703
0    16542
Name: project_is_approved, dtype: int64
```

1.2.9 Univariate Analysis: teacher_number_of_previously_posted_projects

In [74]:

```
temp=project_data.teacher_number_of_previously_posted_projects


prev_posted= project_data[temp>0]
not_posted = project_data[temp == 0]

plt.figure(figsize=(10,4))
sns.distplot(prev_posted['teacher_number_of_previously_posted_projects'], bins=30)
plt.title('Distribution of projects previously posted by teachers')
plt.xlabel("teachers")
plt.show()
```



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 [76]:

```
a=list(project_data[['project_resource_summary']].values)
a=list(map(str,a))

indices=[]
j=0
for i in a:
    if re.search("[0-9]",i):
        indices.append(j)
    j+=1

print(indices)
```

```
[12, 14, 16, 19, 25, 26, 39, 40, 41, 45, 71, 107, 112, 117, 122, 129, 150, 156,
161, 169, 173, 178, 193, 209, 211, 220, 222, 225, 226, 237, 255, 284, 292, 299,
308, 309, 312, 315, 329, 332, 341, 347, 351, 381, 385, 388, 404, 414, 416, 422,
427, 432, 436, 443, 446, 447, 456, 461, 466, 470, 474, 479, 500, 512, 513, 520,
521, 528, 529, 541, 549, 554, 568, 572, 580, 583, 608, 616, 620, 624, 634, 641,
649, 655, 669, 671, 681, 688, 695, 703, 706, 716, 722, 729, 731, 747, 755, 757,
761, 771, 779, 791, 793, 795, 808, 818, 820, 824, 831, 838, 840, 848, 849, 852,
859, 861, 865, 870, 876, 877, 879, 884, 892, 905, 916, 918, 922, 926, 929, 946,
955, 963, 964, 967, 981, 984, 1000, 1019, 1024, 1028, 1032, 1041, 1043, 1044, 10
59, 1062, 1070, 1088, 1102, 1105, 1109, 1114, 1120, 1121, 1122, 1137, 1140, 114
2, 1169, 1183, 1189, 1191, 1194, 1195, 1197, 1201, 1207, 1218, 1221, 1227, 1232,
1240, 1251, 1256, 1261, 1262, 1287, 1294, 1301, 1308, 1326, 1329, 1336, 1337, 13
41, 1343, 1345, 1349, 1364, 1368, 1375, 1387, 1397, 1403, 1404, 1405, 1409, 141
3, 1415, 1437, 1441, 1442, 1449, 1460, 1467, 1468, 1473, 1478, 1485, 1486, 1496,
1497, 1509, 1510, 1522, 1528, 1544, 1545, 1550, 1562, 1570, 1577, 1588, 1589, 15
98, 1601, 1613, 1618, 1619, 1621, 1624, 1626, 1629, 1669, 1671, 1672, 1705, 170
7, 1722, 1741, 1742, 1747, 1748, 1755, 1758, 1765, 1771, 1772, 1796, 1797, 1812,
1814, 1829, 1853, 1857, 1865, 1866, 1881, 1913, 1915, 1919, 1927, 1936, 1940, 19
47, 1960, 1976, 1977, 1984, 1986, 1993, 1995, 2011, 2030, 2032, 2036, 2039, 204
9, 2055, 2063, 2074, 2087, 2088, 2091, 2092, 2094, 2102, 2112, 2113, 2116, 2127,
2131, 2136, 2139, 2143, 2180, 2181, 2182, 2185, 2191, 2192, 2197, 2208, 2224, 22
29, 2237, 2258, 2259, 2262, 2287, 2305, 2311, 2318, 2321, 2326, 2329, 2330, 233
2, 2334, 2339, 2344, 2346, 2347, 2353, 2356, 2374, 2392, 2396, 2410, 2424, 2431,
2437, 2448, 2453, 2463, 2469, 2475, 2477, 2481, 2483, 2484, 2491, 2494, 2495, 25
03, 2504, 2505, 2509, 2514, 2519, 2525, 2527, 2534, 2537, 2553, 2578, 2579, 260
4, 2607, 2609, 2618, 2622, 2625, 2628, 2629, 2631, 2651, 2656, 2662, 2665, 2677,
2678, 2691, 2692, 2714, 2738, 2739, 2755, 2759, 2764, 2767, 2772, 2778, 2779, 27
92, 2794, 2796, 2803, 2808, 2824, 2826, 2834, 2841, 2844, 2847, 2849, 2861, 287
4, 2878, 2888, 2891, 2898, 2899, 2908, 2909, 2912, 2920, 2926, 2949, 2950, 2951,
2962, 2970, 2971, 2976, 2979, 2980, 2986, 2987, 2992, 3002, 3017, 3025, 3029, 30
32, 3034, 3066, 3068, 3072, 3081, 3084, 3086, 3097, 3099, 3107, 3115, 3120, 312
1, 3126, 3136, 3137, 3157, 3162, 3168, 3175, 3177, 3179, 3182, 3186, 3192, 3193,
3210, 3216, 3234, 3243, 3249, 3253, 3256, 3265, 3274, 3275, 3282, 3284, 3303, 33
06, 3311, 3312, 3320, 3332, 3335, 3342, 3344, 3345, 3354, 3355, 3376, 3390, 341
2, 3440, 3460, 3461, 3463, 3472, 3476, 3493, 3497, 3498, 3499, 3500, 3504, 3507,
3513, 3518, 3531, 3542, 3550, 3552, 3559, 3564, 3588, 3592, 3618, 3628, 3633, 36
38, 3639, 3640, 3660, 3665, 3683, 3691, 3692, 3697, 3701, 3704, 3711, 3712, 371
4, 3721, 3730, 3731, 3732, 3742, 3749, 3764, 3770, 3783, 3790, 3793, 3796, 3803,
3810, 3811, 3824, 3827, 3830, 3831, 3855, 3856, 3862, 3870, 3874, 3878, 3882, 38
93, 3900, 3903, 3910, 3911, 3914, 3918, 3920, 3921, 3930, 3946, 3948, 3961, 396
7, 3968, 3969, 3995, 4020, 4031, 4040, 4046, 4060, 4075, 4080, 4092, 4100, 4107,
4109, 4112, 4114, 4118, 4119, 4125, 4128, 4131, 4140, 4144, 4145, 4155, 4162, 41
68, 4170, 4178, 4190, 4192, 4194, 4204, 4206, 4208, 4220, 4231, 4233, 4246, 425
3, 4266, 4278, 4280, 4289, 4296, 4305, 4308, 4312, 4314, 4321, 4322, 4330, 4336,
4337, 4346, 4350, 4368, 4382, 4402, 4404, 4405, 4412, 4420, 4422, 4428, 4432, 44
33, 4445, 4448, 4450, 4464, 4466, 4477, 4478, 4494, 4501, 4503, 4504, 4509, 451
2, 4513, 4517, 4522, 4528, 4531, 4550, 4555, 4557, 4565, 4568, 4571, 4583, 4588,
4589, 4609, 4621, 4628, 4639, 4655, 4664, 4673, 4691, 4700, 4721, 4725, 4730, 47
33, 4750, 4754, 4756, 4763, 4773, 4778, 4783, 4806, 4807, 4812, 4814, 4821, 482
8, 4833, 4836, 4841, 4848, 4851, 4856, 4871, 4889, 4891, 4892, 4894, 4895, 4901,
4904, 4905, 4911, 4919, 4929, 4931, 4933, 4937, 4943, 4956, 4959, 4974, 4981, 49
95, 4996, 4999, 5008, 5013, 5023, 5025, 5029, 5035, 5043, 5044, 5045, 5048, 505
3, 5060, 5071, 5095, 5108, 5115, 5119, 5128, 5149, 5153, 5154, 5157, 5172, 5182,
5184, 5187, 5192, 5197, 5201, 5228, 5237, 5243, 5244, 5252, 5258, 5261, 5264, 52
70, 5280, 5293, 5294, 5318, 5341, 5354, 5360, 5362, 5364, 5383, 5388, 5389, 539
1, 5392, 5401, 5403, 5414, 5423, 5427, 5435, 5439, 5445, 5473, 5481, 5489, 5492,
5493, 5494, 5504, 5505, 5510, 5514, 5522, 5524, 5530, 5539, 5542, 5543, 5557, 55
58, 5564, 5572, 5575, 5588, 5609, 5610, 5612, 5626, 5629, 5636, 5637, 5674, 568
9, 5698, 5709, 5712, 5717, 5720, 5746, 5749, 5760, 5765, 5769, 5771, 5789, 5790,
5811, 5812, 5827, 5830, 5834, 5844, 5845, 5856, 5859, 5861, 5863, 5864, 5872, 58
76, 5877, 5882, 5887, 5911, 5938, 5954, 5969, 5983, 5986, 5993, 6003, 6005, 601
7, 6024, 6027, 6031, 6036, 6049, 6050, 6067, 6076, 6085, 6090, 6092, 6094, 6097,
6105, 6109, 6118, 6122, 6126, 6129, 6130, 6142, 6143, 6146, 6155, 6165, 6166, 61
86, 6203, 6224, 6229, 6232, 6234, 6247, 6261, 6278, 6281, 6283, 6284, 6311, 631
```

In [77]:

```
presence_of_numerical_digits=project_data.iloc[indices][['project_is_approved']]
presence_of_numerical_digits.head(5)
```

Out[77]:

	project_is_approved
12	0
14	0
16	1
19	1
25	0

In [78]:

```
print(presence_of_numerical_digits.shape[0])
print(presence_of_numerical_digits['project_is_approved'].value_counts())
```

```
15762
1    14096
0     1666
Name: project_is_approved, dtype: int64
```

In [79]:

```
presence_of_numerical_digits['project_is_approved'].value_counts()[1]/presence_of_numerical_digits.shape[0]
```

Out[79]:

```
0.8943027534576831
```