Donors choose data analysis

Table of contents

Little History about Data Set

Founded in 2000 by a high school teacher in the Bronx, DonorsChoose.org empowers public school teachers from across the country to request much-needed materials and experiences for their students. At any given time, there are thousands of classroom requests that can be brought to life with a gift of any amount.

Answers to What and Why Questions on Data Set

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 p **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy` • `First Grade Fun`

Feature	Description
`project_grade_category`	Grade level of students for which the project is targeted. One of the follow enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) su categories for the project from the fo 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, Ma & Science`
`school_state`	State where school is located (Two-l U.S. postal code). **Example:** `WY`

Feature	Description
`project_subject_subcategories`	One or more (comma-separated) su subcategories for the project. **Examp** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources nee the project. **Example:** • `My students need hands 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 w submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c`

Feature	Description
`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 previ submitted by the same teacher. **Exa** `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.

Importing required libraries

In [1]:

```
# numpy for easy numerical computations
import numpy as np
# pandas for dataframes and filterings
import pandas as pd
```

```
# sqlite3 library for performing operations on sqlite file
import sqlite3
# matplotlib for plotting graphs
import matplotlib.pyplot as plt
# seaborn library for easy plotting
import seaborn as sbrn
# warnings library for specific settings
import warnings
# regularlanguage for regex operations
import re
# For loading precomputed models
import pickle
# For loading natural language processing tool-kit
import nltk
# For calculating mathematical terms
import math

# For loading files from google drive
from google.colab import drive
# For working with files in google drive
drive.mount('/content/drive')
# tqdm for tracking progress of loops
from tqdm import tqdm_notebook as tqdm
# For creating dictionary of words
from collections import Counter
# For creating BagOfWords Model
from sklearn.feature_extraction.text import CountVectorizer
# For creating TfidfModel
from sklearn.feature_extraction.text import TfidfVectorizer
# For standardizing values
from sklearn.preprocessing import StandardScaler
# For merging sparse matrices along row direction
from scipy.sparse import hstack
# For merging sparse matrices along column direction
from scipy.sparse import vstack
# For calculating TSNE values
from sklearn.manifold import TSNE
# For calculating the accuracy score on cross validate data
from sklearn.metrics import accuracy_score
```

```
# For performing the k-fold cross validation
from sklearn.model_selection import cross_val_score
# For splitting the data set into test and train data
from sklearn import model_selection
# Support Vector classifier for classification
from sklearn.svm import SVC
# For reducing dimensions of data
from sklearn.decomposition import TruncatedSVD
# For using svm classifer - hinge loss function of sgd
from sklearn import linear_model
# For creating samples for making dataset balanced
from sklearn.utils import resample
# For shuffling the dataframes
from sklearn.utils import shuffle
# For calculating roc_curve parameters
from sklearn.metrics import roc_curve
# For calculating auc value
from sklearn.metrics import auc
# For displaying results in table format
from prettytable import PrettyTable
# For generating confusion matrix
from sklearn.metrics import confusion_matrix
# For using gridsearch cv to find best parameter
from sklearn.model_selection import GridSearchCV
# For performing min-max standardization to features
from sklearn.preprocessing import MinMaxScaler
# For calculating sentiment score of the text
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')

warnings.filterwarnings('ignore')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?
client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuser
content.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=emai
l%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2
Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2
Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Faut
h%2Fpeopleapi.readonly&response_type=code
```

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

```
/usr/local/lib/python3.6/dist-packages/nltk/twitter/__init__.py:20: UserWarning: The twython library has not been installed. Some functionality from the twitter package will not be available.
  warnings.warn("The twython library has not been installed. "
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

Reading and Storing Data

In [0]:
```python
projectsData = pd.read_csv('drive/My Drive/train_data.csv');
resourcesData = pd.read_csv('drive/My Drive/resources.csv');
```

In [3]:
```python
projectsData.head(3)
```

Out[3]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ

In [4]:
```
projectsData.tail(3)
```

Out[4]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
109245	143653	p155633	cdbfd04aa041dc6739e9e576b1fb1478	Mrs.	NJ
109246	164599	p206114	6d5675dbfafa1371f0e2f6f1b716fe2d	Mrs.	NY
109247	128381	p191189	ca25d5573f2bd2660f7850a886395927	Ms.	VA

In [5]:
```
resourcesData.head(3)
```

Out[5]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45

In [6]:
```
resourcesData.tail(3)
```

Out[6]:

	id	description	quantity	price
1541269	p031981	Black Electrical Tape (GIANT 3 PACK) Each Roll...	6	8.99
1541270	p031981	Flormoon DC Motor Mini Electric Motor 0.5-3V 1...	2	8.14
1541271	p031981	WAYLLSHINE 6PCS 2 x 1.5V AAA Battery Spring Cl...	2	7.39

Helper functions and classes

In [0]:
```
def equalsBorder(numberOfEqualSigns):
    """
    This function prints passed number of equal signs
    """
    print("="* numberOfEqualSigns);
```

In [0]:
```
# Citation link: https://stackoverflow.com/questions/8924173/how-do-i-p
rint-bold-text-in-python
class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
```

```
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'
    END = '\033[0m'
```

In [0]:
```
def printStyle(text, style):
    "This function prints text with the style passed to it"
    print(style + text + color.END);
```

Shapes of projects data and resources data

In [10]:
```
printStyle("Number of data points in projects data: {}".format(projects
Data.shape[0]), color.BOLD);
printStyle("Number of attributes in projects data:{}".format(projectsDa
ta.shape[1]), color.BOLD);
equalsBorder(60);
printStyle("Number of data points in resources data: {}".format(resourc
esData.shape[0]), color.BOLD);
printStyle("Number of attributes in resources data: {}".format(resource
sData.shape[1]), color.BOLD);
```

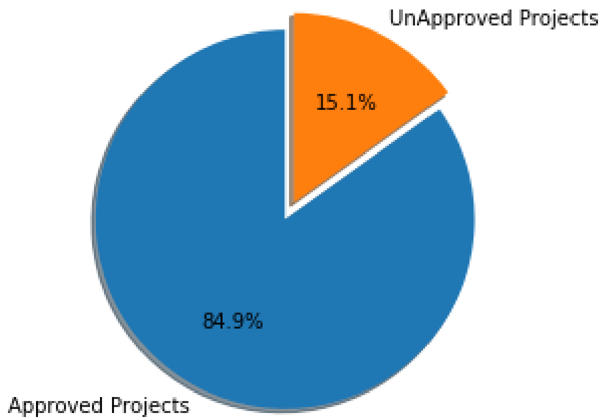
```
Number of data points in projects data: 109248
Number of attributes in projects data:17
============================================================
Number of data points in resources data: 1541272
Number of attributes in resources data: 4
```

Univariate data analysis

In [11]:
```
approvedProjects = projectsData[projectsData.project_is_approved == 1].
shape[0];
unApprovedProjects = projectsData[projectsData.project_is_approved == 0
].shape[0];
totalProjects = projectsData.shape[0];
print("Number of projects approved for funding: {}, ({})".format(approv
```

```
edProjects, (approvedProjects / totalProjects) * 100));
print("Number of projects not approved for funding: {}, ({})".format(un
ApprovedProjects, (unApprovedProjects / totalProjects) * 100));
# Pie chart representation
# Citation: https://matplotlib.org/gallery/pie_and_polar_charts/pie_fea
tures.html
labels = ["Approved Projects", "UnApproved Projects"];
explode = (0, 0.1);
sizes = [approvedProjects, unApprovedProjects];
figure, ax = plt.subplots();
ax.pie(sizes, labels = labels, explode = explode, autopct = '%1.1f%%',
shadow = True, startangle = 90);
ax.axis('equal');
plt.rcParams['figure.figsize'] = (7, 7);
plt.show();
```

```
Number of projects approved for funding: 92706, (84.85830404217927)
Number of projects not approved for funding: 16542, (15.14169595782073
9)
```



Observation:

1. There are more number of approved projects compared to rejected projects. So this is a imbalanced dataset.

Univariate Analysis : 'school_state'

Project proposal percentage in different states

In [12]:
```
groupedByStatesData = pd.DataFrame(projectsData.groupby(['school_state'
])['project_is_approved'].apply(np.mean)).reset_index();
groupedByStatesData.columns = ['state_code', 'number_of_proposals'];
groupedByStatesData = groupedByStatesData.sort_values(by=['number_of_pr
oposals'], ascending = True);
printStyle("5 States with lowest percentage of project approvals:", col
or.BOLD);
equalsBorder(60);
groupedByStatesData.head(5)
```

5 States with lowest percentage of project approvals:
==

Out[12]:

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

In [13]:
```
printStyle("5 states with highest percentage of project approvals: ", c
olor.BOLD);
equalsBorder(60);
groupedByStatesData.tail(5).iloc[::-1]
```

5 states with highest percentage of project approvals:

```
================================================================
```

Out[13]:

	state_code	number_of_proposals
8	DE	0.897959
28	ND	0.888112
47	WA	0.876178
35	OH	0.875152
30	NH	0.873563

In [0]:

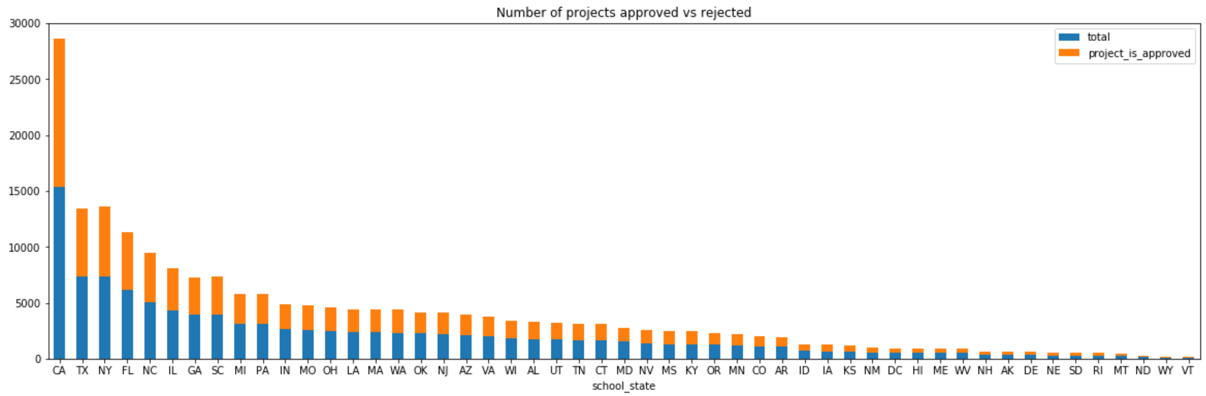
```
def univariateBarPlots(data, col1, col2 = 'project_is_approved', orient
ation = 'vertical', plot = True):
    groupedData = data.groupby(col1);
    # Count number of zeros in dataframe python: https://stackoverflow.
com/a/51540521/4084039
    tempData = pd.DataFrame(groupedData[col2].agg(lambda x: x.eq(1).sum
())).reset_index();
    tempData['total'] = pd.DataFrame(groupedData[col2].agg({'total': 'c
ount'})).reset_index()['total'];
    tempData['approval_rate'] = pd.DataFrame(groupedData[col2].agg({'ap
proval_rate': 'mean'})).reset_index()['approval_rate'];
    tempData.sort_values(by=['total'], inplace = True, ascending = Fals
e);
    tempDataWithTotalAndCol2 = tempData[['total', col2, col1]]
    if plot:
        if(orientation == 'vertical'):
            tempDataWithTotalAndCol2.plot(x = col1, align= 'center', ki
nd = 'bar', title = "Number of projects approved vs rejected", figsize
= (20, 6), stacked = True, rot = 0);
        else:
            tempDataWithTotalAndCol2.plot(x = col1, align= 'center', ki
nd = 'barh', title = "Number of projects approved vs rejected", width =
 0.8, figsize = (23, 20), stacked = True);
    return tempData;
```

In [15]:
```
statesCharacteristicsData = univariateBarPlots(projectsData, 'school_st
ate', 'project_is_approved', orientation = 'vertical');
printStyle("Top 5 states with high project proposals", color.BOLD)
equalsBorder(60);
statesCharacteristicsData.head(5)
```

Top 5 states with high project proposals
==

Out[15]:

	school_state	project_is_approved	total	approval_rate
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



In [16]:
```
printStyle("Top 5 states with least project proposals", color.BOLD)
equalsBorder(60);
statesCharacteristicsData.tail(5)
```

Top 5 states with least project proposals
==

Out[16]:

	school_state	project_is_approved	total	approval_rate
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

Observation:

1. Highest number of project proposals are from CA(California) and it was almost about 16000 projects
2. Every state has more than 80% approval rate.

Univariate Analysis: teacher_prefix

In [17]:

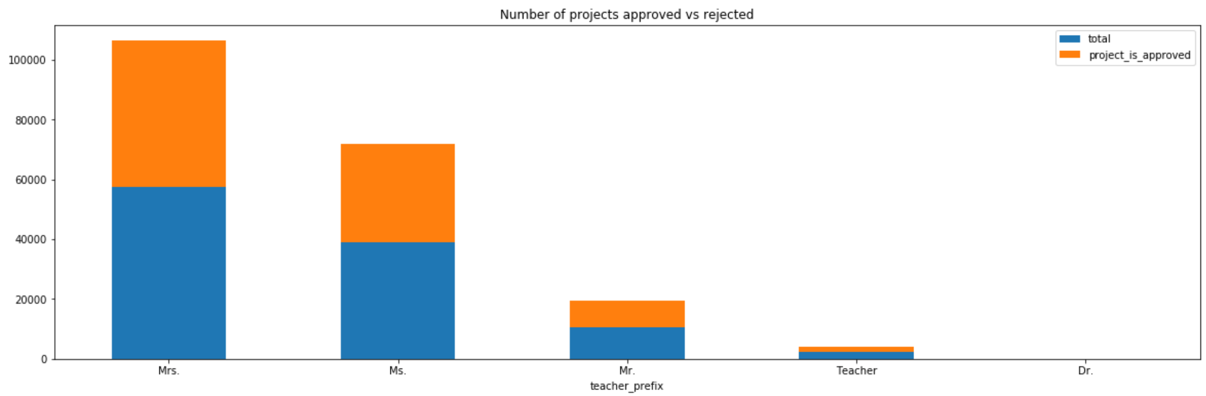
```
teacherPrefixCharacteristicsData = univariateBarPlots(projectsData, 'te
acher_prefix', 'project_is_approved', orientation = 'vertical', plot =
True);
printStyle("Project proposals characteristics based on types of person
s", color.BOLD);
equalsBorder(60);
teacherPrefixCharacteristicsData
```

```
Project proposals characteristics based on types of persons
============================================================
```

Out[17]:

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

	teacher_prefix	project_is_approved	total	approval_rate
1	Mr.	8960	10648	0.841473
4	Teacher	1877	2360	0.795339
0	Dr.	9	13	0.692308



Observataion:

1. When compared to others Dr.'s have proposed very less number of projects.
2. Women have proposed more number of projects than men.

Univariate Analysis: project_grade

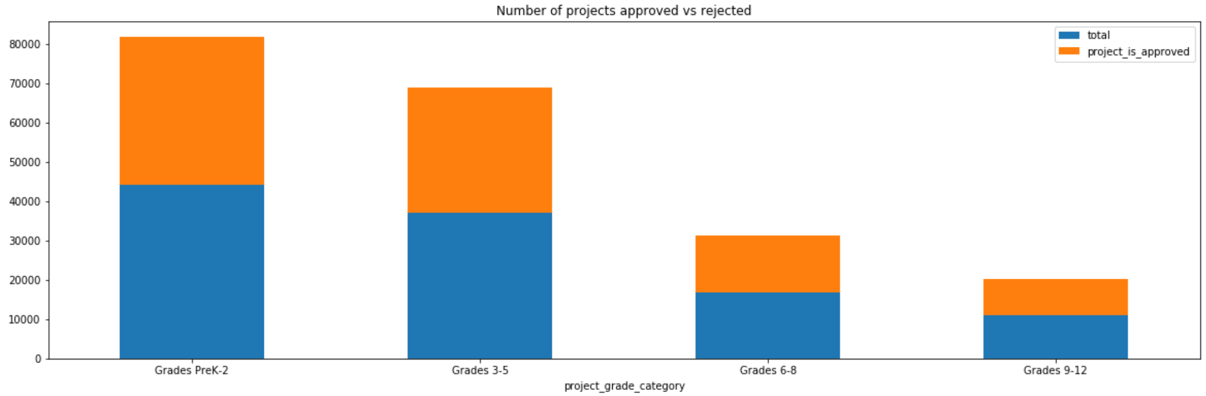
In [18]:
```
gradeCharacteristicsData = univariateBarPlots(projectsData, 'project_grade_category', 'project_is_approved', orientation = 'vertical', plot = True);
printStyle("Project proposal characteristics based on grades", color.BOLD);
equalsBorder(60);
gradeCharacteristicsData
```

Project proposal characteristics based on grades

==

Out[18]:

	project_grade_category	project_is_approved	total	approval_rate
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



Observation:

1. Most number of projects proposed are for students less than grade-5 (for primary school students) which means that children are being taught with project oriented teaching which is great.

Univariate Analysis: project_subject_categories

In [0]:

```
# 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
def cleanCategories(subjectCategories):
    cleanedCategories = []
    for subjectCategory in tqdm(subjectCategories):
        tempCategory = ""
        for category in subjectCategory.split(","):
            if 'The' in category.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
                category = category.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
            category = category.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
            tempCategory += category.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
            tempCategory = tempCategory.replace('&','_')
        cleanedCategories.append(tempCategory)
    return cleanedCategories
```

In [20]:

```
# projectDataWithCleanedCategories = pd.DataFrame(projectsData);
subjectCategories = list(projectsData.project_subject_categories);
cleanedCategories = cleanCategories(subjectCategories);
printStyle("Sample categories: ", color.BOLD);
equalsBorder(60);
print(subjectCategories[0:5]);
equalsBorder(60);
printStyle("Sample cleaned categories: ", color.BOLD);
equalsBorder(60);
print(cleanedCategories[0:5]);
projectsData['cleaned_categories'] = cleanedCategories;
projectsData.head(5)
```

Sample categories:

```
============================================================
['Literacy & Language', 'History & Civics, Health & Sports', 'Health & Sports', 'Literacy & Language, Math & Science', 'Math & Science']
```

```
==============================================================
Sample cleaned categories:
==============================================================
['Literacy_Language ', 'History_Civics Health_Sports ', 'Health_Sports
 ', 'Literacy_Language Math_Science ', 'Math_Science ']
```

Out[20]:

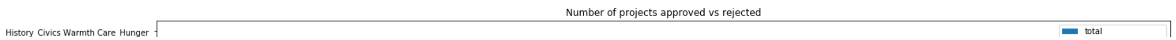
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

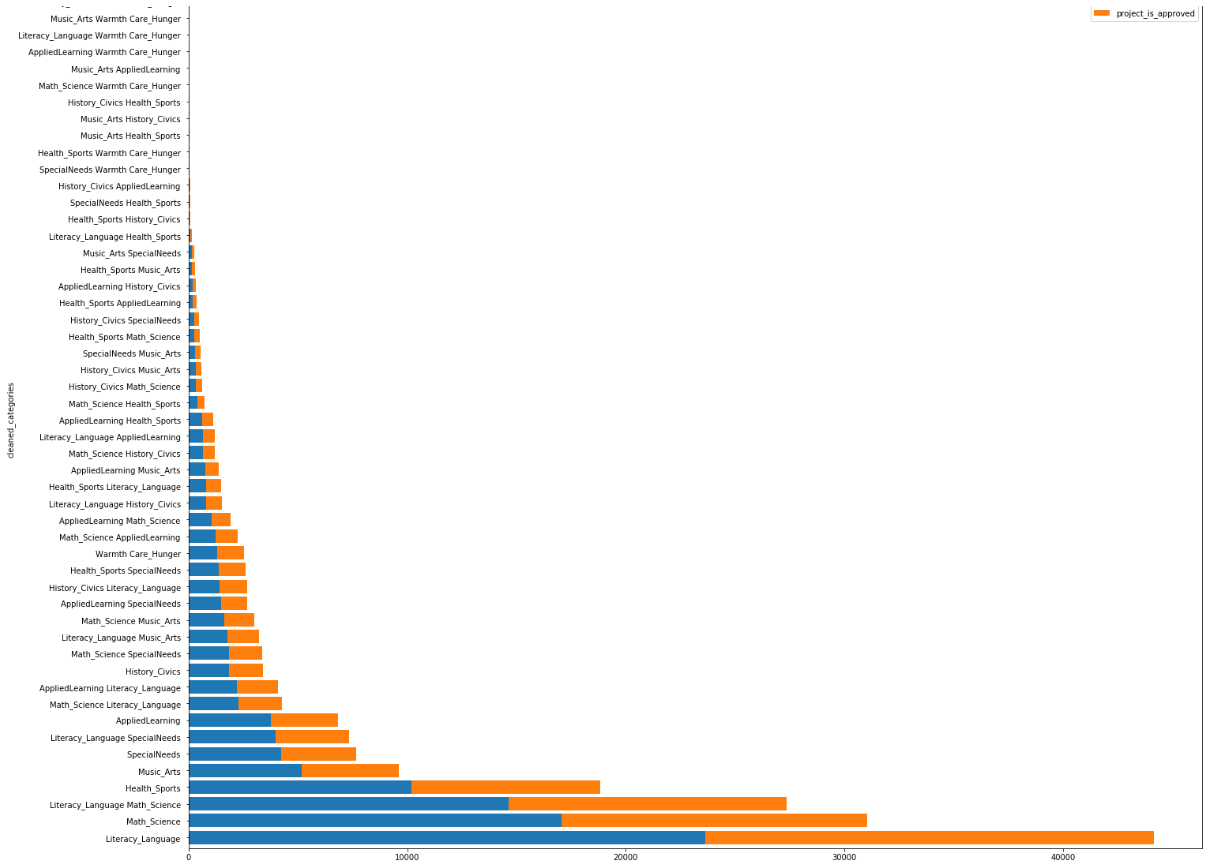
In [21]:
```python
categoriesCharacteristicsData = univariateBarPlots(projectsData, 'cleaned_categories', 'project_is_approved', orientation = 'horizontal', plot = True);
print("Project proposals characteristics based on subject categories");
equalsBorder(60);
categoriesCharacteristicsData.head(5)
```

```
Project proposals characteristics based on subject categories
============================================================
```

Out[21]:

	cleaned_categories	project_is_approved	total	approval_rate
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





In [22]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/
22898595/4084039
categoriesCounter = Counter()
for subjectCategory in projectsData.cleaned_categories.values:
    categoriesCounter.update(subjectCategory.split());
categoriesCounter
```

Out[22]:
```
Counter({'AppliedLearning': 12135,
         'Care_Hunger': 1388,
         'Health_Sports': 14223,
         'History_Civics': 5914,
         'Literacy_Language': 52239,
         'Math_Science': 41421,
```

```
            'Music_Arts': 10293,
            'SpecialNeeds': 13642,
            'Warmth': 1388})
```

In [23]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/
22898595/4084039
categoriesDictionary = dict(categoriesCounter);
sortedCategoriesDictionary = dict(sorted(categoriesDictionary.items(),
key = lambda keyValue: keyValue[1]));
sortedCategoriesData = pd.DataFrame.from_dict(sortedCategoriesDictionar
y, orient='index');
sortedCategoriesData.columns = ['subject_categories'];
printStyle("Number of projects by Subject Categories: ", color.BOLD);
equalsBorder(60);
sortedCategoriesData
```

Number of projects by Subject Categories:

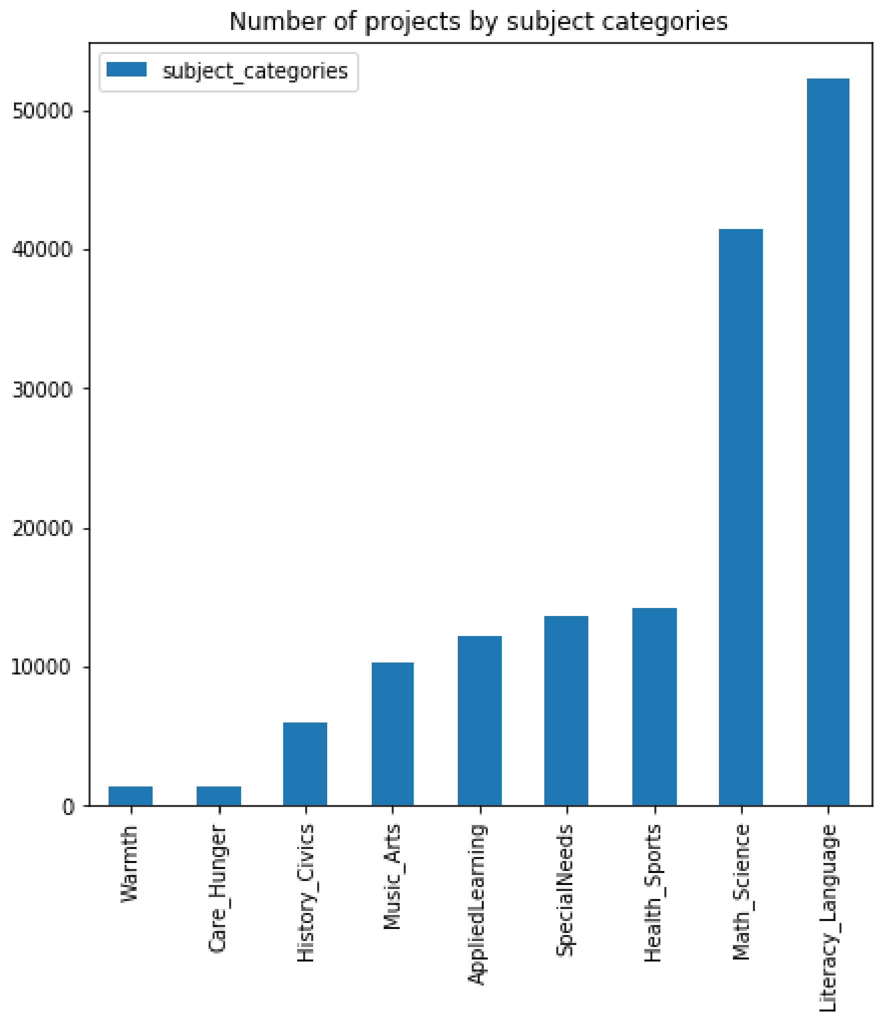
```
============================================================
```

Out[23]:

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

In [24]:
```
sortedCategoriesData.plot(kind = 'bar', title = 'Number of projects by
```

```
subject categories');
```



Observation:

1. Many number of projects proposed belong to multiple subject categories.
2. When compared to others literacy_language & math_science have large number of project proposals.

Univariate Analysis: project_subject_subcategories

In [25]:
```
subjectSubCategories = projectsData.project_subject_subcategories;
cleanedSubCategories = cleanCategories(subjectSubCategories);
printStyle("Sample subject sub categories: ", color.BOLD);
equalsBorder(70);
print(subjectSubCategories[0:5]);
equalsBorder(70);
printStyle("Sample cleaned subject sub categories: ", color.BOLD);
equalsBorder(70);
print(cleanedSubCategories[0:5]);
projectsData['cleaned_sub_categories'] = cleanedSubCategories;
```

```
Sample subject sub categories:
======================================================================
0                        ESL, Literacy
1     Civics & Government, Team Sports
2       Health & Wellness, Team Sports
3                Literacy, Mathematics
4                          Mathematics
Name: project_subject_subcategories, dtype: object
======================================================================
Sample cleaned subject sub categories:
======================================================================
['ESL Literacy ', 'Civics_Government TeamSports ', 'Health_Wellness Tea
mSports ', 'Literacy Mathematics ', 'Mathematics ']
```

In [26]:
```
projectsData.head(5)
```

Out[26]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

In [27]:

```
subCategoriesCharacteristicsData = univariateBarPlots(projectsData, 'cl
eaned_sub_categories', 'project_is_approved', plot = False);
print("Project proposals characteristics based on subject sub categorie
s");
equalsBorder(60);
subCategoriesCharacteristicsData.head(5)
```

```
Project proposals characteristics based on subject sub categories
============================================================
```

Out[27]:

	cleaned_sub_categories	project_is_approved	total	approval_rate
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

In [28]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
subjectsSubCategoriesCounter = Counter();
for subCategory in projectsData.cleaned_sub_categories:
    subjectsSubCategoriesCounter.update(subCategory.split());
subjectsSubCategoriesCounter
```

Out[28]:

```
Counter({'AppliedSciences': 10816,
         'Care_Hunger': 1388,
         'CharacterEducation': 2065,
         'Civics_Government': 815,
         'College_CareerPrep': 2568,
         'CommunityService': 441,
         'ESL': 4367,
         'EarlyDevelopment': 4254,
         'Economics': 269,
         'EnvironmentalScience': 5591,
         'Extracurricular': 810,
         'FinancialLiteracy': 568,
         'ForeignLanguages': 890,
         'Gym_Fitness': 4509,
         'Health_LifeScience': 4235,
         'Health_Wellness': 10234,
         'History_Geography': 3171,
```

```
                'Literacy': 33700,
                'Literature_Writing': 22179,
                'Mathematics': 28074,
                'Music': 3145,
                'NutritionEducation': 1355,
                'Other': 2372,
                'ParentInvolvement': 677,
                'PerformingArts': 1961,
                'SocialSciences': 1920,
                'SpecialNeeds': 13642,
                'TeamSports': 2192,
                'VisualArts': 6278,
                'Warmth': 1388})
```

In [29]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
dictionarySubCategories = dict(subjectsSubCategoriesCounter);
sortedDictionarySubCategories = dict(sorted(dictionarySubCategories.ite
ms(), key = lambda keyValue: keyValue[1]));
sortedSubCategoriesData = pd.DataFrame.from_dict(sortedDictionarySubCat
egories, orient = 'index');
sortedSubCategoriesData.columns = ['subject_sub_categories']
sortedSubCategoriesData.plot(kind = 'bar', title = "Number of projects
 by subject sub categories");
printStyle("Number of projects sorted by subject sub categories: ", col
or.BOLD);
equalsBorder(70);
sortedSubCategoriesData
```

Number of projects sorted by subject sub categories:

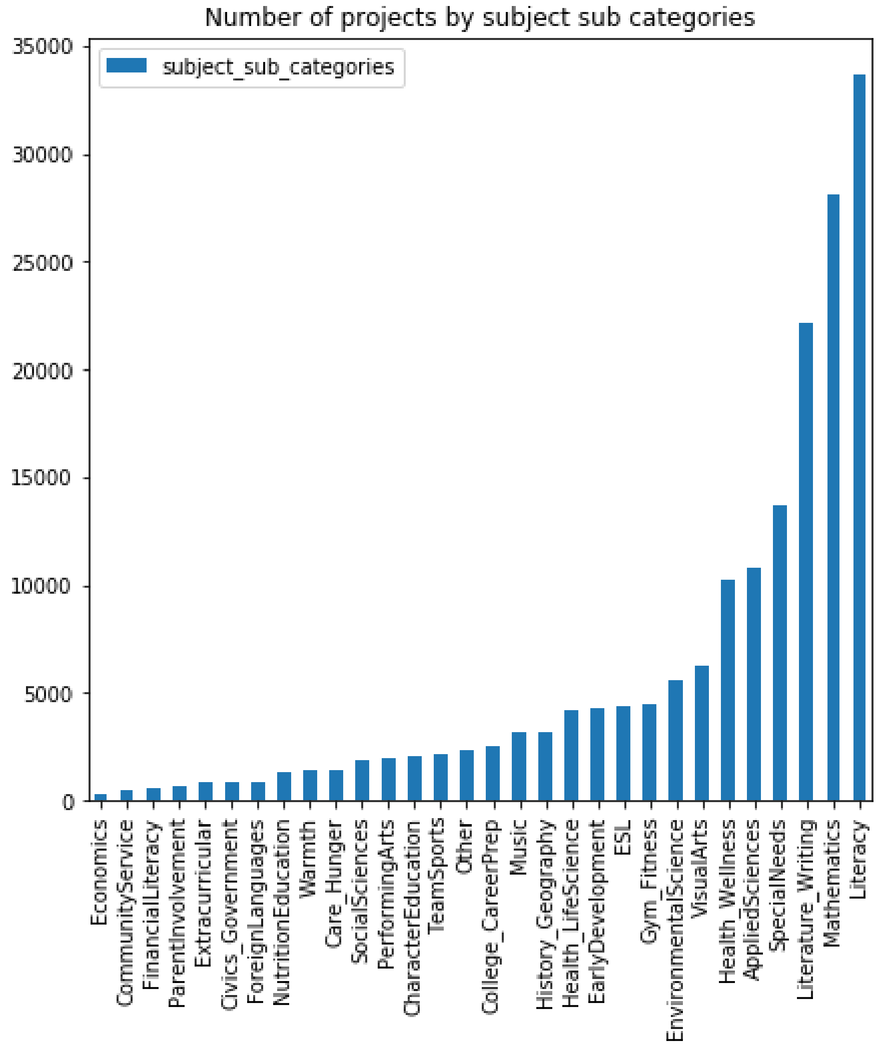
```
======================================================================
```

Out[29]:

	subject_sub_categories
Economics	269
CommunityService	441
FinancialLiteracy	568

	subject_sub_categories
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

	subject_sub_categories
AppliedSciences	10816
SpecialNeeds	13642
Literature_Writing	22179
Mathematics	28074
Literacy	33700



Observation:

1. There are more number of subject subcategories than subject categories.
2. Even more number of projects proposed belong to multiple subject sub categories.

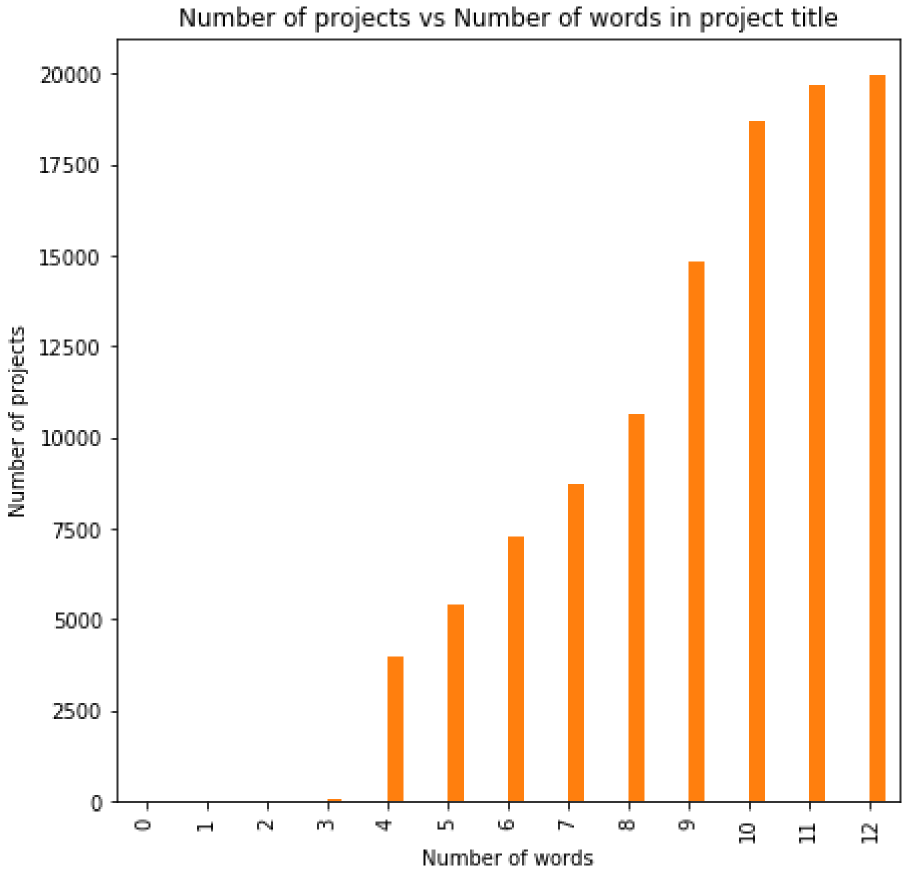
Univariate Analysis : project_title

In [30]:
```
#How to calculate number of words in a string in DataFrame: https://sta
ckoverflow.com/a/37483537/4084039
wordCounts = projectsData['project_title'].str.split().apply(len).value
_counts();
dictionaryWordCounts = dict(wordCounts);
dictionaryWordCounts = dict(sorted(dictionaryWordCounts.items(), key =
lambda kv: kv[1]));
wordCountsData = pd.DataFrame.from_dict({'number_of_words': list(dictio
naryWordCounts.keys()), 'number_of_projects': list(dictionaryWordCounts
.values())}).sort_values(by = ['number_of_projects']);
wordCountsData.plot(kind = 'bar', title = "Number of projects vs Number
 of words in project title", legend = False);
plt.xlabel('Number of words');
plt.ylabel('Number of projects');
wordCountsData
```

Out[30]:

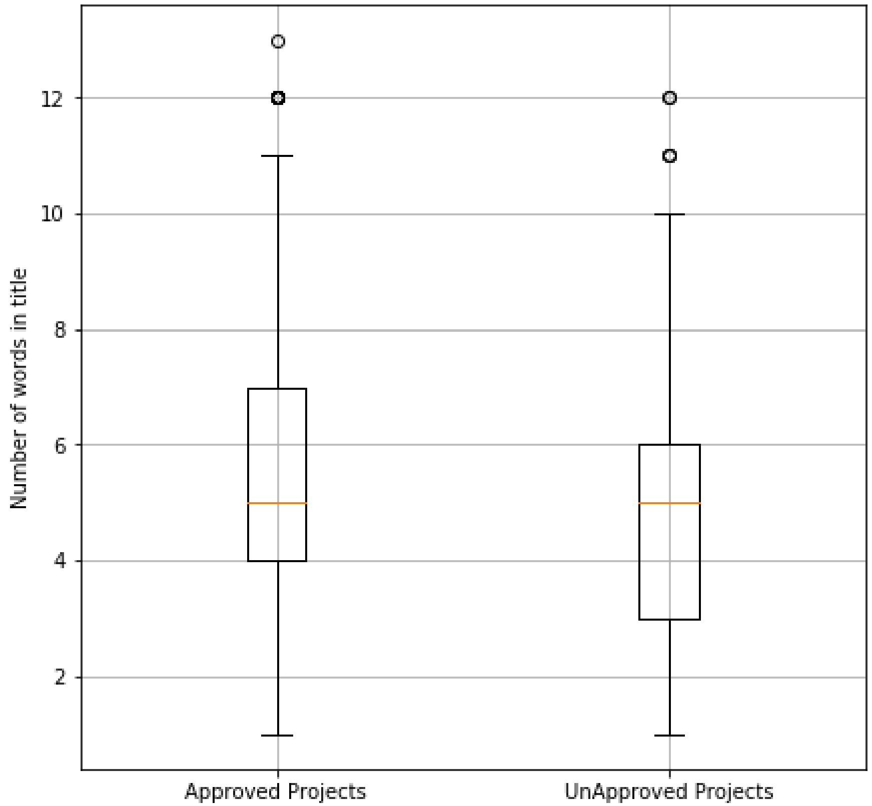
	number_of_words	number_of_projects
0	13	1
1	12	11
2	11	30
3	1	31
4	10	3968
5	9	5383
6	8	7289
7	2	8733
8	7	10631
9	6	14824
10	3	18691

	number_of_words	number_of_projects
11	5	19677
12	4	19979



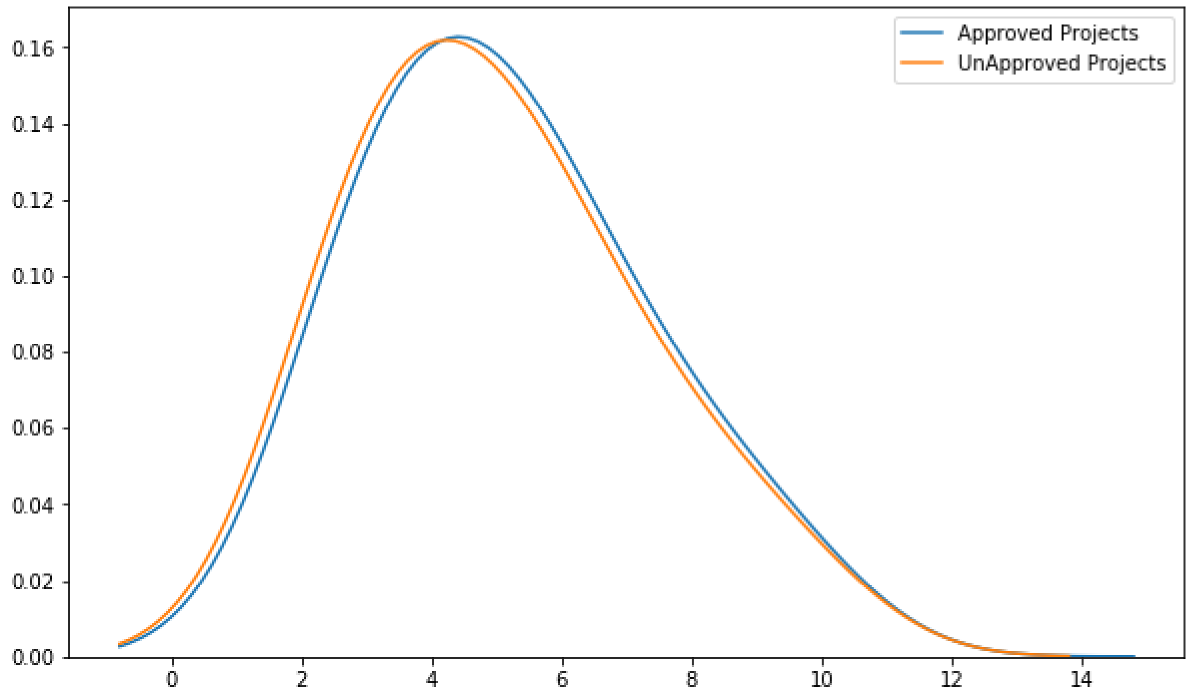
In [31]:
```
approvedNumberOfProjects = projectsData[projectsData.project_is_approve
d == 1]['project_title'].str.split().apply(len);
approvedNumberOfProjects = approvedNumberOfProjects.values
unApprovedNumberOfProjects = projectsData[projectsData.project_is_appro
ved == 0]['project_title'].str.split().apply(len);
unApprovedNumberOfProjects = unApprovedNumberOfProjects.values
```

```
plt.boxplot([approvedNumberOfProjects, unApprovedNumberOfProjects]);
plt.grid();
plt.xticks([1, 2], ['Approved Projects', 'UnApproved Projects']);
plt.ylabel('Number of words in title');
plt.show();
```



In [32]:

```
plt.figure(figsize = (10, 6));
sbrn.kdeplot(approvedNumberOfProjects, label = "Approved Projects", bw
= 0.6);
sbrn.kdeplot(unApprovedNumberOfProjects, label = "UnApproved Projects",
 bw = 0.6);
plt.legend();
plt.show();
```



Observations:

1. Most of the approved projects have between 4 to 8 number of words in their project_title.
2. Most of the rejected projects have between 3 to 6 number of words in their project_title.

Univariate Analysis: project_essay_1,2,3,4

In [33]:
```
projectsData['project_essay'] = projectsData['project_essay_1'].map(str
) + projectsData['project_essay_2'].map(str) + \
                                projectsData['project_essay_3'].map(str
) + projectsData['project_essay_4'].map(str);
projectsData.head(5)
```

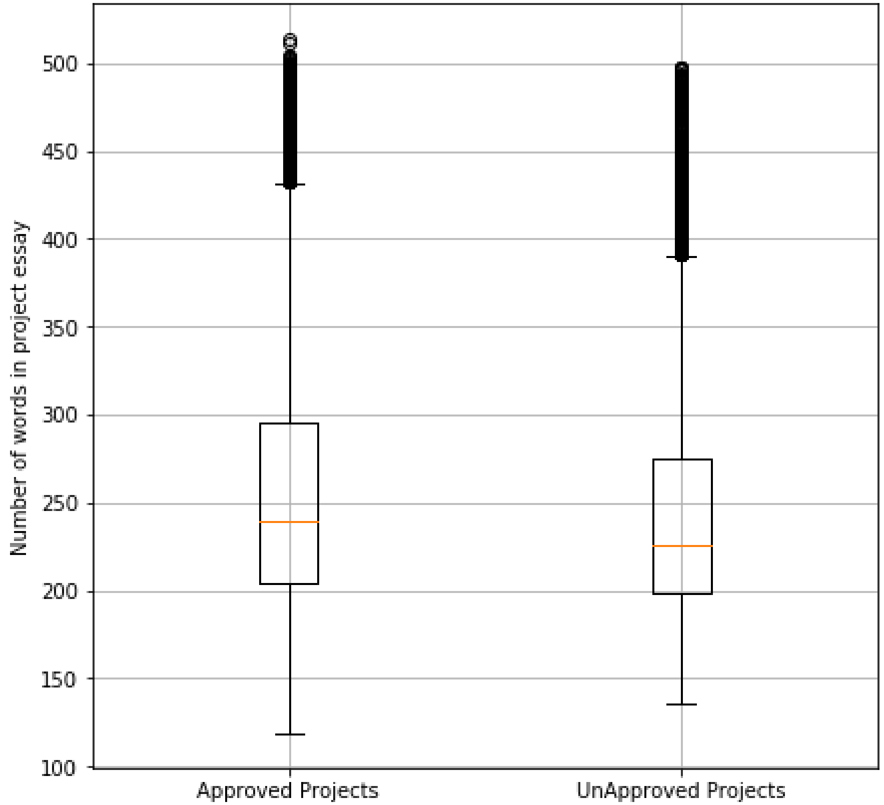
Out[33]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

In [34]:

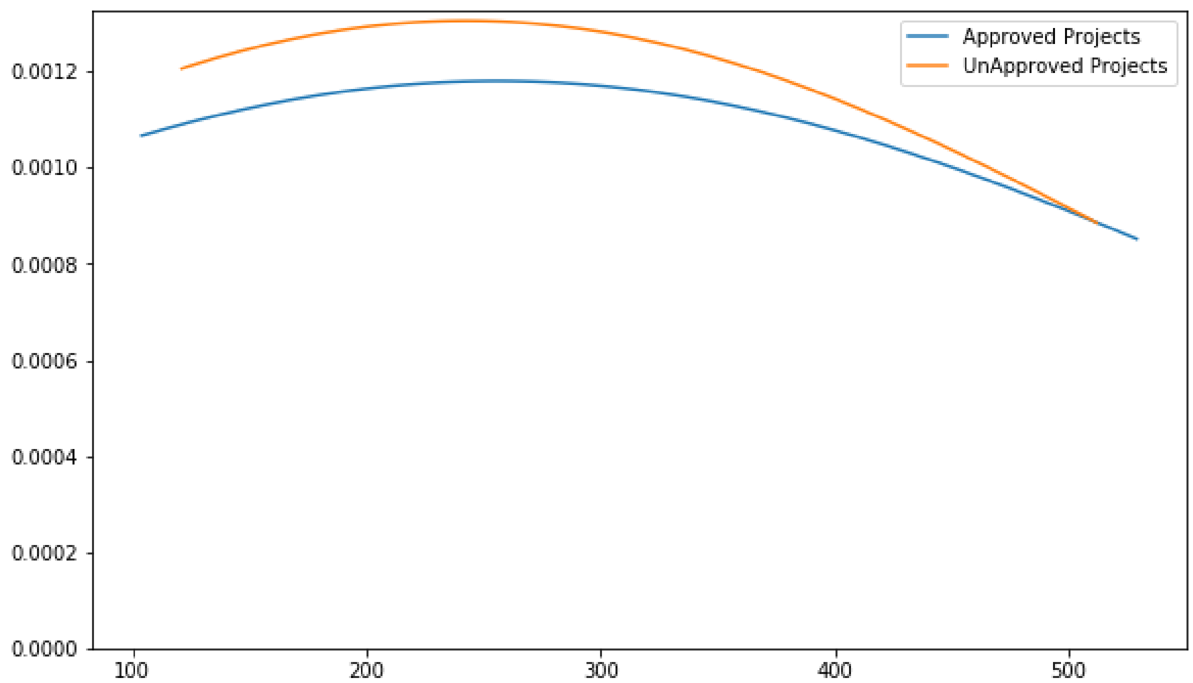
```
approvedNumberOfProjects = projectsData[projectsData.project_is_approve
d == 1]['project_essay'].str.split().apply(len);
approvedNumberOfProjects = approvedNumberOfProjects.values
unApprovedNumberOfProjects = projectsData[projectsData.project_is_appro
ved == 0]['project_essay'].str.split().apply(len);
```

```
unApprovedNumberOfProjects = unApprovedNumberOfProjects.values
plt.boxplot([approvedNumberOfProjects, unApprovedNumberOfProjects]);
plt.grid();
plt.xticks([1, 2], ['Approved Projects', 'UnApproved Projects']);
plt.ylabel('Number of words in project essay');
plt.show();
```



In [35]:

```
plt.figure(figsize = (10, 6));
sbrn.kdeplot(approvedNumberOfProjects, label = "Approved Projects", bw
= 5);
sbrn.kdeplot(unApprovedNumberOfProjects, label = "UnApproved Projects",
 bw = 5);
plt.legend();
plt.show();
```



Observation:

1. The approved and rejected projects overlap largely when plotted based on number of words in project_essay. So we cannot predict any observation which will be useful for classification.

Univariate Analysis: price

In [36]:
```
projectsData.head(5)
```

Out[36]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

In [37]:
```
resourcesData.head(5)
```

Out[37]:

	id	description	quantity	price

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [38]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframe
s-indexes-for-all-groups-in-one-step
priceAndQuantityData = resourcesData.groupby('id').agg({'price': 'sum',
 'quantity': 'sum'}).reset_index();
priceAndQuantityData.head(5)
```

Out[38]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98
4	p000005	485.99	8

In [39]:

```
projectsData.shape
```

Out[39]: (109248, 20)

In [40]:

```
projectsData = pd.merge(projectsData, priceAndQuantityData, on = 'id',
how = 'left');
print(projectsData.shape);
projectsData.head(3)
```

(109248, 22)

Out[40]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ

In [41]:
```
projectsData[projectsData['id'] == 'p253737']
```

Out[41]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	p
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2

In [42]:

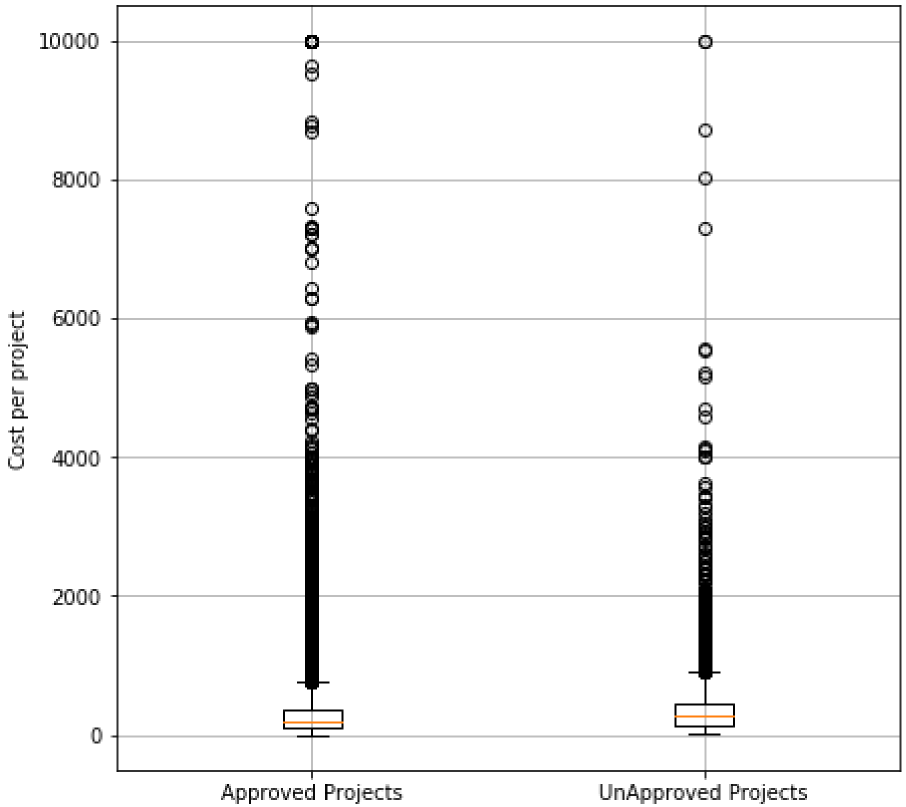
```
priceAndQuantityData[priceAndQuantityData['id'] == 'p253737']
```

Out[42]:

	id	price	quantity
253736	p253737	154.6	23

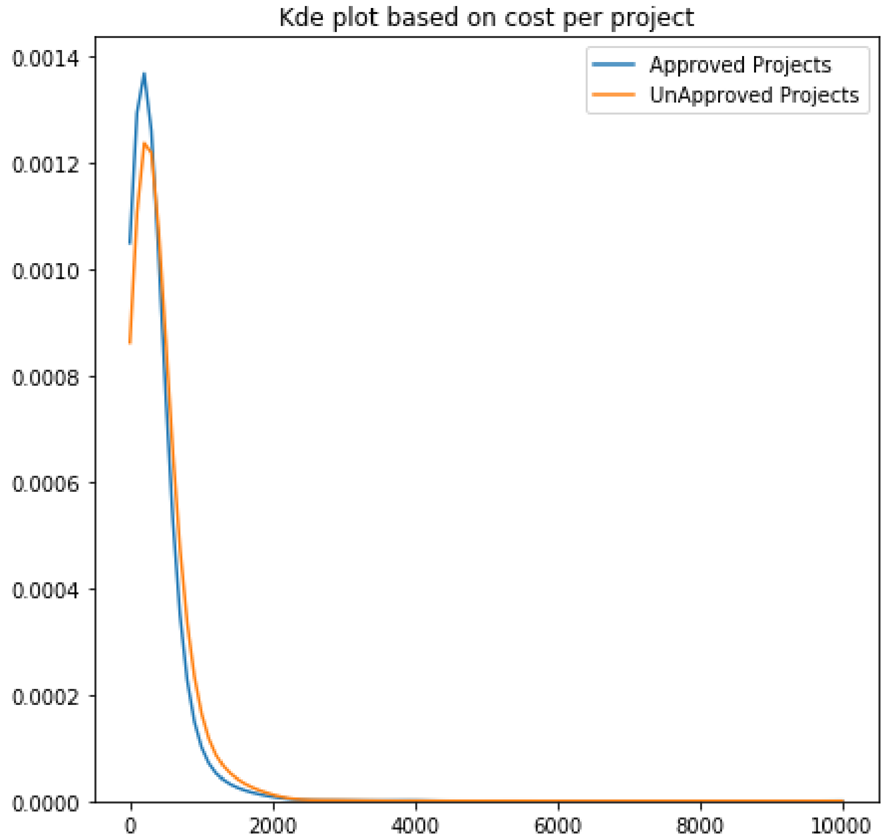
In [43]:

```
approvedProjectsPrice = projectsData[projectsData['project_is_approved'
] == 1].price;
unApprovedProjectsPrice = projectsData[projectsData['project_is_approve
d'] == 0].price;
plt.boxplot([approvedProjectsPrice, unApprovedProjectsPrice]);
plt.grid();
plt.xticks([1, 2], ['Approved Projects', 'UnApproved Projects']);
plt.ylabel('Cost per project');
plt.show();
```



In [44]:

```
plt.title("Kde plot based on cost per project");
sbrn.kdeplot(approvedProjectsPrice, label = "Approved Projects", bw =
0.6);
sbrn.kdeplot(unApprovedProjectsPrice, label = "UnApproved Projects", bw
 = 0.6);
plt.legend();
plt.show();
```



In [45]:
```
pricePercentilesApproved = [round(np.percentile(approvedProjectsPrice, percentile), 3) for percentile in np.arange(0, 100, 5)];
pricePercentilesUnApproved = [round(np.percentile(unApprovedProjectsPrice, percentile), 3) for percentile in np.arange(0, 100, 5)];
percentileValuePricesData = pd.DataFrame({'Percentile': np.arange(0, 100, 5), 'Approved projects': pricePercentilesApproved, 'UnApproved Projects': pricePercentilesUnApproved});
percentileValuePricesData
```

Out[45]:

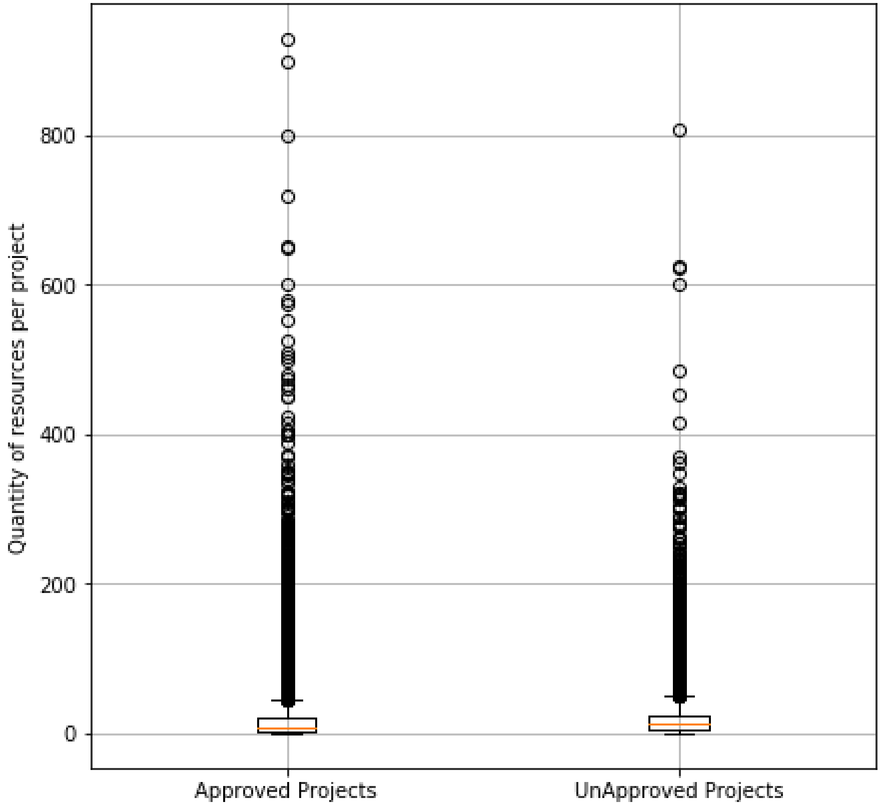
	Percentile	Approved projects	UnApproved Projects

	Percentile	Approved projects	UnApproved Projects
0	0	0.660	1.970
1	5	13.590	41.900
2	10	33.880	73.670
3	15	58.000	99.109
4	20	77.380	118.560
5	25	99.950	140.892
6	30	116.680	162.230
7	35	137.232	184.014
8	40	157.000	208.632
9	45	178.265	235.106
10	50	198.990	263.145
11	55	223.990	292.610
12	60	255.630	325.144
13	65	285.412	362.390
14	70	321.225	399.990
15	75	366.075	449.945
16	80	411.670	519.282
17	85	479.000	618.276
18	90	593.110	739.356
19	95	801.598	992.486

Observation:

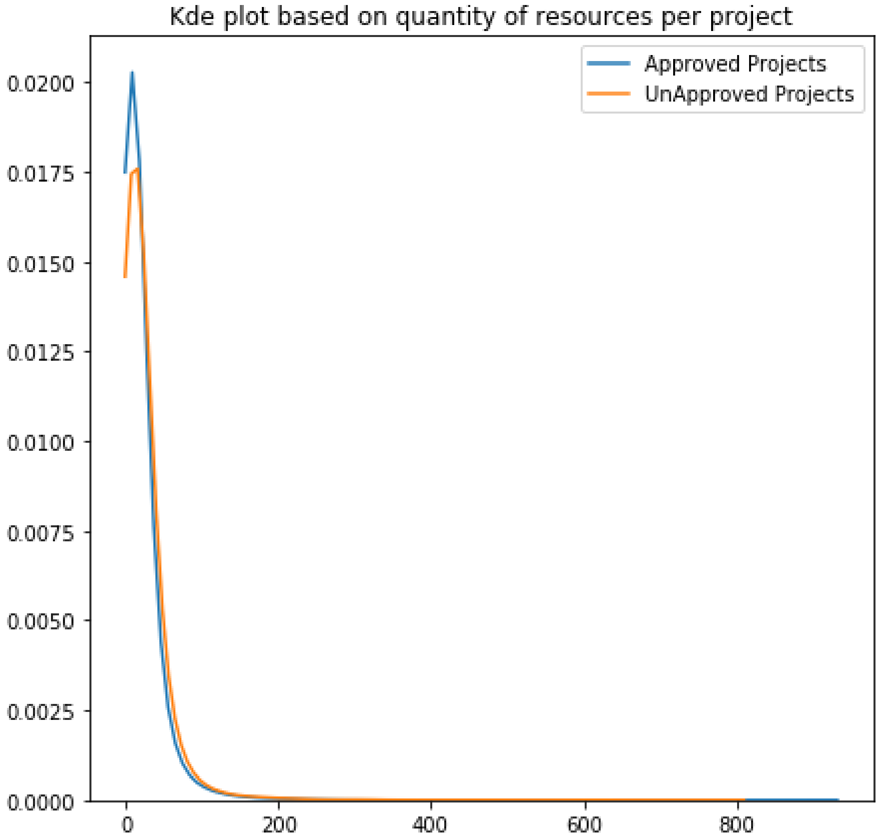
1. Most of the projects proposed are of less cost.

In [46]:
```
approvedProjectsQuantity = projectsData[projectsData['project_is_approv
ed'] == 1].quantity;
unApprovedProjectsQuantity = projectsData[projectsData['project_is_appr
oved'] == 0].quantity;
plt.boxplot([approvedProjectsQuantity, unApprovedProjectsQuantity]);
plt.grid();
plt.xticks([1, 2], ['Approved Projects', 'UnApproved Projects']);
plt.ylabel('Quantity of resources per project');
plt.show();
```



In [47]:
```
plt.title("Kde plot based on quantity of resources per project");
```

```
sbrn.kdeplot(approvedProjectsQuantity, label = "Approved Projects", bw
= 0.6);
sbrn.kdeplot(unApprovedProjectsQuantity, label = "UnApproved Projects",
 bw = 0.6);
plt.legend();
plt.show();
```



In [48]:
```
quantityPercentilesApproved = [round(np.percentile(approvedProjectsQuan
tity, percentile), 3) for percentile in np.arange(0, 100, 5)];
quantityPercentilesUnApproved = [round(np.percentile(unApprovedProjects
Quantity, percentile), 3) for percentile in np.arange(0, 100, 5)];
percentileValueQuantitiesData = pd.DataFrame({'Percentile': np.arange(0
, 100, 5), 'Approved projects': quantityPercentilesApproved, 'UnApprove
```

```
d Projects': quantityPercentilesUnApproved});
percentileValueQuantitiesData
```

Out[48]:

	Percentile	Approved projects	UnApproved Projects
0	0	1.0	1.0
1	5	1.0	2.0
2	10	1.0	3.0
3	15	2.0	4.0
4	20	3.0	5.0
5	25	3.0	6.0
6	30	4.0	7.0
7	35	5.0	8.0
8	40	6.0	9.0
9	45	7.0	10.0
10	50	8.0	12.0
11	55	10.0	13.0
12	60	11.0	15.0
13	65	14.0	18.0
14	70	16.0	20.0
15	75	20.0	24.0
16	80	25.0	29.0
17	85	30.0	35.0
18	90	38.0	45.0
19	95	56.0	63.0

In [49]:
```
sbrn.set_style('whitegrid');
sbrn.FacetGrid(projectsData, hue = 'project_is_approved', size = 6) \
    .map(plt.scatter, 'price', 'quantity') \
    .add_legend();
plt.title("Scatter plot between price and quantity based project approv
al and rejection");
plt.show();
```

Observation:

1. When plotted scatter plot between approved and rejected projects based on price and quantity there is huge overlap. So the projects approval is not actually depending on price and quantity resources of the project.

Univariate Analysis: teacher_number_of_previously_posted_projects

In [50]:
```
projectsData.head(5)
```

Out[50]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

In [51]:

```
previouslyPostedApprovedNumberData = projectsData.groupby('teacher_numb
er_of_previously_posted_projects')['project_is_approved'].agg(lambda x:
 x.eq(1).sum()).reset_index();
previouslyPostedRejectedNumberData = projectsData.groupby('teacher_numb
er_of_previously_posted_projects')['project_is_approved'].agg(lambda x:
 x.eq(0).sum()).reset_index();
print("Total number of projects approved: ", len(projectsData[projectsD
ata['project_is_approved'] == 1]));
print("Total number of projects rejected: ", len(projectsData[projectsD
ata['project_is_approved'] == 0]));
print("Number of projects approved categorized by previously_posted: ",
 previouslyPostedApprovedNumberData['project_is_approved'].sum());
print("Number of projects rejected categorized by previously_posted: ",
 previouslyPostedRejectedNumberData['project_is_approved'].sum());
previouslyPostedNumberData = pd.merge(previouslyPostedApprovedNumberDat
a, previouslyPostedRejectedNumberData, on = 'teacher_number_of_previous
ly_posted_projects', how = 'inner');
previouslyPostedNumberData.head(5)
```

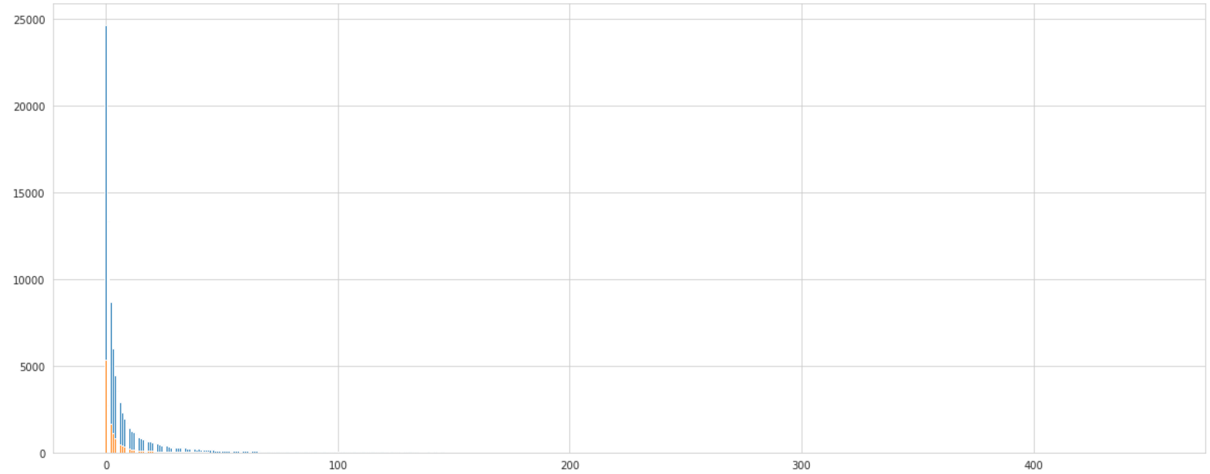
```
Total number of projects approved:  92706
Total number of projects rejected:  16542
Number of projects approved categorized by previously_posted:  92706
Number of projects rejected categorized by previously_posted:  16542
```

Out[51]:

	teacher_number_of_previously_posted_projects	project_is_approved_x	project_is_ap
0	0	24652	5362

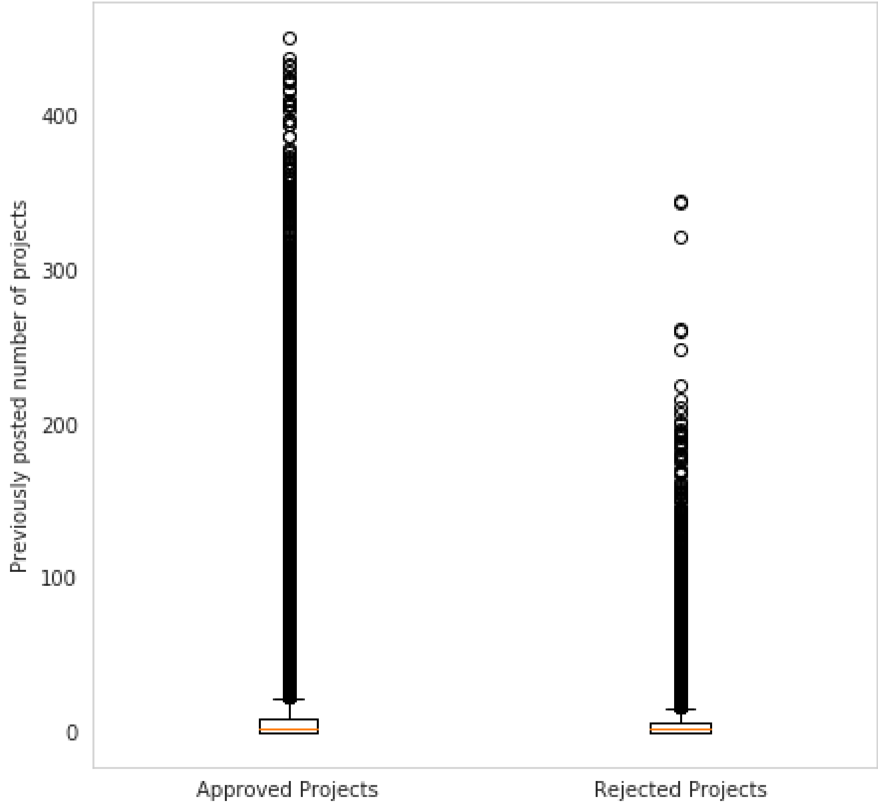
	teacher_number_of_previously_posted_projects	project_is_approved_x	project_is_ap
1	1	13329	2729
2	2	8705	1645
3	3	5997	1113
4	4	4452	814

In [52]:
```
plt.figure(figsize = (20, 8));
plt.bar(previouslyPostedNumberData.teacher_number_of_previously_posted_
projects, previouslyPostedNumberData.project_is_approved_x);
plt.bar(previouslyPostedNumberData.teacher_number_of_previously_posted_
projects, previouslyPostedNumberData.project_is_approved_y);
plt.show();
```

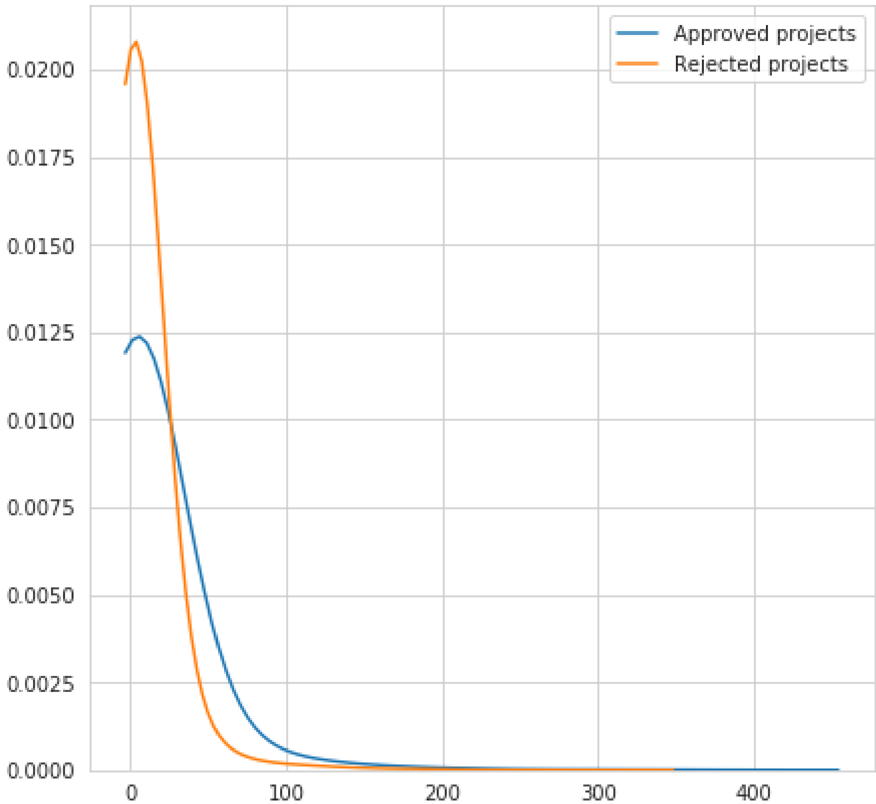


In [53]:
```
previouslyPostedApprovedData = projectsData[projectsData['project_is_ap
proved'] == 1].teacher_number_of_previously_posted_projects;
previouslyPostedRejectedData = projectsData[projectsData['project_is_ap
proved'] == 0].teacher_number_of_previously_posted_projects;
plt.boxplot([previouslyPostedApprovedData, previouslyPostedRejectedData
]);
```

```
plt.grid();
plt.xticks([1, 2], ['Approved Projects', 'Rejected Projects']);
plt.ylabel('Previously posted number of projects');
plt.show();
```



In [54]:
```
sbrn.kdeplot(previouslyPostedApprovedData, label = "Approved projects",
 bw = 1);
sbrn.kdeplot(previouslyPostedRejectedData, label = "Rejected projects",
 bw = 1);
plt.show();
```



Observation:

1. Most of the projects approved and rejected are with less number of teacher_number_of_previously_posted_projects. So the approval is not much depending on how many number of projects proposed by teacher previously.

In [0]:
```
def stringContainsNumbers(string):
    return any([character.isdigit() for character in string])
```

In [56]:
```
numericResourceApprovedData = projectsData[(projectsData['project_resou
rce_summary'].apply(stringContainsNumbers) == True) & (projectsData['pr
```

```
oject_is_approved'] == 1)]
textResourceApprovedData = projectsData[(projectsData['project_resource
_summary'].apply(stringContainsNumbers) == False) & (projectsData['proj
ect_is_approved'] == 1)]
numericResourceRejectedData = projectsData[(projectsData['project_resou
rce_summary'].apply(stringContainsNumbers) == True) & (projectsData['pr
oject_is_approved'] == 0)]
textResourceRejectedData = projectsData[(projectsData['project_resource
_summary'].apply(stringContainsNumbers) == False) & (projectsData['proj
ect_is_approved'] == 0)]
print("Checking whether numbers in resource summary will be useful for
 project approval?");
equalsBorder(70);
print("Number of approved projects with numbers in resource summary: ",
 numericResourceApprovedData.shape[0]);
print("Number of rejected projects with numbers in resource summary: ",
 numericResourceRejectedData.shape[0]);
print("Number of approved projects without numbers in resource summary:
 ", textResourceApprovedData.shape[0]);
print("Number of rejected projects without numbers in resource summary:
 ", textResourceRejectedData.shape[0]);
```

```
Checking whether numbers in resource summary will be useful for project
approval?
======================================================================
Number of approved projects with numbers in resource summary:  14090
Number of rejected projects with numbers in resource summary:  1666
Number of approved projects without numbers in resource summary:  78616
Number of rejected projects without numbers in resource summary:  14876
```

Observation:

1. The rejection rate of project is less when projects resource summary has numbers in it.
2. Even the number of projects approved without numbers in resource summary is high which means that the classification does not actually depends on whether resource summary contains numerical digits or not.

Conclusion of univariate analysis:

1. There is huge overlap of approved and rejected projects when taken for all single featurs. So, this project cannot be classified using single features.
2. project_title is some what better in text type of feature because of less overlap than others.
3. The project approval is not depending on resources cost, but the probability of project rejection is more when resources cost is more.

Preprocessing data

In [0]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
t'
# All stopwords that are needed to be removed in the text
stopWords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'o
urs', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselve
s', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
s', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'th
is', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'h
ave', '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', 'h
ow', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 's
o', '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', "is
n'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"]);
def preProcessingWithAndWithoutStopWords(texts):
    """
    This function takes list of texts and returns preprocessed list of
 texts one with
    stop words and one without stopwords.
    """
    # Variable for storing preprocessed text with stop words
    preProcessedTextsWithStopWords = [];
    # Variable for storing preprocessed text without stop words
    preProcessedTextsWithoutStopWords = [];

    # Looping over list of texts for performing pre processing
    for text in tqdm(texts, total = len(texts)):
        # Removing all links in the text
        text = re.sub(r"http\S+", "", text);

        # Removing all html tags in the text
        text = re.sub(r"<\w+/>", "", text);
        text = re.sub(r"<\w+>", "", text);

        # https://stackoverflow.com/a/47091490/4084039
        # Replacing all below words with adverbs
        text = re.sub(r"won't", "will not", text)
        text = re.sub(r"can\'t", "can not", text)
        text = re.sub(r"n\'t", " not", text)
        text = re.sub(r"\'re", " are", text)
        text = re.sub(r"\'s", " is", text)
        text = re.sub(r"\'d", " would", text)
        text = re.sub(r"\'ll", " will", text)
        text = re.sub(r"\'t", " not", text)
```

```
        text = re.sub(r"\'ve", " have", text)
        text = re.sub(r"\'m", " am", text)

        # Removing backslash symbols in text
        text = text.replace('\\r', ' ');
        text = text.replace('\\n', ' ');
        text = text.replace('\\"', ' ');

        # Removing all special characters of text
        text = re.sub(r"[^a-zA-Z0-9]+", " ", text);

        # Converting whole review text into lower case
        text = text.lower();

        # adding this preprocessed text without stopwords to list
        preProcessedTextsWithStopWords.append(text);

        # removing stop words from text
        textWithoutStopWords = ' '.join([word for word in text.split()
if word not in stopWords]);
        # adding this preprocessed text without stopwords to list
        preProcessedTextsWithoutStopWords.append(textWithoutStopWords);

    return [preProcessedTextsWithStopWords, preProcessedTextsWithoutSto
pWords];
```

In [58]:

```
texts = [projectsData['project_essay'].values[0]]
preProcessedTextsWithStopWords, preProcessedTextsWithoutStopWords = pre
ProcessingWithAndWithoutStopWords(texts);
print("Example project essay without pre-processing: ");
equalsBorder(70);
print(texts);
equalsBorder(70);
print("Example project essay with stop words and pre-processing: ");
equalsBorder(70);
print(preProcessedTextsWithStopWords);
equalsBorder(70);
print("Example project essay without stop words and pre-processing: ");
```

```
equalsBorder(70);
print(preProcessedTextsWithoutStopWords);
```

```
Example project essay without pre-processing:
======================================================================
['My students are English learners that are working on English as their
second or third languages. We are a melting pot of refugees, immigrant
s, and native-born Americans bringing the gift of language to our schoo
l. \\r\\n\\r\\n We have over 24 languages represented in our English Le
arner program with students at every level of mastery.  We also have ov
er 40 countries represented with the families within our school.  Each
student brings a wealth of knowledge and experiences to us that open ou
r eyes to new cultures, beliefs, and respect.\\"The limits of your lang
uage are the limits of your world.\\"-Ludwig Wittgenstein  Our English
learner\'s have a strong support system at home that begs for more reso
urces.  Many times our parents are learning to read and speak English a
long side of their children.  Sometimes this creates barriers for paren
ts to be able to help their child learn phonetics, letter recognition,
and other reading skills.\\r\\n\\r\\nBy providing these dvd\'s and play
ers, students are able to continue their mastery of the English languag
e 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 o
f this program.  These educational videos will be specially chosen by t
he English Learner Teacher and will be sent home regularly to watch.  T
he 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 opport
unity to check out a dvd player to use for the year.  The plan is to us
e these videos and educational dvd\'s for the years to come for other E
L students.\\r\\nnannan']
======================================================================
Example project essay with stop words and pre-processing:
======================================================================
['my students are english learners that are working on english as their
second or third languages we are a melting pot of refugees immigrants a
nd native born americans bringing the gift of language to our school we
have over 24 languages represented in our english learner program with
students at every level of mastery we also have over 40 countries repre
sented with the families within our school each student brings a wealth
of knowledge and experiences to us that open our eyes to new cultures b
eliefs and respect the limits of your language are the limits of your w
```

```
eliefs and respect the limits of your language are the limits of your w
orld ludwig wittgenstein our english learner is have a strong support s
ystem at home that begs for more resources many times our parents are l
earning to read and speak english along side of their children sometime
s this creates barriers for parents to be able to help their child lear
n phonetics letter recognition and other reading skills by providing th
ese dvd is and players students are able to continue their mastery of t
he english language even if no one at home is able to assist all famili
es with students within the level 1 proficiency status will be a offere
d to be a part of this program these educational videos will be special
ly chosen by the english learner teacher and will be sent home regularl
y to watch the videos are to help the child develop early reading skill
s parents that do not have access to a dvd player will have the opportu
nity to check out a dvd player to use for the year the plan is to use t
hese videos and educational dvd is for the years to come for other el s
tudents nannan']
=====================================================================
Example project essay without stop words and pre-processing:
=====================================================================
['students english learners working english second third languages melt
ing pot refugees immigrants native born americans bringing gift languag
e school 24 languages represented english learner program students ever
y level mastery also 40 countries represented families within school st
udent brings wealth knowledge experiences us open eyes new cultures bel
iefs respect limits language limits world ludwig wittgenstein english l
earner strong support system home begs resources many times parents lea
rning read speak english along side children sometimes creates barriers
parents able help child learn phonetics letter recognition reading skil
ls providing dvd players students able continue mastery english languag
e even no one home able assist families students within level 1 profici
ency status offered part program educational videos specially chosen en
glish learner teacher sent home regularly watch videos help child devel
op early reading skills parents not access dvd player opportunity check
dvd player use year plan use videos educational dvd years come el stude
nts nannan']
```

In [59]:

```
projectEssays = projectsData['project_essay'];
preProcessedEssaysWithStopWords, preProcessedEssaysWithoutStopWords = p
reProcessingWithAndWithoutStopWords(projectEssays);
```

In [60]:
```
preProcessedEssaysWithoutStopWords[0:3]
```

Out[60]:
```
['students english learners working english second third languages melt
ing pot refugees immigrants native born americans bringing gift languag
e school 24 languages represented english learner program students ever
y level mastery also 40 countries represented families within school st
udent brings wealth knowledge experiences us open eyes new cultures bel
iefs respect limits language limits world ludwig wittgenstein english l
earner strong support system home begs resources many times parents lea
rning read speak english along side children sometimes creates barriers
parents able help child learn phonetics letter recognition reading skil
ls providing dvd players students able continue mastery english languag
e even no one home able assist families students within level 1 profici
ency status offered part program educational videos specially chosen en
glish learner teacher sent home regularly watch videos help child devel
op early reading skills parents not access dvd player opportunity check
dvd player use year plan use videos educational dvd years come el stude
nts nannan',
 'students arrive school eager learn polite generous strive best know e
ducation succeed life help improve lives school focuses families low in
comes tries give student education deserve not much students use materi
als given best projector need school crucial academic improvement stude
nts technology continues grow many resources internet teachers use grow
th students however school limited resources particularly technology wi
thout disadvantage one things could really help classrooms projector pr
ojector not crucial instruction also growth students projector show pre
sentations documentaries photos historical land sites math problems muc
h projector make teaching learning easier also targeting different type
s learners classrooms auditory visual kinesthetic etc nannan',
 'true champions not always ones win guts mia hamm quote best describes
students cholla middle school approach playing sports especially girls
boys soccer teams teams made 7th 8th grade students not opportunity pla
y organized sport due family financial difficulties teach title one mid
dle school urban neighborhood 74 students qualify free reduced lunch ma
ny come activity sport opportunity poor homes students love participate
sports learn new skills apart team atmosphere school lacks funding meet
students needs concerned lack exposure not prepare participating sports
```

```
teams high school end school year goal provide students opportunity lea
rn variety soccer skills positive qualities person actively participate
s team students campus come school knowing face uphill battle comes par
ticipating organized sports players would thrive field confidence appro
priate soccer equipment play soccer best abilities students experience
helpful person part team teaches positive supportive encouraging others
students using soccer equipment practice games daily basis learn practi
ce necessary skills develop strong soccer team experience create opport
unity students learn part team positive contribution teammates students
get opportunity learn practice variety soccer skills use skills game ac
cess type experience nearly impossible without soccer equipment student
s players utilize practice games nannan']
```

In [61]:

```
projectTitles = projectsData['project_title'];
preProcessedProjectTitlesWithStopWords, preProcessedProjectTitlesWithou
tStopWords = preProcessingWithAndWithoutStopWords(projectTitles);
preProcessedProjectTitlesWithoutStopWords[0:5]
```

Out[61]:

```
['educational support english learners home',
 'wanted projector hungry learners',
 'soccer equipment awesome middle school students',
 'techie kindergarteners',
 'interactive math tools']
```

In [62]:

```
projectsData['preprocessed_titles'] = preProcessedProjectTitlesWithoutS
topWords;
projectsData['preprocessed_essays'] = preProcessedEssaysWithoutStopWord
s;
projectsData.shape
```

Out[62]:

```
(109248, 24)
```

Preparing data for classification and modelling

In [0]:

```
pd.DataFrame(projectsData.columns, columns = ['All features in projects
 data'])
```

Out[0]:

	All features in projects data
0	Unnamed: 0
1	id
2	teacher_id
3	teacher_prefix
4	school_state
5	project_submitted_datetime
6	project_grade_category
7	project_subject_categories
8	project_subject_subcategories
9	project_title
10	project_essay_1
11	project_essay_2
12	project_essay_3
13	project_essay_4
14	project_resource_summary
15	teacher_number_of_previously_posted_projects
16	project_is_approved
17	cleaned_categories
18	cleaned_sub_categories
19	project_essay
20	price

	All features in projects data
21	quantity
22	preprocessed_titles
23	preprocessed_essays

Useful features:

Here we will consider only below features for classification and we can ignore the other features

Categorical data:

1. **school_state** - categorical data
2. **project_grade_category** - categorical data
3. **cleaned_categories** - categorical data
4. **cleaned_sub_categories** - categorical data
5. **teacher_prefix** - categorical data

Text data:

1. **project_resource_summary** - text data
2. **project_title - text data**
3. **project_resource_summary** - text data

Numerical data:

1. **teacher_number_of_previously_posted_projects** - numerical data
2. **price** - numerical data
3. **quantity** - numerical data

Vectorizing categorical data

1. Vectorizing cleaned_categories(project_subject_categories cleaned) - One Hot Encoding

In [0]:
```
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique cleaned_categories
subjectsCategoriesVectorizer = CountVectorizer(vocabulary = list(sorted
CategoriesDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with cleaned_categories values
subjectsCategoriesVectorizer.fit(projectsData['cleaned_categories'].val
ues);
# Vectorizing categories using one-hot-encoding
categoriesVectors = subjectsCategoriesVectorizer.transform(projectsData
['cleaned_categories'].values);
```

In [0]:
```
print("Features used in vectorizing categories: ");
equalsBorder(70);
print(subjectsCategoriesVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of cleaned_categories matrix after vectorization(one-hot-e
ncoding): ", categoriesVectors.shape);
equalsBorder(70);
print("Sample vectors of categories: ");
equalsBorder(70);
print(categoriesVectors[0:4])
```

```
Features used in vectorizing categories:
======================================================================
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearn
ing', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Langua
ge']
======================================================================
Shape of cleaned_categories matrix after vectorization(one-hot-encodin
g):  (109248, 9)
======================================================================
Sample vectors of categories:
```

```
======================================================================
  (0, 8)	1
  (1, 2)	1
  (1, 6)	1
  (2, 6)	1
  (3, 7)	1
  (3, 8)	1
```

2. Vectorizing cleaned_sub_categories(project_subject_sub_categories cleaned) - One Hot Encoding

In [0]:

```
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique cleaned_sub_categories
subjectsSubCategoriesVectorizer = CountVectorizer(vocabulary = list(sor
tedDictionarySubCategories.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with cleaned_sub_categories values
subjectsSubCategoriesVectorizer.fit(projectsData['cleaned_sub_categorie
s'].values);
# Vectorizing sub categories using one-hot-encoding
subCategoriesVectors = subjectsSubCategoriesVectorizer.transform(projec
tsData['cleaned_sub_categories'].values);
```

In [0]:

```
print("Features used in vectorizing subject sub categories: ");
equalsBorder(70);
print(subjectsSubCategoriesVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of cleaned_categories matrix after vectorization(one-hot-e
ncoding): ", subCategoriesVectors.shape);
equalsBorder(70);
print("Sample vectors of categories: ");
equalsBorder(70);
print(subCategoriesVectors[0:4])
```

```
Features used in vectorizing subject sub categories:
======================================================================
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolveme
```

```
nt', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutri
tionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingA
rts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPre
p', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopme
nt', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Healt
h_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing',
'Mathematics', 'Literacy']
======================================================================
Shape of cleaned_categories matrix after vectorization(one-hot-encodin
g):  (109248, 30)
======================================================================
Sample vectors of categories:
======================================================================
  (0, 20)	1
  (0, 29)	1
  (1, 5)	1
  (1, 13)	1
  (2, 13)	1
  (2, 24)	1
  (3, 28)	1
  (3, 29)	1
```

3. Vectorizing teacher_prefix - One Hot Encoding

In [0]:
```
def giveCounter(data):
    counter = Counter();
    for dataValue in data:
        counter.update(str(dataValue).split());
    return counter
```

In [0]:
```
giveCounter(projectsData['teacher_prefix'].values)
```

Out[0]:
```
Counter({'Mrs.': 57269,
         'Mr.': 10648,
         'Ms.': 38955,
         'Teacher': 2360,
```

```
            'nan': 3,
            'Dr.': 13})
```

In [0]:
```
projectsData = projectsData.dropna(subset = ['teacher_prefix']);
projectsData.shape
```

Out[0]:
```
(109245, 22)
```

In [0]:
```
teacherPrefixDictionary = dict(giveCounter(projectsData['teacher_prefi
x'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique teacher_prefix
teacherPrefixVectorizer = CountVectorizer(vocabulary = list(teacherPref
ixDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with teacher_prefix values
teacherPrefixVectorizer.fit(projectsData['teacher_prefix'].values);
# Vectorizing teacher_prefix using one-hot-encoding
teacherPrefixVectors = teacherPrefixVectorizer.transform(projectsData[
'teacher_prefix'].values);
```

In [0]:
```
print("Features used in vectorizing teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of teacher_prefix matrix after vectorization(one-hot-encod
ing): ", teacherPrefixVectors.shape);
equalsBorder(70);
print("Sample vectors of teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectors[0:100]);
```

```
Features used in vectorizing teacher_prefix:
======================================================================
['Mrs.', 'Mr.', 'Ms.', 'Teacher', 'Dr.']
======================================================================
Shape of teacher_prefix matrix after vectorization(one-hot-encoding):
(109245, 5)
======================================================================
Sample vectors of teacher_prefix:
```

```
=====================================================================
  (27, 3)       1
  (75, 3)       1
  (82, 3)       1
  (88, 3)       1
```

In [0]:
```
teacherPrefixes = [prefix.replace('.', '') for prefix in projectsData[
'teacher_prefix'].values];
teacherPrefixes[0:5]
```

Out[0]:
```
['Mrs', 'Mr', 'Ms', 'Mrs', 'Mrs']
```

In [0]:
```
projectsData['teacher_prefix'] = teacherPrefixes;
projectsData.head(3)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms	AZ

3 rows × 22 columns

In [0]:
```
teacherPrefixDictionary = dict(giveCounter(projectsData['teacher_prefi
x'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique teacher_prefix
teacherPrefixVectorizer = CountVectorizer(vocabulary = list(teacherPref
ixDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with teacher_prefix values
teacherPrefixVectorizer.fit(projectsData['teacher_prefix'].values);
# Vectorizing teacher_prefix using one-hot-encoding
teacherPrefixVectors = teacherPrefixVectorizer.transform(projectsData[
'teacher_prefix'].values);
```

In [0]:
```
print("Features used in vectorizing teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of teacher_prefix matrix after vectorization(one-hot-encod
ing): ", teacherPrefixVectors.shape);
equalsBorder(70);
print("Sample vectors of teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectors[0:4]);
```

```
Features used in vectorizing teacher_prefix:
======================================================================
['Mrs', 'Mr', 'Ms', 'Teacher', 'Dr']
======================================================================
Shape of teacher_prefix matrix after vectorization(one-hot-encoding):
(109245, 5)
======================================================================
Sample vectors of teacher_prefix:
======================================================================
  (0, 0)        1
  (1, 1)        1
  (2, 2)        1
  (3, 0)        1
```

4. Vectorizing school_state - One Hot Encoding

In [0]:
```
schoolStateDictionary = dict(giveCounter(projectsData['school_state'].v
alues));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique school states
schoolStateVectorizer = CountVectorizer(vocabulary = list(schoolStateDi
ctionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with school_state values
schoolStateVectorizer.fit(projectsData['school_state'].values);
# Vectorizing school_state using one-hot-encoding
schoolStateVectors = schoolStateVectorizer.transform(projectsData['scho
ol_state'].values);
```

In [0]:
```
print("Features used in vectorizing school_state: ");
equalsBorder(70);
print(schoolStateVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of school_state matrix after vectorization(one-hot-encodin
g): ", schoolStateVectors.shape);
equalsBorder(70);
print("Sample vectors of school_state: ");
equalsBorder(70);
print(schoolStateVectors[0:4]);
```

```
Features used in vectorizing school_state:
======================================================================
['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY', 'O
K', 'MA', 'NV', 'OH', 'PA', 'AL', 'LA', 'VA', 'AR', 'WA', 'WV', 'ID',
'TN', 'MS', 'CO', 'UT', 'IL', 'MI', 'HI', 'IA', 'RI', 'NJ', 'MO', 'DE',
'MN', 'ME', 'WY', 'ND', 'OR', 'AK', 'MD', 'WI', 'SD', 'NE', 'NM', 'DC',
'KS', 'MT', 'NH', 'VT']
======================================================================
Shape of school_state matrix after vectorization(one-hot-encoding):  (1
09245, 51)
======================================================================
Sample vectors of school_state:
```

```
==========================================================================
  (0, 0)        1
  (1, 1)        1
  (2, 2)        1
  (3, 3)        1
```

5. Vectorizing project_grade_category - One Hot Encoding

In [0]:
```
giveCounter(projectsData['project_grade_category'])
```

Out[0]:
```
Counter({'Grades': 109245,
         'PreK-2': 44225,
         '6-8': 16923,
         '3-5': 37135,
         '9-12': 10962})
```

In [0]:
```
cleanedGrades = []
for grade in projectsData['project_grade_category'].values:
    grade = grade.replace(' ', '');
    grade = grade.replace('-', 'to');
    cleanedGrades.append(grade);
cleanedGrades[0:4]
```

Out[0]:
```
['GradesPreKto2', 'Grades6to8', 'Grades6to8', 'GradesPreKto2']
```

In [0]:
```
projectsData['project_grade_category'] = cleanedGrades
projectsData.head(4)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs	KY

4 rows × 22 columns

In [0]:

```
projectGradeDictionary = dict(giveCounter(projectsData['project_grade_c
ategory'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique project grade categories
projectGradeVectorizer = CountVectorizer(vocabulary = list(projectGrade
Dictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with project_grade_category values
projectGradeVectorizer.fit(projectsData['project_grade_category'].value
s);
```

```
# Vectorizing project_grade_category using one-hot-encoding
projectGradeVectors = projectGradeVectorizer.transform(projectsData['pr
oject_grade_category'].values);
```

In [0]:
```
print("Features used in vectorizing project_grade_category: ");
equalsBorder(70);
print(projectGradeVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of school_state matrix after vectorization(one-hot-encodin
g): ", projectGradeVectors.shape);
equalsBorder(70);
print("Sample vectors of school_state: ");
equalsBorder(70);
print(projectGradeVectors[0:4]);
```

```
Features used in vectorizing project_grade_category:
======================================================================
['GradesPreKto2', 'Grades6to8', 'Grades3to5', 'Grades9to12']
======================================================================
Shape of school_state matrix after vectorization(one-hot-encoding):  (1
09245, 4)
======================================================================
Sample vectors of school_state:
======================================================================
  (0, 0)	1
  (1, 1)	1
  (2, 1)	1
  (3, 0)	1
```

In [0]:
```
projectsDataSub = projectsData[0:40000];
preProcessedEssaysWithoutStopWordsSub = preProcessedEssaysWithoutStopWo
rds[0:40000];
preProcessedProjectTitlesWithoutStopWordsSub = preProcessedProjectTitle
sWithoutStopWords[0:40000];
```

Vectorizing Text Data

Bag of Words

1. Vectorizing project_essay

In [0]:
```
# Initializing countvectorizer for bag of words vectorization of prepro
cessed project essays
bowEssayVectorizer = CountVectorizer(min_df = 10);
# Transforming the preprocessed essays to bag of words vectors
bowEssayModel = bowEssayVectorizer.fit_transform(preProcessedEssaysWith
outStopWordsSub);
```

In [0]:
```
print("Some of the Features used in vectorizing preprocessed essays: "
);
equalsBorder(70);
print(bowEssayVectorizer.get_feature_names()[-40:]);
equalsBorder(70);
print("Shape of preprocessed essay matrix after vectorization: ", bowEs
sayModel.shape);
equalsBorder(70);
print("Sample bag-of-words vector of preprocessed essay: ");
equalsBorder(70);
print(bowEssayModel[0])
```

```
Some of the Features used in vectorizing preprocessed essays:
======================================================================
['yeats', 'yell', 'yelling', 'yellow', 'yemen', 'yes', 'yesterday', 'ye
t', 'yield', 'yields', 'yoga', 'york', 'younannan', 'young', 'younger',
'youngest', 'youngsters', 'youth', 'youthful', 'youths', 'youtube', 'yu
mmy', 'zeal', 'zearn', 'zen', 'zenergy', 'zero', 'zest', 'zip', 'ziplo
c', 'zippers', 'zipping', 'zone', 'zoned', 'zones', 'zoo', 'zoom', 'zoo
ming', 'zoos', 'zumba']
======================================================================
Shape of preprocessed essay matrix after vectorization:  (40000, 11077)
======================================================================
Sample bag-of-words vector of preprocessed essay:
======================================================================
  (0, 6533)	1
  (0, 3306)	1
```

```
(0, 1981)     1
(0, 11036)    1
(0, 7347)     1
(0, 11029)    1
(0, 10530)    2
(0, 1734)     1
(0, 6855)     1
(0, 7374)     2
(0, 232)      1
(0, 6687)     1
(0, 3211)     1
(0, 2805)     1
(0, 10766)    1
(0, 8133)     1
(0, 8803)     1
(0, 9831)     1
(0, 1794)     1
(0, 9237)     1
(0, 10639)    3
(0, 3274)     2
(0, 7068)     1
(0, 6798)     1
(0, 9399)     1
:     :
(0, 6123)     2
(0, 5785)     2
(0, 3613)     1
(0, 7703)     2
(0, 5732)     3
(0, 8269)     2
(0, 67)       1
(0, 8670)     2
(0, 5664)     3
(0, 4383)     1
(0, 1339)     1
(0, 553)      1
(0, 1248)     1
(0, 6549)     1
(0, 5003)     1
```

```
  (0, 8116)	1
  (0, 7501)	1
  (0, 6207)	1
  (0, 5665)	2
  (0, 9968)	1
  (0, 8736)	1
  (0, 10964)	1
  (0, 5733)	1
  (0, 3449)	7
  (0, 9553)	5
```

2. Vectorizing project_title

In [0]:
```
# Initializing countvectorizer for bag of words vectorization of prepro
cessed project titles
bowTitleVectorizer = CountVectorizer(min_df = 10);
# Transforming the preprocessed project titles to bag of words vectors
bowTitleModel = bowTitleVectorizer.fit_transform(preProcessedProjectTit
lesWithoutStopWordsSub);
```

In [0]:
```
print("Some of the Features used in vectorizing preprocessed titles: "
);
equalsBorder(70);
print(bowTitleVectorizer.get_feature_names()[-40:]);
equalsBorder(70);
print("Shape of preprocessed title matrix after vectorization: ", bowTi
tleModel.shape);
equalsBorder(70);
print("Sample bag-of-words vector of preprocessed title: ");
equalsBorder(70);
print(bowTitleModel[0])
```

```
Some of the Features used in vectorizing preprocessed titles:
======================================================================
['wireless', 'wise', 'wish', 'within', 'without', 'wizards', 'wo', 'wob
ble', 'wobbles', 'wobbling', 'wobbly', 'wonder', 'wonderful', 'wonder
s', 'word', 'words', 'work', 'workers', 'working', 'works', 'workshop',
'world', 'worlds', 'worms', 'worth', 'would', 'wow', 'write', 'writer',
```

```
'writers', 'writing', 'ye', 'year', 'yearbook', 'yes', 'yoga', 'young',
'youth', 'zone', 'zoom']
======================================================================
Shape of preprocessed title matrix after vectorization:  (40000, 1774)
======================================================================
Sample bag-of-words vector of preprocessed title:
======================================================================
  (0, 766)	1
  (0, 906)	1
  (0, 514)	1
  (0, 1553)	1
  (0, 483)	1
```

Tf-Idf Vectorization

1. Vectorizing project_essay

In [0]:

```
# Intializing tfidf vectorizer for tf-idf vectorization of preprocessed
 project essays
tfIdfEssayVectorizer = TfidfVectorizer(min_df = 10);
# Transforming the preprocessed project essays to tf-idf vectors
tfIdfEssayModel = tfIdfEssayVectorizer.fit_transform(preProcessedEssays
WithoutStopWordsSub);
```

In [0]:

```
print("Some of the Features used in tf-idf vectorizing preprocessed ess
ays: ");
equalsBorder(70);
print(tfIdfEssayVectorizer.get_feature_names()[-40:]);
equalsBorder(70);
print("Shape of preprocessed title matrix after tf-idf vectorization: "
, tfIdfEssayModel.shape);
equalsBorder(70);
print("Sample Tf-Idf vector of preprocessed essay: ");
equalsBorder(70);
print(tfIdfEssayModel[0])
```

```
Some of the Features used in tf-idf vectorizing preprocessed essays:
======================================================================
['yeats', 'yell', 'yelling', 'yellow', 'yemen', 'yes', 'yesterday', 'ye
t', 'yield', 'yields', 'yoga', 'york', 'younannan', 'young', 'younger',
'youngest', 'youngsters', 'youth', 'youthful', 'youths', 'youtube', 'yu
mmy', 'zeal', 'zearn', 'zen', 'zenergy', 'zero', 'zest', 'zip', 'ziplo
c', 'zippers', 'zipping', 'zone', 'zoned', 'zones', 'zoo', 'zoom', 'zoo
ming', 'zoos', 'zumba']
======================================================================
Shape of preprocessed title matrix after tf-idf vectorization:  (40000,
11077)
======================================================================
Sample Tf-Idf vector of preprocessed essay:
======================================================================
  (0, 9553)	0.07732161197654648
  (0, 3449)	0.2978137199079083
  (0, 5733)	0.03611311825070974
  (0, 10964)	0.03819325396356506
  (0, 8736)	0.04966730436190034
  (0, 9968)	0.05933894161734909
  (0, 5665)	0.13189136979245247
  (0, 6207)	0.09909858268088724
  (0, 7501)	0.09797369103397546
  (0, 8116)	0.09716121418147701
  (0, 5003)	0.09174889764250635
  (0, 6549)	0.07739523816315956
  (0, 1248)	0.09041771504928811
  (0, 553)	0.09502243963232913
  (0, 1339)	0.07922532406820633
  (0, 4383)	0.08387324724715874
  (0, 5664)	0.12052414724469786
  (0, 8670)	0.03565737676523101
  (0, 67)	0.0797508795755641
  (0, 8269)	0.18440093271700464
  (0, 5732)	0.23244852084297085
  (0, 7703)	0.0932371184396508
  (0, 3613)	0.033250154942777416
  (0, 5785)	0.08336998078832462
  (0, 6123)	0.18451571587493337
```

```
  :      :
  (0, 9399)     0.0680639151319745
  (0, 6798)     0.08632328546640713
  (0, 7068)     0.046135007257522224
  (0, 3274)     0.10489683635458984
  (0, 10639)    0.2063461965343629
  (0, 9237)     0.1100116652395096
  (0, 1794)     0.07900547931629058
  (0, 9831)     0.03792376194008962
  (0, 8803)     0.09740047454864696
  (0, 8133)     0.09001501053091984
  (0, 10766)    0.07024528926492071
  (0, 2805)     0.05089165427462248
  (0, 3211)     0.06222851802675729
  (0, 6687)     0.022226920710368445
  (0, 232)      0.040248356980164615
  (0, 7374)     0.1846309297399045
  (0, 6855)     0.03799907965204156
  (0, 1734)     0.07743897673831124
  (0, 10530)    0.05491069896079749
  (0, 11029)    0.030886589234837624
  (0, 7347)     0.06268239285732621
  (0, 11036)    0.04610937510882687
  (0, 1981)     0.02654012905964554
  (0, 3306)     0.1031894334469226
  (0, 6533)     0.016043824658976313
```

2. Vectorizing project_title

In [0]:

```
# Intializing tfidf vectorizer for tf-idf vectorization of preprocessed
 project titles
tfIdfTitleVectorizer = TfidfVectorizer(min_df = 10);
# Transforming the preprocessed project titles to tf-idf vectors
tfIdfTitleModel = tfIdfTitleVectorizer.fit_transform(preProcessedProjec
tTitlesWithoutStopWordsSub);
```

In [0]:

```
print("Some of the Features used in tf-idf vectorizing preprocessed tit
```

```
les: ");
equalsBorder(70);
print(tfIdfTitleVectorizer.get_feature_names()[-40:]);
equalsBorder(70);
print("Shape of preprocessed title matrix after tf-idf vectorization: "
, tfIdfTitleModel.shape);
equalsBorder(70);
print("Sample Tf-Idf vector of preprocessed title: ");
equalsBorder(70);
print(tfIdfTitleModel[0])
```

```
Some of the Features used in tf-idf vectorizing preprocessed titles:
======================================================================
['wireless', 'wise', 'wish', 'within', 'without', 'wizards', 'wo', 'wob
ble', 'wobbles', 'wobbling', 'wobbly', 'wonder', 'wonderful', 'wonder
s', 'word', 'words', 'work', 'workers', 'working', 'works', 'workshop',
'world', 'worlds', 'worms', 'worth', 'would', 'wow', 'write', 'writer',
'writers', 'writing', 'ye', 'year', 'yearbook', 'yes', 'yoga', 'young',
'youth', 'zone', 'zoom']
======================================================================
Shape of preprocessed title matrix after tf-idf vectorization:  (40000,
1774)
======================================================================
Sample Tf-Idf vector of preprocessed title:
======================================================================
  (0, 483)      0.5356140846908081
  (0, 1553)     0.4441059196924978
  (0, 514)      0.4615835742389133
  (0, 906)      0.3400969810242112
  (0, 766)      0.4326223894644794
```

Average Word2Vector Vectorization

In [0]:

```
# stronging variables into pickle files python: http://www.jessicayung.
com/how-to-use-pickle-to-save-and-load-variables-in-python/
# We should have glove_vectors file for creating below model
with open('glove_vectors', 'rb') as f:
```

```
    gloveModel = pickle.load(f)
    gloveWords =  set(gloveModel.keys())
```

In [0]:
```
print("Glove vector of sample word: ");
equalsBorder(70);
print(gloveModel['technology']);
equalsBorder(70);
print("Shape of glove vector: ", gloveModel['technology'].shape);
```

```
Glove vector of sample word: 
======================================================================
[-0.26078   -0.36898   -0.022831   0.21666    0.16672   -0.20268
 -3.1219     0.33057    0.71512    0.28874    0.074368  -0.033203
  0.23783    0.21052    0.076562   0.13007   -0.31706   -0.45888
 -0.45463   -0.13191    0.49761    0.072704   0.16811    0.18846
 -0.16688   -0.21973    0.08575   -0.19577   -0.2101    -0.32436
 -0.56336    0.077996  -0.22758   -0.66569    0.14824    0.038945
  0.50881   -0.1352     0.49966   -0.4401    -0.022335  -0.22744
  0.22086    0.21865    0.36647    0.30495   -0.16565    0.038759
  0.28108   -0.2167     0.12453    0.65401    0.34584   -0.2557
 -0.046363  -0.31111   -0.020936  -0.17122   -0.77114    0.29289
 -0.14625    0.39541   -0.078938   0.051127   0.15076    0.085126
  0.183     -0.06755    0.26312    0.0087276  0.0066415  0.37033
  0.03496   -0.12627   -0.052626  -0.34897    0.14672    0.14799
 -0.21821   -0.042785   0.2661    -1.1105     0.31789    0.27278
  0.054468  -0.27458    0.42732   -0.44101   -0.19302   -0.32948
  0.61501   -0.22301   -0.36354   -0.34983   -0.16125   -0.17195
 -3.363      0.45146   -0.13753    0.31107    0.2061     0.33063
  0.45879    0.24256    0.042342   0.074837  -0.12869    0.12066
  0.42843   -0.4704    -0.18937    0.32685    0.26079    0.20518
 -0.18432   -0.47658    0.69193    0.18731   -0.12516    0.35447
 -0.1969    -0.58981   -0.88914    0.5176     0.13177   -0.078557
  0.032963  -0.19411    0.15109    0.10547   -0.1113    -0.61533
  0.0948    -0.3393    -0.20071   -0.30197    0.29531    0.28017
  0.16049    0.25294   -0.44266   -0.39412    0.13486    0.25178
 -0.044114   1.1519     0.32234   -0.34323   -0.10713   -0.15616
  0.031206   0.46636   -0.52761   -0.39296   -0.068424  -0.04072
  0.41508   -0.34564    0.71001   -0.364      0.2996     0.032281

  0.34035    0.23452    0.78342    0.48045   -0.1609     0.40102
```

```
-0.071795  -0.16531    0.082153   0.52065    0.24194    0.17113
 0.33552   -0.15725   -0.38984    0.59337   -0.19388   -0.39864
-0.47901    1.0835     0.24473    0.41309    0.64952    0.46846
 0.024386  -0.72087   -0.095061   0.10095   -0.025229   0.29435
-0.57696    0.53166   -0.0058338 -0.3304     0.19661   -0.085206
 0.34225    0.56262    0.19924   -0.027111  -0.44567    0.17266
 0.20887   -0.40702    0.63954    0.50708   -0.31862   -0.39602
-0.1714    -0.040006  -0.45077   -0.32482   -0.0316     0.54908
-0.1121     0.12951   -0.33577   -0.52768   -0.44592   -0.45388
 0.66145    0.33023   -1.9089     0.5318     0.21626   -0.13152
 0.48258    0.68028   -0.84115   -0.51165    0.40017    0.17233
-0.033749   0.045275   0.37398   -0.18252    0.19877    0.1511
 0.029803   0.16657   -0.12987   -0.50489    0.55311   -0.22504
 0.13085   -0.78459    0.36481   -0.27472    0.031805   0.53052
-0.20078    0.46392   -0.63554    0.040289  -0.19142   -0.0097011
 0.068084  -0.10602    0.25567    0.096125  -0.10046    0.15016
-0.26733   -0.26494    0.057888   0.062678  -0.11596    0.28115
 0.25375   -0.17954    0.20615    0.24189    0.062696   0.27719
-0.42601   -0.28619   -0.44697   -0.082253  -0.73415   -0.20675
-0.60289   -0.06728    0.15666   -0.042614   0.41368   -0.17367
-0.54012    0.23883    0.23075    0.13608   -0.058634  -0.089705
 0.18469    0.023634   0.16178    0.23384    0.24267    0.091846 ]
======================================================================
Shape of glove vector:  (300,)
```

In [0]:

```
def getWord2VecVectors(texts):
    word2VecTextsVectors = [];
    for preProcessedText in tqdm(texts):
        word2VecTextVector = np.zeros(300);
        numberOfWordsInText = 0;
        for word in preProcessedText.split():
            if word in gloveWords:
                word2VecTextVector += gloveModel[word];
                numberOfWordsInText += 1;
        if numberOfWordsInText != 0:
            word2VecTextVector = word2VecTextVector / numberOfWordsInTe
xt;
        word2VecTextsVectors.append(word2VecTextVector);
    return word2VecTextsVectors;
```

1. Vectorizing project_essay

In [0]:
```
word2VecEssaysVectors = getWord2VecVectors(preProcessedEssaysWithoutSto
pWords);
```

In [0]:
```
print("Shape of Word2Vec vectorization matrix of essays: {},{}".format(
len(word2VecEssaysVectors), len(word2VecEssaysVectors[0])));
equalsBorder(70);
print("Sample essay: ");
equalsBorder(70);
print(preProcessedEssaysWithoutStopWords[0]);
equalsBorder(70);
print("Word2Vec vector of sample essay: ");
equalsBorder(70);
print(word2VecEssaysVectors[0]);
```

```
Shape of Word2Vec vectorization matrix of essays: 109248,300
======================================================================
Sample essay: 
======================================================================
students english learners working english second third languages meltin
g pot refugees immigrants native born americans bringing gift language 
school 24 languages represented english learner program students every 
level mastery also 40 countries represented families within school stud
ent brings wealth knowledge experiences us open eyes new cultures belie
fs respect limits language limits world ludwig wittgenstein english lea
rner strong support system home begs resources many times parents learn
ing read speak english along side children sometimes creates barriers p
arents able help child learn phonetics letter recognition reading skill
s providing dvd players students able continue mastery english language
even no one home able assist families students within level 1 proficien
cy status offered part program educational videos specially chosen engl
ish learner teacher sent home regularly watch videos help child develop
early reading skills parents not access dvd player opportunity check dv
d player use year plan use videos educational dvd years come el student
s nannan
======================================================================
```

```
Word2Vec vector of sample essay:
======================================================================
[-1.40030644e-02  8.78995685e-02  3.50108161e-02 -5.90358980e-03
 -5.93166809e-02 -6.21039893e-02 -2.96711248e+00  9.45840302e-02
 -8.18737785e-03  4.46964161e-02 -7.64722101e-02  6.97099444e-02
  8.44441262e-02 -1.22974138e-01 -3.55310208e-02 -8.90947154e-02
  1.20959579e-01 -1.21977699e-01  4.61334597e-02 -3.33640832e-02
  1.24900557e-01  7.18837631e-02 -6.14885114e-02 -2.67269047e-02
  6.82086621e-02 -3.60263034e-02  1.17172255e-01 -1.17868631e-01
 -1.13467710e-01 -9.25920168e-02 -2.42461725e-01 -7.92963658e-02
  3.52513154e-03  1.79752468e-01 -4.69217812e-02 -3.56593007e-02
 -7.95331477e-03 -6.71107383e-04 -1.80828067e-02 -1.16224805e-02
 -3.69645852e-02  1.61287176e-01 -1.75201329e-01 -6.02256376e-02
  1.48811886e-02 -9.00106181e-02  7.72160490e-02  7.42989819e-02
 -1.02682389e-02 -1.33311658e-01 -2.82030537e-02 -7.71051879e-03
  7.33988450e-02  3.54095087e-02 -5.80719597e-03 -8.70242758e-02
 -3.57117638e-02  2.78475651e-02 -1.54957291e-01 -3.24157495e-02
 -5.93266570e-02 -8.80254174e-02  2.18914318e-01 -1.22730395e-02
 -1.05831485e-01  1.53985730e-01  7.15618933e-02 -3.97147470e-02
  1.47169116e-01 -4.50476644e-03 -1.49678829e-01  5.52201396e-02
  3.04915879e-02 -6.24086617e-02 -7.68483134e-02 -7.50149195e-02
 -1.07105068e-01 -2.69954530e-02  1.28067340e-01 -3.42946330e-02
  4.24139667e-02 -4.49685043e-01  1.52793905e-01 -9.06178181e-02
 -6.67951510e-02 -2.72063766e-02  7.37261792e-02 -8.64977130e-02
  1.64616877e-01  4.86745523e-02 -4.44542828e-02 -3.04823530e-02
  2.63897436e-02 -6.59345034e-02 -5.21813664e-02 -7.45015886e-02
 -2.21975948e+00  8.57858456e-02  7.73778584e-02  1.14644799e-01
 -1.50536483e-01 -5.17326940e-02  3.23826117e-02 -1.15700542e-01
  7.15651973e-02  9.15412617e-02  5.41334631e-02 -1.25451318e-01
  2.80941483e-02 -3.95890262e-02 -1.67010497e-02  1.74708879e-02
  4.58374505e-02  2.56664910e-01  3.74891134e-02  3.00990497e-02
 -2.18904765e-01  9.37672966e-02  9.99403436e-02  5.26255996e-02
 -6.67958718e-02  5.97650946e-02  4.14311192e-02 -6.85917603e-02
  1.72453235e-02  1.02485026e-01  3.02940430e-02  9.59998859e-03
  1.96364913e-02  1.22438477e-01  7.98410557e-02  1.92611322e-02
  6.44085906e-03  4.94252148e-03 -5.36137718e-03 -1.17976934e-01
  1.77991634e-01 -2.51954819e-02  8.02478188e-02  2.29125079e-01
  3.79080403e-02  1.22892819e-02  7.19621470e-02 -9.25031570e-02
 -8.86571674e-02 -4.74898563e-02  1.68688409e-02 -1.15134901e-01
```

```
  1.76528904e-01 -6.30485141e-02 -4.99678329e-02 -1.00350507e-01
  1.25089302e-02 -4.08706114e-02  4.50565289e-02  2.49286074e-02
 -1.29713758e-03 -3.21404376e-02 -2.52972249e-02 -9.63531510e-02
  8.42448993e-04 -7.29482953e-03 -3.77497893e-02 -9.35034987e-02
 -3.45719793e-02  7.15921796e-02 -1.29330935e-01  1.28508101e-02
  4.24846988e-02 -8.43078228e-02  4.79772134e-02 -3.05753799e-02
 -3.03772013e-02 -2.10572558e-01 -1.05464289e-03  5.18230436e-02
 -4.39921874e-02  5.29591584e-02 -1.08551689e-01  2.88053128e-02
 -4.88957058e-02  2.31962381e-01 -2.90986193e-02 -2.83725755e-02
 -6.80350899e-02 -6.99966387e-02 -6.80414679e-02 -7.63552362e-02
 -1.59287859e-02 -2.59947651e-03 -7.81848121e-03 -1.14299579e-01
 -2.02054698e-02  1.21184430e-03  2.59984919e-02 -7.64172013e-02
  9.47882617e-03 -5.71751181e-02  1.25667972e-01 -4.60388139e-02
  5.51296403e-02 -6.73280980e-02 -2.06862389e-02  1.12049165e-01
 -7.63451436e-02  4.71124027e-02  6.32404235e-02 -2.13828034e-02
  1.24239236e-01  5.08985235e-02  2.05136711e-03  1.45916498e-02
  4.25123886e-02 -9.41766832e-02 -3.08569389e-02 -2.57995470e-02
 -3.53808765e-02 -7.16000389e-02  1.35426121e-02  4.57596799e-02
 -1.85721693e-01 -6.62042523e-02 -1.45448285e-01  5.50366758e-02
 -2.09367026e+00  1.23479489e-01 -1.46630889e-01 -8.86940765e-02
 -7.32806463e-02 -1.48629733e-01  3.23867248e-03  7.08553181e-02
  1.10315906e-02 -2.35431879e-02 -7.69633283e-02 -1.13640894e-01
  9.96301846e-02 -5.70585054e-02 -5.45997987e-04  9.42995174e-02
 -1.40422433e-01 -5.03571812e-04 -2.50305216e-01  3.79384141e-02
 -6.44086637e-02 -1.53146188e-02 -2.55858274e-02 -1.10195376e-01
  1.62183899e-02 -1.61929591e-02  2.03421993e-02  1.21424534e-01
  5.02740463e-02  2.37900799e-02  9.07398322e-02  1.57962685e-02
  3.73036075e-02 -8.14876248e-02  1.37349395e-01 -8.17880913e-02
  9.27907812e-02  6.76093826e-03 -5.22928389e-02  6.02994188e-02
  8.28096711e-03 -1.05344042e-01 -1.02705751e-01  2.45275938e-02
 -1.18970611e-02  9.86759282e-02 -1.92870134e-02  9.71936577e-03
 -1.40249490e-01  1.61314103e-01 -4.55344879e-02  2.21929812e-02
  9.54108215e-02 -1.25028370e-02  2.89625007e-02  1.65818081e-02
 -2.34467852e-02 -7.88610081e-02  3.34242148e-03  4.43269879e-02
 -4.08419376e-02  6.06990416e-02  2.33916564e-02 -1.02773899e-02
  9.21596550e-02  9.90483805e-02  7.50525638e-03 -4.07725570e-03
 -6.93980047e-02 -3.50341946e-02 -8.79849597e-02 -4.10474223e-02
  4.55004698e-03  2.27073689e-01  1.37340472e-01  4.43856114e-02]
```

2. Vectorizing project_title

In [0]:
```
word2VecTitlesVectors = getWord2VecVectors(preProcessedProjectTitlesWit
houtStopWords);
```

In [0]:
```
print("Shape of Word2Vec vectorization matrix of project titles: {}, {}
".format(len(word2VecTitlesVectors), len(word2VecTitlesVectors[0])));
equalsBorder(70);
print("Sample title: ");
equalsBorder(70);
print(preProcessedProjectTitlesWithoutStopWords[0]);
equalsBorder(70);
print("Word2Vec vector of sample title: ");
equalsBorder(70);
print(word2VecTitlesVectors[0]);
```

```
Shape of Word2Vec vectorization matrix of project titles: 109248, 300
======================================================================
Sample title:
======================================================================
educational support english learners home
======================================================================
Word2Vec vector of sample title:
======================================================================
[-4.1285000e-02  4.4970000e-02  1.4283080e-01  1.9901860e-02
 -8.4519200e-02 -4.3207400e-01 -2.8496800e+00 -2.2953320e-01
  2.1736960e-01  3.4239600e-01 -7.5568200e-02  1.8077600e-01
  1.3998316e-01 -1.6401800e-01 -2.9812820e-01 -2.5030200e-01
  2.0420960e-01 -1.6882720e-01  6.5439800e-02 -1.6061000e-01
  2.2179020e-01  2.9944900e-01  2.7358000e-02 -8.8528800e-02
  1.5856400e-01  6.2905000e-02  2.0427440e-01 -1.9312560e-01
 -9.2904600e-02 -2.2050020e-01 -5.7761060e-01 -1.2101294e-01
  1.6846980e-01  2.8212460e-01 -1.8210120e-01  1.7754000e-02
  1.4805200e-01  4.1059000e-02  3.1145000e-02 -9.5658000e-02
 -9.6840000e-03  2.4896520e-01 -2.5047440e-01  7.7859000e-02
 -3.7512000e-03 -2.7071920e-01  2.5586200e-02  2.3205600e-01
  1.0154800e-01 -5.2259200e-01 -1.3211440e-01  1.1908300e-01
  2.7147196e-01  5.6135400e-02 -5.3140200e-02 -1.4937160e-01
```

```
-1.0488160e-01  1.2059600e-01 -1.2639620e-01 -1.4316640e-01
-2.2147600e-01 -1.9137800e-01  1.6595340e-01 -5.6078000e-02
 3.9884400e-02  1.0854760e-01  1.5552920e-01  7.8204600e-02
 9.5928000e-02 -6.2156000e-03 -1.1407312e-01  3.6862800e-02
-8.7530020e-02 -4.7668000e-02 -2.3264200e-01 -6.1687200e-02
-3.1690916e-01 -1.1851380e-01  1.4931240e-01 -7.7857200e-02
 1.8634840e-01 -4.6202100e-01  2.7096800e-01 -3.0512800e-02
-2.1226400e-01 -1.5356200e-02  1.0844260e-01 -8.2669200e-02
 2.8918600e-01  1.3372960e-01 -8.3522800e-02  4.6474200e-02
 2.0703580e-01 -2.1937640e-01 -1.0252400e-01 -2.5177000e-01
-2.8408000e+00  1.6622880e-01  1.1216234e-01  2.0837920e-01
-1.5711600e-01 -1.9159400e-01 -1.4992160e-01 -2.7392820e-01
 3.4989140e-01  1.3991600e-01  1.6275200e-01  1.3887200e-01
 1.8212760e-01 -3.2218600e-02  4.3172000e-02  1.8323640e-01
 1.2295780e-01  4.4706600e-01  2.1688400e-02 -3.8988200e-02
-3.2467400e-01  3.8389160e-01 -1.4416560e-01  1.1117380e-01
-1.6218300e-01  1.3871928e-01  1.4305240e-01 -7.6173200e-02
 8.9476800e-02  2.6043820e-01  5.1114000e-02  1.0619800e-01
 1.5968840e-01  1.0530680e-01  8.6300000e-02  1.4667260e-01
 1.2320460e-02 -6.6124620e-02 -1.1017760e-01 -1.5091940e-01
 2.1297280e-01 -3.2808520e-01  1.4493194e-01  2.1848680e-01
-4.1809800e-03  8.5340000e-02 -1.2410789e-01 -2.2308140e-01
 8.8026000e-02  1.9555000e-01 -3.7981400e-02 -1.7720080e-01
 3.4328600e-01 -3.7459600e-01 -1.7268200e-01 -2.1554400e-01
-1.1533400e-01  9.9680000e-02 -1.9032980e-01  8.6249800e-02
 7.6682200e-02 -9.1090380e-02 -9.3714000e-02 -1.7333260e-01
 8.6429960e-02 -6.7933600e-02 -8.6470600e-02 -2.2431600e-01
-2.8319800e-01  1.0138200e-01 -2.8114320e-01 -1.1168240e-01
 2.1770560e-02 -1.3971160e-01  2.1795080e-01 -1.1995600e-01
-1.3166600e-02 -3.4848260e-01 -3.0102000e-02  2.3396200e-02
 2.8840000e-02  2.8763000e-01 -2.3679600e-02  1.1806440e-01
-3.2261460e-01  2.2622920e-01  1.9506400e-02  1.4363200e-01
-1.3668380e-01 -1.0521880e-01 -3.9385400e-03 -4.6388000e-02
-7.7493780e-02 -2.4700800e-02 -5.2006200e-02 -2.6299360e-01
-2.5607520e-01  2.1704520e-01  5.6336000e-02 -6.3474400e-02
-1.0400400e-01 -1.7901000e-01  2.0326180e-01 -2.8708740e-01
 1.0132000e-01 -1.6278080e-01  1.2441440e-01  3.2699820e-01
-4.8321600e-02 -3.6052800e-02  2.2539620e-01 -8.2764000e-03
 3.1087258e-01  2.4090500e-01 -9.9590000e-02  1.2362460e-01
```

```
  1.7440000e-03 -1.6117280e-01  7.4570000e-02  3.1281120e-02
 -1.1758000e-02 -1.8464800e-02 -2.0872020e-01 -3.9510000e-03
 -5.7714400e-01 -1.8090080e-01 -2.8288200e-01 -2.4662120e-01
 -1.8806540e+00  4.4765400e-01 -2.9412700e-01 -1.7280000e-02
 -3.1931600e-01 -1.9190500e-01 -1.1642000e-02  1.7475600e-01
  1.3068840e-01  1.1943000e-01 -1.7219524e-01  1.9224000e-02
  2.2620000e-01 -1.0821980e-01  1.3789060e-01  2.6989320e-01
 -2.4364960e-01 -1.3650800e-01 -3.0984180e-01 -3.9546200e-02
 -1.1410800e-01 -6.6744640e-02  1.6330620e-01 -4.0601000e-01
  9.3793000e-02 -8.3026800e-02  9.0567600e-02  3.1595600e-01
  1.6786620e-01  1.0099860e-01  3.5043600e-02  6.6221200e-02
 -3.5907800e-02 -2.4589760e-01  2.6006800e-01 -8.0637000e-02
  1.5359624e-01 -1.1078680e-01 -5.6956400e-02  2.2253080e-01
  3.5808000e-02 -1.8873860e-01 -2.5032660e-01  3.6167400e-02
 -2.2424700e-01  2.7863640e-01  2.2622600e-02  1.3753300e-01
 -2.3369620e-01  2.8058040e-01  5.0818000e-02 -3.4805800e-02
  1.7916600e-01 -7.5374000e-02  7.1228900e-02  1.7556000e-01
 -5.8004120e-01 -2.0522500e-01 -1.3367960e-01  1.3656000e-02
 -2.9052200e-02  1.3698600e-02  1.1746340e-01 -2.3288400e-02
  2.7706200e-01  1.6106000e-01 -2.0183340e-01  5.7781800e-02
 -2.0954400e-01 -1.4111260e-02 -3.1186860e-01 -2.9536360e-02
 -1.7226500e-01  3.5709400e-01  2.9448200e-01  8.5600000e-05]
```

Tf-Idf Weighted Word2Vec Vectorization

1. Vectorizing project_essay

In [0]:

```
# Initializing tfidf vectorizer
tfIdfEssayTempVectorizer = TfidfVectorizer();
# Vectorizing preprocessed essays using tfidf vectorizer initialized ab
ove
tfIdfEssayTempVectorizer.fit(preProcessedEssaysWithoutStopWords);
# Saving dictionary in which each word is key and it's idf is value
tfIdfEssayDictionary = dict(zip(tfIdfEssayTempVectorizer.get_feature_na
mes(), list(tfIdfEssayTempVectorizer.idf_)));
```

```
# Creating set of all unique words used by tfidf vectorizer
tfIdfEssayWords = set(tfIdfEssayTempVectorizer.get_feature_names());
```

In [0]:
```
# Creating list to save tf-idf weighted vectors of essays
tfIdfWeightedWord2VecEssaysVectors = [];
# Iterating over each essay
for essay in tqdm(preProcessedEssaysWithoutStopWords):
    # Sum of tf-idf values of all words in a particular essay
    cumulativeSumTfIdfWeightOfEssay = 0;
    # Tf-Idf weighted word2vec vector of a particular essay
    tfIdfWeightedWord2VecEssayVector = np.zeros(300);
    # Splitting essay into list of words
    splittedEssay = essay.split();
    # Iterating over each word
    for word in splittedEssay:
        # Checking if word is in glove words and set of words used by t
fIdf essay vectorizer
        if (word in gloveWords) and (word in tfIdfEssayWords):
            # Tf-Idf value of particular word in essay
            tfIdfValueWord = tfIdfEssayDictionary[word] * (essay.count(
word) / len(splittedEssay));
            # Making tf-idf weighted word2vec
            tfIdfWeightedWord2VecEssayVector += tfIdfValueWord * gloveM
odel[word];
            # Summing tf-idf weight of word to cumulative sum
            cumulativeSumTfIdfWeightOfEssay += tfIdfValueWord;
    if cumulativeSumTfIdfWeightOfEssay != 0:
        # Taking average of sum of vectors with tf-idf cumulative sum
        tfIdfWeightedWord2VecEssayVector = tfIdfWeightedWord2VecEssayVe
ctor / cumulativeSumTfIdfWeightOfEssay;
    # Appending the above calculated tf-idf weighted vector of particul
ar essay to list of vectors of essays
    tfIdfWeightedWord2VecEssaysVectors.append(tfIdfWeightedWord2VecEssa
yVector);
```

In [0]:
```
print("Shape of Tf-Idf weighted Word2Vec vectorization matrix of projec
t essays: {}, {}".format(len(tfIdfWeightedWord2VecEssaysVectors), len(t
```

```
fIdfWeightedWord2VecEssaysVectors[0])));
equalsBorder(70);
print("Sample Essay: ");
equalsBorder(70);
print(preProcessedEssaysWithoutStopWords[0]);
equalsBorder(70);
print("Tf-Idf Weighted Word2Vec vector of sample essay: ");
equalsBorder(70);
print(tfIdfWeightedWord2VecEssaysVectors[0]);
```

```
Shape of Tf-Idf weighted Word2Vec vectorization matrix of project essay
s: 109248, 300
======================================================================
Sample Essay:
======================================================================
students english learners working english second third languages meltin
g pot refugees immigrants native born americans bringing gift language
school 24 languages represented english learner program students every
level mastery also 40 countries represented families within school stud
ent brings wealth knowledge experiences us open eyes new cultures belie
fs respect limits language limits world ludwig wittgenstein english lea
rner strong support system home begs resources many times parents learn
ing read speak english along side children sometimes creates barriers p
arents able help child learn phonetics letter recognition reading skill
s providing dvd players students able continue mastery english language
even no one home able assist families students within level 1 proficien
cy status offered part program educational videos specially chosen engl
ish learner teacher sent home regularly watch videos help child develop
early reading skills parents not access dvd player opportunity check dv
d player use year plan use videos educational dvd years come el student
s nannan
======================================================================
Tf-Idf Weighted Word2Vec vector of sample essay:
======================================================================
[-5.37582850e-02  7.68689598e-02  7.85741822e-02  4.38958976e-02
 -8.56874440e-02 -1.20832331e-01 -2.68120986e+00  7.17018732e-02
  1.03799206e-04 -5.17255299e-03 -2.67529751e-02  7.40185988e-02
  1.36881934e-01 -8.62706493e-02 -6.35020145e-02 -8.44084597e-02
  1.27523921e-01 -1.77105602e-01  3.68451284e-02 -5.74471880e-02
  1.86477259e-01  9.28786009e-02 -9.73137896e-02 -1.15230456e-02
```

```
  4.41962185e-02 -9.32894883e-02  1.11912943e-01 -1.17540961e-01
 -1.22150893e-01 -9.14028838e-02 -1.73918944e-01 -4.54143189e-02
 -7.82036060e-02  3.05617633e-01 -8.71850266e-02  6.31466708e-03
  1.15683161e-01  1.71477594e-02 -5.52983597e-02  9.08989585e-02
 -3.89808292e-04  1.97696142e-01 -4.08078376e-01 -5.39990199e-02
 -1.20129600e-02 -1.12456389e-01  2.92046345e-02  1.37924729e-01
  2.83465620e-02 -2.26817169e-01 -2.29639267e-02  6.94257143e-03
  5.80535394e-02  2.86454339e-02 -7.51508216e-02 -6.21569354e-02
 -1.41805544e-01  2.78707358e-02 -1.63165999e-01 -1.29716251e-01
 -5.67625355e-02 -8.59507500e-02  3.54019902e-01 -4.96274469e-02
 -6.88414062e-02  1.58623510e-01  1.24798600e-01  4.29711440e-02
  7.82814323e-02 -1.73260116e-02 -1.23679491e-01  1.47617250e-01
  4.27083617e-02 -1.16531047e-01 -1.27122530e-01 -5.93638332e-03
 -1.99224414e-01 -8.66160391e-02  2.47701354e-01  1.61218205e-02
  3.56880345e-02 -3.71320273e-01  2.65501745e-01 -4.56454865e-02
 -7.85433814e-02 -5.99177835e-02  4.42212779e-02 -8.20739267e-02
  2.14031939e-01  2.42131497e-02 -1.34069697e-01  7.15871686e-03
  4.00667270e-02 -6.75881497e-02 -7.07967357e-02 -2.15984749e-02
 -2.09734597e+00  1.02300477e-01  6.61169899e-02  5.70146517e-02
 -1.91302495e-01 -1.38114014e-01 -1.10709961e-01 -1.66994098e-01
  9.17800823e-02  1.35327093e-01  2.20333244e-02 -3.83844831e-02
  2.57206511e-02 -5.54503565e-02 -3.41973653e-03  1.99777588e-02
  4.85050396e-02  2.13190534e-01  4.64281665e-02  6.51171751e-02
 -5.80015838e-02  1.19900386e-01  1.18803830e-01  7.05550873e-02
 -1.87330886e-01  1.41219129e-01  1.33569574e-01  1.00530000e-01
  4.14498415e-02  1.39860952e-01 -7.95709830e-02  9.70242332e-02
  1.07442882e-01  9.00794808e-02  7.47745032e-02  4.18772282e-02
 -7.10347826e-03 -7.62379756e-03 -7.31715828e-02 -1.16370646e-01
  2.82271708e-01 -5.30885621e-02  4.51472249e-02  2.61376253e-01
  1.29080066e-02  3.96843846e-02  1.04430681e-01 -1.30495811e-01
 -1.17999239e-01 -1.02810089e-01 -6.52713784e-02 -1.81350799e-01
  1.55415740e-01 -4.43517889e-02 -8.34350788e-02 -1.31445407e-01
 -8.87524029e-02 -1.15321245e-02  8.67587067e-03  3.55646447e-02
 -4.32365925e-02  2.44285859e-03  2.73165854e-02 -1.91651165e-01
  6.70942750e-03  1.45533103e-02 -5.95191056e-02 -9.78336553e-02
 -4.61200683e-02  1.04017495e-02 -1.68129330e-01 -5.53455289e-02
 -1.95353920e-02 -3.24088827e-03  9.94121739e-02 -2.20584067e-02
  1.36190091e-02 -3.13014669e-01  4.46748268e-02  6.11251996e-02
 -5.59088914e-02  8.07071841e-02 -7.80920682e-02  1.05535003e-02
```

```
 -8.49705076e-02  1.87800458e-01 -5.53305425e-02 -4.05296946e-02
 -1.68105655e-02 -9.64697267e-02 -1.00114054e-01 -1.25303984e-01
 -6.77861115e-02  1.38106300e-02  4.97948787e-02 -1.04414463e-01
  3.12147536e-03 -2.46650333e-02  1.56250756e-02 -3.41987984e-02
  2.90197738e-02 -1.30795750e-01  1.71425098e-01 -1.33199913e-01
 -4.35452619e-02 -1.52841321e-01  3.37717104e-02  2.11400042e-01
 -1.08493100e-01  6.64905827e-02  4.45687503e-02 -3.38898797e-03
  1.47302984e-01  3.10931848e-02  6.94873935e-03 -3.79090162e-02
  3.97055902e-02 -3.12563998e-02  2.99815273e-02 -9.30892230e-03
 -3.37192802e-02 -7.79667288e-02  4.20509297e-02  4.33535394e-02
 -2.38238094e-01 -4.11188300e-02 -1.93930088e-01  1.15012485e-01
 -2.14605373e+00  1.36975648e-01 -1.79026305e-01 -1.42630498e-01
 -1.37558424e-01 -1.55433436e-01 -6.96701214e-02  1.05328488e-01
  3.43486342e-02 -2.37676310e-03 -6.80980842e-02 -1.92470331e-01
  1.54727348e-01 -7.47455695e-02 -1.58054203e-02  3.33369549e-02
 -1.70510752e-01 -5.74331307e-02 -2.38994456e-01  5.64188931e-02
 -8.55051184e-02 -5.52984572e-02 -5.00408589e-02 -6.81572658e-02
  5.15848477e-03 -3.58487773e-02  7.00056842e-02  1.33127170e-01
  5.57938159e-02  1.03106840e-01  4.18598320e-02 -2.78162076e-03
  8.83131944e-02 -1.31482831e-01  1.34875022e-01 -8.31772344e-02
  1.62319378e-01  9.25839856e-02 -7.07548194e-02  1.74355644e-01
  1.53106818e-02 -1.74504449e-01 -5.39158255e-02 -1.16968555e-02
 -1.37824311e-01  1.07713713e-01  4.48548015e-02  1.07272158e-01
 -1.59084558e-01  1.94342786e-01 -4.73514319e-02 -4.87250503e-02
  2.82023483e-02 -4.18474756e-02  8.04397595e-02 -3.34005484e-02
 -1.00808502e-01 -1.15380334e-01  7.05894205e-02  2.92052920e-02
 -5.72604859e-02 -7.39274088e-03  1.44106517e-02 -2.64282237e-02
  2.31512689e-01  1.50161666e-01 -5.21462274e-02 -1.00796916e-02
 -4.47392305e-02  4.83958092e-02 -2.21927272e-01 -9.69846899e-02
 -5.91211767e-03  2.52508756e-01  1.08677704e-01  5.05047869e-02]
```

2. Vectorizing project_title

In [0]:

```
# Initializing tfidf vectorizer
tfIdfTitleTempVectorizer = TfidfVectorizer();
# Vectorizing preprocessed titles using tfidf vectorizer initialized ab
ove
```

```
tfIdfTitleTempVectorizer.fit(preProcessedProjectTitlesWithoutStopWords
);
# Saving dictionary in which each word is key and it's idf is value
tfIdfTitleDictionary = dict(zip(tfIdfTitleTempVectorizer.get_feature_na
mes(), list(tfIdfTitleTempVectorizer.idf_)));
# Creating set of all unique words used by tfidf vectorizer
tfIdfTitleWords = set(tfIdfTitleTempVectorizer.get_feature_names());
```

In [0]:

```
# Creating list to save tf-idf weighted vectors of project titles
tfIdfWeightedWord2VecTitlesVectors = [];
# Iterating over each title
for title in tqdm(preProcessedProjectTitlesWithoutStopWords):
    # Sum of tf-idf values of all words in a particular project title
    cumulativeSumTfIdfWeightOfTitle = 0;
    # Tf-Idf weighted word2vec vector of a particular project title
    tfIdfWeightedWord2VecTitleVector = np.zeros(300);
    # Splitting title into list of words
    splittedTitle = title.split();
    # Iterating over each word
    for word in splittedTitle:
        # Checking if word is in glove words and set of words used by t
fIdf title vectorizer
        if (word in gloveWords) and (word in tfIdfTitleWords):
            # Tf-Idf value of particular word in title
            tfIdfValueWord = tfIdfTitleDictionary[word] * (title.count(
word) / len(splittedTitle));
            # Making tf-idf weighted word2vec
            tfIdfWeightedWord2VecTitleVector += tfIdfValueWord * gloveM
odel[word];
            # Summing tf-idf weight of word to cumulative sum
            cumulativeSumTfIdfWeightOfTitle += tfIdfValueWord;
    if cumulativeSumTfIdfWeightOfTitle != 0:
        # Taking average of sum of vectors with tf-idf cumulative sum
        tfIdfWeightedWord2VecTitleVector = tfIdfWeightedWord2VecTitleVe
ctor / cumulativeSumTfIdfWeightOfTitle;
    # Appending the above calculated tf-idf weighted vector of particul
ar title to list of vectors of project titles
    tfIdfWeightedWord2VecTitlesVectors.append(tfIdfWeightedWord2VecTitl
eVector);
```

```python
In [0]: print("Shape of Tf-Idf weighted Word2Vec vectorization matrix of project titles: {}, {}".format(len(tfIdfWeightedWord2VecTitlesVectors), len(tfIdfWeightedWord2VecTitlesVectors[0])));
equalsBorder(70);
print("Sample Title: ");
equalsBorder(70);
print(preProcessedProjectTitlesWithoutStopWords[0]);
equalsBorder(70);
print("Tf-Idf Weighted Word2Vec vector of sample title: ");
equalsBorder(70);
print(tfIdfWeightedWord2VecTitlesVectors[0]);
```

```
Shape of Tf-Idf weighted Word2Vec vectorization matrix of project titles: 109248, 300
======================================================================
Sample Title:
======================================================================
educational support english learners home
======================================================================
Tf-Idf Weighted Word2Vec vector of sample title:
======================================================================
[-3.23904891e-02  5.58064810e-02  1.32666911e-01  3.84227573e-02
 -6.71984492e-02 -4.30940397e-01 -2.84607947e+00 -2.45905055e-01
  1.96794858e-01  3.19604663e-01 -6.12568872e-02  1.59218099e-01
  1.25129027e-01 -1.67580327e-01 -2.82644062e-01 -2.47555536e-01
  2.18304104e-01 -1.57431101e-01  7.66481545e-02 -1.61436633e-01
  2.38451267e-01  2.86712258e-01  2.70730890e-02 -9.74962294e-02
  1.67511144e-01  7.18131102e-02  1.82846112e-01 -1.96778087e-01
 -8.19948978e-02 -2.25877630e-01 -5.54573752e-01 -1.28462870e-01
  1.61012606e-01  2.94412658e-01 -1.63196910e-01 -1.23217523e-02
  1.37466355e-01  4.45437696e-02  4.65691769e-02 -1.17867965e-01
 -2.41502151e-03  2.24350668e-01 -2.51274676e-01  8.29431360e-02
 -1.65996673e-02 -2.47747576e-01  1.45110611e-03  2.37117949e-01
  9.71345150e-02 -5.13516477e-01 -1.40296688e-01  1.42775548e-01
  2.89949805e-01  6.49771690e-02 -3.41581088e-02 -1.58076306e-01
 -1.07731741e-01  7.59015357e-02 -1.21511682e-01 -1.16519972e-01
 -2.27321940e-01 -1.63525257e-01  1.80860125e-01 -4.17314689e-02
  4.60171896e-02  1.00024674e-01  1.54588362e-01  8.25394911e-02
  7.45768118e-02 -1.80240543e-02 -1.22956246e-01 -4.97450371e-03
```

```
 7.45768118e-02 -1.80240545e-02 -1.22950246e-01 -4.97450571e-03
-8.06577406e-02 -5.00614538e-02 -2.15836210e-01 -5.89271531e-02
-3.26363335e-01 -1.32706775e-01  1.61236199e-01 -1.25038790e-01
 1.96493846e-01 -4.95095193e-01  2.34765396e-01 -4.44646606e-02
-2.04266125e-01 -3.21415735e-02  8.48111983e-02 -7.27603472e-02
 2.79183660e-01  1.18968262e-01 -7.43300594e-02  6.34587771e-02
 1.99863053e-01 -2.13382053e-01 -1.01221319e-01 -2.49884070e-01
-2.92249478e+00  1.60273141e-01  7.74579728e-02  1.85323805e-01
-1.33255909e-01 -2.00013519e-01 -1.31974722e-01 -2.62288530e-01
 3.54852941e-01  1.18537924e-01  1.62207829e-01  1.24436802e-01
 1.98867481e-01 -4.87526944e-03  3.00886908e-02  2.09330567e-01
 1.17189984e-01  3.94887340e-01  2.52941492e-02 -5.13348554e-02
-2.91140828e-01  4.06939567e-01 -1.70319175e-01  1.17651155e-01
-1.66813086e-01  1.53049826e-01  1.41255472e-01 -8.10785736e-02
 9.57549943e-02  2.73610111e-01  5.85622995e-02  7.91410001e-02
 1.47619459e-01  9.75521835e-02  6.74487028e-02  1.53125504e-01
 2.02791106e-02 -5.59403852e-02 -1.02109913e-01 -1.22913427e-01
 1.99873969e-01 -3.21872719e-01  1.38343165e-01  2.17196179e-01
 4.95201760e-03  8.52128333e-02 -1.45880901e-01 -2.10862397e-01
 1.20343357e-01  2.15598061e-01 -1.14038072e-02 -1.72172799e-01
 3.24157324e-01 -3.82818101e-01 -1.87580283e-01 -2.00827204e-01
-1.41863370e-01  9.63016678e-02 -2.01659119e-01  6.74342164e-02
 7.12185747e-02 -1.04314039e-01 -9.08169483e-02 -1.63495605e-01
 9.68230169e-02 -5.01176209e-02 -8.34015616e-02 -1.88998660e-01
-2.84065057e-01  1.16975197e-01 -2.80836800e-01 -9.33191327e-02
 3.79583269e-02 -1.22755412e-01  2.30408258e-01 -1.31968890e-01
 9.72824714e-03 -3.44272546e-01 -2.09522211e-03  2.45944018e-02
 2.94077607e-02  2.67568157e-01 -2.69460269e-02  1.25412311e-01
-3.47031083e-01  2.09328241e-01  1.25385338e-02  1.55654760e-01
-1.41368915e-01 -1.01749781e-01 -4.77312036e-04 -4.82325465e-02
-7.15727478e-02 -3.63658602e-02 -4.33504397e-02 -2.71410315e-01
-2.40079853e-01  2.01171435e-01  6.39005674e-02 -4.86787485e-02
-1.48623863e-01 -1.72130906e-01  1.97761227e-01 -3.13043504e-01
 1.07772898e-01 -1.54518908e-01  1.31855435e-01  3.39703669e-01
-4.51652340e-02 -4.05998340e-02  2.03610454e-01  8.84982054e-03
 3.05974297e-01  2.54736700e-01 -1.06925907e-01  1.27066655e-01
-1.88835779e-02 -1.56632041e-01  8.45142200e-02  5.70681135e-02
 1.01119358e-02 -6.62387316e-03 -2.18552410e-01  1.20985419e-02
-5.54006219e-01 -1.72367117e-01 -2.90325016e-01 -2.34816399e-01

-1.94243114e+00  4.36715446e-01 -2.80713863e-01 -6.33991309e-03
```

```
-1.94245114e+00  4.36713440e-01 -2.80713803e-01 -6.33991309e-03
-2.90035778e-01 -1.98732349e-01  2.96737137e-02  1.50873684e-01
 1.16943997e-01  1.39741722e-01 -1.82238609e-01  4.09714520e-02
 2.37176600e-01 -1.24515116e-01  1.41648743e-01  2.64206287e-01
-2.40551078e-01 -1.40415333e-01 -2.92432371e-01 -3.03761027e-02
-9.90320454e-02 -8.43648662e-02  1.81116706e-01 -4.05719699e-01
 1.22898740e-01 -8.80109292e-02  1.09543672e-01  2.96110858e-01
 1.85027885e-01  9.14976115e-02  9.63416424e-03  5.50340717e-02
-2.59328007e-02 -2.43942768e-01  2.54260096e-01 -1.03280950e-01
 1.56799018e-01 -9.58635926e-02 -4.31948365e-02  2.01228907e-01
 5.20033765e-02 -2.08030399e-01 -2.49149283e-01  3.11752465e-02
-2.39410711e-01  2.54421815e-01  3.50420005e-02  1.31625993e-01
-2.19027956e-01  2.75093693e-01  4.31276229e-02 -6.89266192e-02
 1.80694153e-01 -9.77254221e-02  6.52789959e-02  1.81468103e-01
-5.79288980e-01 -1.91501478e-01 -1.43298895e-01  1.56769073e-02
-2.28584041e-02 -7.96762354e-03  1.38764109e-01 -2.67804890e-02
 3.02808634e-01  1.63688874e-01 -1.98263925e-01  8.94007093e-02
-2.01132765e-01  8.29230669e-03 -3.17426319e-01 -4.07929287e-02
-1.63872993e-01  3.69860278e-01  2.90009047e-01  4.56005599e-02]
```

Vectorizing numerical features

1. Vectorizing price

In [0]:
```
# Standardizing the price data using StandardScaler(Uses mean and std f
or standardization)
priceScaler = StandardScaler();
priceScaler.fit(projectsData['price'].values.reshape(-1, 1));
priceStandardized = priceScaler.transform(projectsData['price'].values.
reshape(-1, 1));
```

In [0]:
```
print("Shape of standardized matrix of prices: ", priceStandardized.sha
pe);
equalsBorder(70);
print("Sample original prices: ");
equalsBorder(70);
```

```
print(projectsData['price'].values[0:5]);
print("Sample standardized prices: ");
equalsBorder(70);
print(priceStandardized[0:5]);
```

```
Shape of standardized matrix of prices:  (109245, 1)
======================================================================
Sample original prices:
======================================================================
[154.6  299.   516.85 232.9   67.98]
Sample standardized prices:
======================================================================
[[-0.39052147]
 [ 0.00240752]
 [ 0.5952024 ]
 [-0.17745817]
 [-0.62622444]]
```

2. Vectorizing quantity

In [0]:

```
# Standardizing the quantity data using StandardScaler(Uses mean and st
d for standardization)
quantityScaler = StandardScaler();
quantityScaler.fit(projectsData['quantity'].values.reshape(-1, 1));
quantityStandardized = quantityScaler.transform(projectsData['quantity'
].values.reshape(-1, 1));
```

In [0]:

```
print("Shape of standardized matrix of quantities: ", quantityStandardi
zed.shape);
equalsBorder(70);
print("Sample original quantities: ");
equalsBorder(70);
print(projectsData['quantity'].values[0:5]);
print("Sample standardized quantities: ");
equalsBorder(70);
print(quantityStandardized[0:5]);
```

```
Shape of standardized matrix of quantities:  (109245, 1)
```

```
======================================================================
Sample original quantities:
======================================================================
[23  1 22  4  4]
Sample standardized quantities:
======================================================================
[[ 0.23045805]
 [-0.6097785 ]
 [ 0.19226548]
 [-0.49520079]
 [-0.49520079]]
```

3. Vectorizing teacher_number_of_previously_posted_projects

In [0]:
```
# Standardizing the teacher_number_of_previously_posted_projects data u
sing StandardScaler(Uses mean and std for standardization)
previouslyPostedScaler = StandardScaler();
previouslyPostedScaler.fit(projectsData['teacher_number_of_previously_p
osted_projects'].values.reshape(-1, 1));
previouslyPostedStandardized = previouslyPostedScaler.transform(project
sData['teacher_number_of_previously_posted_projects'].values.reshape(-1
, 1));
```

In [0]:
```
print("Shape of standardized matrix of teacher_number_of_previously_pos
ted_projects: ", previouslyPostedStandardized.shape);
equalsBorder(70);
print("Sample original quantities: ");
equalsBorder(70);
print(projectsData['teacher_number_of_previously_posted_projects'].valu
es[0:5]);
print("Sample standardized teacher_number_of_previously_posted_project
s: ");
equalsBorder(70);
print(previouslyPostedStandardized[0:5]);
```

```
Shape of standardized matrix of teacher_number_of_previously_posted_pro
jects:  (109245, 1)
======================================================================
```

```
Sample original quantities:
==============================================================================
[0 7 1 4 1]
Sample standardized teacher_number_of_previously_posted_projects:
==============================================================================
[[-0.40153083]
 [-0.14952695]
 [-0.36553028]
 [-0.25752861]
 [-0.36553028]]
```

Taking 6k points(to avoid memory errors)

In [0]:

```
numberOfPoints = 6000;
# Categorical data
categoriesVectorsSub = categoriesVectors[0:numberOfPoints];
subCategoriesVectorsSub = subCategoriesVectors[0:numberOfPoints];
teacherPrefixVectorsSub = teacherPrefixVectors[0:numberOfPoints];
schoolStateVectorsSub = schoolStateVectors[0:numberOfPoints];
projectGradeVectorsSub = projectGradeVectors[0:numberOfPoints];

# Text data
bowEssayModelSub = bowEssayModel[0:numberOfPoints];
bowTitleModelSub = bowTitleModel[0:numberOfPoints];
tfIdfEssayModelSub = tfIdfEssayModel[0:numberOfPoints];
tfIdfTitleModelSub = tfIdfTitleModel[0:numberOfPoints];
word2VecEssaysVectorsSub = word2VecEssaysVectors[0:numberOfPoints];
word2VecTitlesVectorsSub = word2VecTitlesVectors[0:numberOfPoints];
tfIdfWeightedWord2VecEssaysVectorsSub = tfIdfWeightedWord2VecEssaysVect
ors[0:numberOfPoints];
tfIdfWeightedWord2VecTitlesVectorsSub = tfIdfWeightedWord2VecTitlesVect
ors[0:numberOfPoints];

# Numerical data
priceStandardizedSub = priceStandardized[0:numberOfPoints];
quantityStandardizedSub = quantityStandardized[0:numberOfPoints];
```

```
previouslyPostedStandardizedSub = previouslyPostedStandardized[0:number
OfPoints];
```

In [0]:
```
classesDataSub = projectsData['project_is_approved'][0:numberOfPoints].
values
```

In [0]:
```
classesDataSub.shape
```

Out[0]: (6000,)

Classification & Modelling using support vector machine

Classification using data(orginal dimensions) by support vector machine

Splitting Data(Only training and test)

In [121]:
```
projectsData = projectsData.dropna(subset = ['teacher_prefix']);
projectsData.shape
```

Out[121]: (109245, 30)

In [122]:
```
classesData = projectsData['project_is_approved']
print(classesData.shape)
```

(109245,)

In [0]:
```
trainingData, testData, classesTraining, classesTest = model_selection.
train_test_split(projectsData, classesData, test_size =  0.3, random_st
ate = 0, stratify = classesData);
```

```
trainingData, crossValidateData, classesTraining, classesCrossValidate
= model_selection.train_test_split(trainingData, classesTraining, test_
size = 0.3, random_state = 0, stratify = classesTraining);
```

In [124]:
```
print("Shapes of splitted data: ");
equalsBorder(70);

print("testData shape: ", testData.shape);
print("classesTest: ", classesTest.shape);
print("trainingData shape: ", trainingData.shape);
print("classesTraining shape: ", classesTraining.shape);
```

```
Shapes of splitted data:
======================================================================
testData shape:  (32774, 30)
classesTest:  (32774,)
trainingData shape:  (53529, 30)
classesTraining shape:  (53529,)
```

In [125]:
```
print("Number of negative points: ", trainingData[trainingData['project
_is_approved'] == 0].shape);
print("Number of positive points: ", trainingData[trainingData['project
_is_approved'] == 1].shape);
```

```
Number of negative points:  (8105, 30)
Number of positive points:  (45424, 30)
```

In [0]:
```
vectorizedFeatureNames = [];
```

Vectorizing categorical data

1. Vectorizing cleaned_categories(project_subject_categories cleaned) - One Hot Encoding

In [0]:
```
# Using CountVectorizer for performing one-hot-encoding by setting voca
```

```
bulary as list of all unique cleaned_categories
subjectsCategoriesVectorizer = CountVectorizer(vocabulary = list(sorted
CategoriesDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with cleaned_categories values
subjectsCategoriesVectorizer.fit(trainingData['cleaned_categories'].val
ues);
# Vectorizing categories using one-hot-encoding
categoriesVectors = subjectsCategoriesVectorizer.transform(trainingData
['cleaned_categories'].values);
```

In [128]:

```
print("Features used in vectorizing categories: ");
equalsBorder(70);
print(subjectsCategoriesVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of cleaned_categories matrix after vectorization(one-hot-e
ncoding): ", categoriesVectors.shape);
equalsBorder(70);
print("Sample vectors of categories: ");
equalsBorder(70);
print(categoriesVectors[0:4])
```

```
Features used in vectorizing categories:
======================================================================
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearn
ing', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Langua
ge']
======================================================================
Shape of cleaned_categories matrix after vectorization(one-hot-encodin
g):  (53529, 9)
======================================================================
Sample vectors of categories:
======================================================================
  (0, 3)	1
  (0, 7)	1
  (1, 7)	1
  (1, 8)	1
  (2, 6)	1
  (2, 7)	1
  (3, 7)	1
```

2. Vectorizing cleaned_sub_categories(project_subject_sub_categories cleaned) - One Hot Encoding

In [0]:

```
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique cleaned_sub_categories
subjectsSubCategoriesVectorizer = CountVectorizer(vocabulary = list(sor
tedDictionarySubCategories.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with cleaned_sub_categories values
subjectsSubCategoriesVectorizer.fit(trainingData['cleaned_sub_categorie
s'].values);
# Vectorizing sub categories using one-hot-encoding
subCategoriesVectors = subjectsSubCategoriesVectorizer.transform(traini
ngData['cleaned_sub_categories'].values);
```

In [130]:

```
print("Features used in vectorizing subject sub categories: ");
equalsBorder(70);
print(subjectsSubCategoriesVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of cleaned_categories matrix after vectorization(one-hot-e
ncoding): ", subCategoriesVectors.shape);
equalsBorder(70);
print("Sample vectors of categories: ");
equalsBorder(70);
print(subCategoriesVectors[0:4])
```

```
Features used in vectorizing subject sub categories:
======================================================================
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolveme
nt', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutri
tionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingA
rts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPre
p', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopme
nt', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Healt
h_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing',
'Mathematics', 'Literacy']
======================================================================
```

```
Shape of cleaned_categories matrix after vectorization(one-hot-encodin
g):  (53529, 30)
======================================================================
Sample vectors of categories:
======================================================================
  (0, 23)	1
  (0, 25)	1
  (1, 28)	1
  (1, 29)	1
  (2, 24)	1
  (2, 25)	1
  (3, 28)	1
```

3. Vectorizing teacher_prefix - One Hot Encoding

In [0]:

```
def giveCounter(data):
    counter = Counter();
    for dataValue in data:
        counter.update(str(dataValue).split());
    return counter
```

In [132]:

```
giveCounter(trainingData['teacher_prefix'].values)
```

Out[132]:

```
Counter({'Dr.': 4, 'Mr.': 5206, 'Mrs.': 28216, 'Ms.': 18934, 'Teacher':
1169})
```

In [0]:

```
teacherPrefixDictionary = dict(giveCounter(trainingData['teacher_prefi
x'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique teacher_prefix
teacherPrefixVectorizer = CountVectorizer(vocabulary = list(teacherPref
ixDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with teacher_prefix values
teacherPrefixVectorizer.fit(trainingData['teacher_prefix'].values);
# Vectorizing teacher_prefix using one-hot-encoding
teacherPrefixVectors = teacherPrefixVectorizer.transform(trainingData[
'teacher_prefix'].values);
```

In [134]:
```
print("Features used in vectorizing teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of teacher_prefix matrix after vectorization(one-hot-encod
ing): ", teacherPrefixVectors.shape);
equalsBorder(70);
print("Sample vectors of teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectors[0:100]);
```

```
Features used in vectorizing teacher_prefix:
======================================================================
['Ms.', 'Mrs.', 'Teacher', 'Mr.', 'Dr.']
======================================================================
Shape of teacher_prefix matrix after vectorization(one-hot-encoding):
(53529, 5)
======================================================================
Sample vectors of teacher_prefix:
======================================================================
  (21, 2)	1
```

In [135]:
```
teacherPrefixes = [prefix.replace('.', '') for prefix in trainingData[
'teacher_prefix'].values];
teacherPrefixes[0:5]
```

Out[135]:
```
['Ms', 'Ms', 'Mrs', 'Mrs', 'Mrs']
```

In [136]:
```
trainingData['teacher_prefix'] = teacherPrefixes;
trainingData.head(3)
```

Out[136]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
66637	174395	p233512	c9e73f31af5ad4c7d3a140e81554da3b	Ms	GA
76424	11981	p088047	e1aa00913e0009364b5c7c3c4ab9a6f5	Ms	WA
34433	11994	p210041	a6c5d41f4e18aca1530159f7cee84084	Mrs	NC

In [0]:

```
teacherPrefixDictionary = dict(giveCounter(trainingData['teacher_prefi
x'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique teacher_prefix
teacherPrefixVectorizer = CountVectorizer(vocabulary = list(teacherPref
ixDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with teacher_prefix values
teacherPrefixVectorizer.fit(trainingData['teacher_prefix'].values);
# Vectorizing teacher_prefix using one-hot-encoding
teacherPrefixVectors = teacherPrefixVectorizer.transform(trainingData[
'teacher_prefix'].values);
```

In [138]:

```
print("Features used in vectorizing teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of teacher_prefix matrix after vectorization(one-hot-encod
ing): ", teacherPrefixVectors.shape);
```

```
equalsBorder(70);
print("Sample vectors of teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectors[0:4]);
```

```
Features used in vectorizing teacher_prefix:
======================================================================
['Ms', 'Mrs', 'Teacher', 'Mr', 'Dr']
======================================================================
Shape of teacher_prefix matrix after vectorization(one-hot-encoding):
(53529, 5)
======================================================================
Sample vectors of teacher_prefix:
======================================================================
  (0, 0)	1
  (1, 0)	1
  (2, 1)	1
  (3, 1)	1
```

4. Vectorizing school_state - One Hot Encoding

In [0]:

```
schoolStateDictionary = dict(giveCounter(trainingData['school_state'].values));
# Using CountVectorizer for performing one-hot-encoding by setting vocabulary as list of all unique school states
schoolStateVectorizer = CountVectorizer(vocabulary = list(schoolStateDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with school_state values
schoolStateVectorizer.fit(trainingData['school_state'].values);
# Vectorizing school_state using one-hot-encoding
schoolStateVectors = schoolStateVectorizer.transform(trainingData['school_state'].values);
```

In [140]:

```
print("Features used in vectorizing school_state: ");
equalsBorder(70);
print(schoolStateVectorizer.get_feature_names());
equalsBorder(70);
```

```
print("Shape of school_state matrix after vectorization(one-hot-encodin
g): ", schoolStateVectors.shape);
equalsBorder(70);
print("Sample vectors of school_state: ");
equalsBorder(70);
print(schoolStateVectors[0:4]);
```

```
Features used in vectorizing school_state:
======================================================================
['GA', 'WA', 'NC', 'MI', 'NV', 'KY', 'CA', 'CT', 'PA', 'SC', 'WV', 'C
O', 'FL', 'AZ', 'MS', 'OH', 'LA', 'TX', 'NY', 'IN', 'MO', 'KS', 'IA',
'NJ', 'AR', 'MA', 'WI', 'OK', 'UT', 'MN', 'OR', 'DC', 'VA', 'AL', 'NM',
'TN', 'IL', 'HI', 'DE', 'MD', 'ID', 'SD', 'NH', 'NE', 'ME', 'MT', 'AK',
'ND', 'VT', 'WY', 'RI']
======================================================================
Shape of school_state matrix after vectorization(one-hot-encoding):  (5
3529, 51)
======================================================================
Sample vectors of school_state:
======================================================================
  (0, 0)	1
  (1, 1)	1
  (2, 2)	1
  (3, 3)	1
```

5. Vectorizing project_grade_category - One Hot Encoding

In [141]:
```
giveCounter(trainingData['project_grade_category'])
```

Out[141]:
```
Counter({'3-5': 18193,
         '6-8': 8300,
         '9-12': 5289,
         'Grades': 53529,
         'PreK-2': 21747})
```

In [142]:
```
cleanedGrades = []
for grade in trainingData['project_grade_category'].values:
    grade = grade.replace(' ', '');
```

```
    grade = grade.replace('-', 'to');
    cleanedGrades.append(grade);
cleanedGrades[0:4]
```

Out[142]: ['Grades3to5', 'GradesPreKto2', 'Grades3to5', 'Grades3to5']

In [143]:
```
trainingData['project_grade_category'] = cleanedGrades
trainingData.head(4)
```

Out[143]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
66637	174395	p233512	c9e73f31af5ad4c7d3a140e81554da3b	Ms	GA
76424	11981	p088047	e1aa00913e0009364b5c7c3c4ab9a6f5	Ms	WA
34433	11994	p210041	a6c5d41f4e18aca1530159f7cee84084	Mrs	NC
84559	145506	p030629	8c9462aaf17c6a5869fe54c62af8b23c	Mrs	MI

In [0]:
```
projectGradeDictionary = dict(giveCounter(trainingData['project_grade_c
ategory'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique project grade categories
```

```
projectGradeVectorizer = CountVectorizer(vocabulary = list(projectGrade
Dictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with project_grade_category values
projectGradeVectorizer.fit(trainingData['project_grade_category'].value
s);
# Vectorizing project_grade_category using one-hot-encoding
projectGradeVectors = projectGradeVectorizer.transform(trainingData['pr
oject_grade_category'].values);
```

In [145]:

```
print("Features used in vectorizing project_grade_category: ");
equalsBorder(70);
print(projectGradeVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of school_state matrix after vectorization(one-hot-encodin
g): ", projectGradeVectors.shape);
equalsBorder(70);
print("Sample vectors of school_state: ");
equalsBorder(70);
print(projectGradeVectors[0:4]);
```

```
Features used in vectorizing project_grade_category:
======================================================================
['Grades3to5', 'GradesPreKto2', 'Grades6to8', 'Grades9to12']
======================================================================
Shape of school_state matrix after vectorization(one-hot-encoding):  (5
3529, 4)
======================================================================
Sample vectors of school_state:
======================================================================
  (0, 0)	1
  (1, 1)	1
  (2, 0)	1
  (3, 0)	1
```

Vectorizing Text Data

In [146]:
```
preProcessedEssaysWithStopWords, preProcessedEssaysWithoutStopWords = p
reProcessingWithAndWithoutStopWords(trainingData['project_essay']);
preProcessedProjectTitlesWithStopWords, preProcessedProjectTitlesWithou
tStopWords = preProcessingWithAndWithoutStopWords(trainingData['project
_title']);
```

In [0]:
```
bagOfWordsVectorizedFeatures = [];
```

Bag of Words

1. Vectorizing project_essay

In [0]:
```
# Initializing countvectorizer for bag of words vectorization of prepro
cessed project essays
bowEssayVectorizer = CountVectorizer(min_df = 10, max_features = 5000);
# Transforming the preprocessed essays to bag of words vectors
bowEssayModel = bowEssayVectorizer.fit_transform(preProcessedEssaysWith
outStopWords);
```

In [149]:
```
print("Some of the Features used in vectorizing preprocessed essays: "
);
equalsBorder(70);
print(bowEssayVectorizer.get_feature_names()[-40:]);
equalsBorder(70);
print("Shape of preprocessed essay matrix after vectorization: ", bowEs
sayModel.shape);
equalsBorder(70);
print("Sample bag-of-words vector of preprocessed essay: ");
equalsBorder(70);
print(bowEssayModel[0])
```

```
Some of the Features used in vectorizing preprocessed essays:
======================================================================
```

```
['worrying', 'worse', 'worst', 'worth', 'worthwhile', 'worthy', 'woul
d', 'wow', 'write', 'writer', 'writers', 'writing', 'writings', 'writte
n', 'wrong', 'wrote', 'xylophone', 'xylophones', 'yard', 'year', 'yearb
ook', 'yearly', 'yearn', 'yearning', 'years', 'yes', 'yesterday', 'ye
t', 'yoga', 'york', 'young', 'younger', 'youngest', 'youth', 'youtube',
'zero', 'zest', 'zip', 'zone', 'zoo']
======================================================================
Shape of preprocessed essay matrix after vectorization:  (53529, 5000)
======================================================================
Sample bag-of-words vector of preprocessed essay:
======================================================================
  (0, 4549)	1
  (0, 2057)	2
  (0, 4482)	1
  (0, 2398)	1
  (0, 817)	1
  (0, 3956)	2
  (0, 4366)	9
  (0, 2284)	1
  (0, 3868)	2
  (0, 138)	1
  (0, 2657)	1
  (0, 4273)	1
  (0, 244)	1
  (0, 148)	3
  (0, 596)	1
  (0, 3038)	3
  (0, 2451)	3
  (0, 2136)	2
  (0, 2631)	2
  (0, 1693)	4
  (0, 2800)	1
  (0, 2838)	1
  (0, 4201)	1
  (0, 2676)	2
  (0, 3631)	1
  :	:
  (0, 4678)	1
  (0, 874)	1
```

```
  (0, 1643)	1
  (0, 2061)	1
  (0, 4839)	1
  (0, 2858)	1
  (0, 807)	2
  (0, 958)	1
  (0, 2972)	1
  (0, 2996)	1
  (0, 2036)	1
  (0, 4547)	1
  (0, 2030)	1
  (0, 1547)	1
  (0, 1554)	1
  (0, 4527)	1
  (0, 183)	1
  (0, 4457)	1
  (0, 4578)	1
  (0, 3665)	1
  (0, 819)	1
  (0, 3548)	1
  (0, 2329)	1
  (0, 3270)	1
  (0, 3015)	1
```

2. Vectorizing project_title

In [0]:

```
# Initializing countvectorizer for bag of words vectorization of prepro
cessed project titles
bowTitleVectorizer = CountVectorizer(min_df = 10);
# Transforming the preprocessed project titles to bag of words vectors
bowTitleModel = bowTitleVectorizer.fit_transform(preProcessedProjectTit
lesWithoutStopWords);
```

In [151]:

```
print("Some of the Features used in vectorizing preprocessed titles: "
);
equalsBorder(70);
print(bowTitleVectorizer.get_feature_names()[-40:]);
```

```
equalsBorder(70);
print("Shape of preprocessed title matrix after vectorization: ", bowTi
tleModel.shape);
equalsBorder(70);
print("Sample bag-of-words vector of preprocessed title: ");
equalsBorder(70);
print(bowTitleModel[0])
```

```
Some of the Features used in vectorizing preprocessed titles:
======================================================================
['wobble', 'wobbles', 'wobbling', 'wobbly', 'women', 'wonder', 'wonderf
ul', 'wonders', 'word', 'words', 'work', 'workers', 'working', 'works',
'worksheets', 'workshop', 'world', 'worlds', 'worms', 'worth', 'would',
'wow', 'wrestling', 'write', 'writer', 'writers', 'writing', 'written',
'xylophone', 'ye', 'year', 'yearbook', 'years', 'yes', 'yoga', 'yogis',
'young', 'youngest', 'youth', 'zone']
======================================================================
Shape of preprocessed title matrix after vectorization:  (53529, 2097)
======================================================================
Sample bag-of-words vector of preprocessed title:
======================================================================
  (0, 1772)	1
  (0, 329)	1
```

Tf-Idf Vectorization

1. Vectorizing project_essay

In [0]:

```
# Intializing tfidf vectorizer for tf-idf vectorization of preprocessed
 project essays
tfIdfEssayVectorizer = TfidfVectorizer(min_df = 10, max_features = 5000
);
# Transforming the preprocessed project essays to tf-idf vectors
tfIdfEssayModel = tfIdfEssayVectorizer.fit_transform(preProcessedEssays
WithoutStopWords);
```

In [153]:
```
print("Some of the Features used in tf-idf vectorizing preprocessed ess
ays: ");
equalsBorder(70);
print(tfIdfEssayVectorizer.get_feature_names()[-40:]);
equalsBorder(70);
print("Shape of preprocessed title matrix after tf-idf vectorization: "
, tfIdfEssayModel.shape);
equalsBorder(70);
print("Sample Tf-Idf vector of preprocessed essay: ");
equalsBorder(70);
print(tfIdfEssayModel[0])
```

```
Some of the Features used in tf-idf vectorizing preprocessed essays:
======================================================================
['worrying', 'worse', 'worst', 'worth', 'worthwhile', 'worthy', 'woul
d', 'wow', 'write', 'writer', 'writers', 'writing', 'writings', 'writte
n', 'wrong', 'wrote', 'xylophone', 'xylophones', 'yard', 'year', 'yearb
ook', 'yearly', 'yearn', 'yearning', 'years', 'yes', 'yesterday', 'ye
t', 'yoga', 'york', 'young', 'younger', 'youngest', 'youth', 'youtube',
'zero', 'zest', 'zip', 'zone', 'zoo']
======================================================================
Shape of preprocessed title matrix after tf-idf vectorization:  (53529,
5000)
======================================================================
Sample Tf-Idf vector of preprocessed essay:
======================================================================
  (0, 3015)	0.018051068352693163
  (0, 3270)	0.08620378214515413
  (0, 2329)	0.0868099712148866
  (0, 3548)	0.03989961379391851
  (0, 819)	0.03249360548155223
  (0, 3665)	0.04281871407058692
  (0, 4578)	0.0351796237345842
  (0, 4457)	0.06890520374374785
  (0, 183)	0.08712124261849882
  (0, 4527)	0.06478904776131371
  (0, 1554)	0.05814510281984422
  (0, 1547)	0.08338171723911092
  (0, 2030)	0.04214570871240633
  (0, 4547)	0.07880844018762262
```

```
(0, 2036)	0.09031933761596805
(0, 2996)	0.08874752356244359
(0, 2972)	0.06350584145816565
(0, 958)	0.0613390145189937
(0, 807)	0.13587268941092268
(0, 2858)	0.055859746638881234
(0, 4839)	0.03574434738102111
(0, 2061)	0.09267456514811052
(0, 1643)	0.03720683017383728
(0, 874)	0.0765942316269479
(0, 4678)	0.11049865549086603
:	:
(0, 3631)	0.08677371745168418
(0, 2676)	0.07932359724134007
(0, 4201)	0.08369698697657602
(0, 2838)	0.05433784357835223
(0, 2800)	0.027249812838612678
(0, 1693)	0.19919775680158697
(0, 2631)	0.04704082949651936
(0, 2136)	0.09124153230707235
(0, 2451)	0.20164827458339624
(0, 3038)	0.08633333603161536
(0, 596)	0.08078971350072964
(0, 148)	0.16733527409646612
(0, 244)	0.03227209350906117
(0, 4273)	0.06878347355128343
(0, 2657)	0.04644230701675306
(0, 138)	0.09111577094048866
(0, 3868)	0.1653640440161924
(0, 2284)	0.09654443358148647
(0, 4366)	0.15651690626229334
(0, 3956)	0.04010637372510135
(0, 817)	0.060203527703254454
(0, 2398)	0.0762562423159442
(0, 4482)	0.04255975892365114
(0, 2057)	0.07273237168369645
(0, 4549)	0.06613153877057763
```

2. Vectorizing project_title

In [0]:
```
# Intializing tfidf vectorizer for tf-idf vectorization of preprocessed
 project titles
tfIdfTitleVectorizer = TfidfVectorizer(min_df = 10);
# Transforming the preprocessed project titles to tf-idf vectors
tfIdfTitleModel = tfIdfTitleVectorizer.fit_transform(preProcessedProjec
tTitlesWithoutStopWords);
```

In [155]:
```
print("Some of the Features used in tf-idf vectorizing preprocessed tit
les: ");
equalsBorder(70);
print(tfIdfTitleVectorizer.get_feature_names()[-40:]);
equalsBorder(70);
print("Shape of preprocessed title matrix after tf-idf vectorization: "
, tfIdfTitleModel.shape);
equalsBorder(70);
print("Sample Tf-Idf vector of preprocessed title: ");
equalsBorder(70);
print(tfIdfTitleModel[0])
```

```
Some of the Features used in tf-idf vectorizing preprocessed titles:
======================================================================
['wobble', 'wobbles', 'wobbling', 'wobbly', 'women', 'wonder', 'wonderf
ul', 'wonders', 'word', 'words', 'work', 'workers', 'working', 'works',
'worksheets', 'workshop', 'world', 'worlds', 'worms', 'worth', 'would',
'wow', 'wrestling', 'write', 'writer', 'writers', 'writing', 'written',
'xylophone', 'ye', 'year', 'yearbook', 'years', 'yes', 'yoga', 'yogis',
'young', 'youngest', 'youth', 'zone']
======================================================================
Shape of preprocessed title matrix after tf-idf vectorization:  (53529,
2097)
======================================================================
Sample Tf-Idf vector of preprocessed title:
======================================================================
  (0, 329)      0.5001682594739306
  (0, 1772)     0.865928237335415
```

Average Word2Vector Vectorization

In [0]:
```
# stronging variables into pickle files python: http://www.jessicayung.
com/how-to-use-pickle-to-save-and-load-variables-in-python/
# We should have glove_vectors file for creating below model
with open('drive/My Drive/glove_vectors', 'rb') as f:
    gloveModel = pickle.load(f)
    gloveWords =  set(gloveModel.keys())
```

In [157]:
```
print("Glove vector of sample word: ");
equalsBorder(70);
print(gloveModel['technology']);
equalsBorder(70);
print("Shape of glove vector: ", gloveModel['technology'].shape);
```

```
Glove vector of sample word: 
======================================================================
[-0.26078   -0.36898   -0.022831   0.21666    0.16672   -0.20268
 -3.1219     0.33057    0.71512    0.28874    0.074368  -0.033203
  0.23783    0.21052    0.076562   0.13007   -0.31706   -0.45888
 -0.45463   -0.13191    0.49761    0.072704   0.16811    0.18846
 -0.16688   -0.21973    0.08575   -0.19577   -0.2101    -0.32436
 -0.56336    0.077996  -0.22758   -0.66569    0.14824    0.038945
  0.50881   -0.1352     0.49966   -0.4401    -0.022335  -0.22744
  0.22086    0.21865    0.36647    0.30495   -0.16565    0.038759
  0.28108   -0.2167     0.12453    0.65401    0.34584   -0.2557
 -0.046363  -0.31111   -0.020936  -0.17122   -0.77114    0.29289
 -0.14625    0.39541   -0.078938   0.051127   0.15076    0.085126
  0.183     -0.06755    0.26312    0.0087276  0.0066415  0.37033
  0.03496   -0.12627   -0.052626  -0.34897    0.14672    0.14799
 -0.21821   -0.042785   0.2661    -1.1105     0.31789    0.27278
  0.054468  -0.27458    0.42732   -0.44101   -0.19302   -0.32948
  0.61501   -0.22301   -0.36354   -0.34983   -0.16125   -0.17195
 -3.363      0.45146   -0.13753    0.31107    0.2061     0.33063
  0.45879    0.24256    0.042342   0.074837  -0.12869    0.12066
  0.42843   -0.4704    -0.18937    0.32685    0.26079    0.20518
 -0.18432   -0.47658    0.69193    0.18731   -0.12516    0.35447
 -0.1969    -0.58981   -0.88914    0.5176     0.13177   -0.078557
  0.032963  -0.19411    0.15109    0.10547   -0.1113    -0.61533
```

```
  0.0948    -0.3393    -0.20071   -0.30197    0.29531    0.28017
  0.16049    0.25294   -0.44266   -0.39412    0.13486    0.25178
 -0.044114   1.1519     0.32234   -0.34323   -0.10713   -0.15616
  0.031206   0.46636   -0.52761   -0.39296   -0.068424  -0.04072
  0.41508   -0.34564    0.71001   -0.364      0.2996     0.032281
  0.34035    0.23452    0.78342    0.48045   -0.1609     0.40102
 -0.071795  -0.16531    0.082153   0.52065    0.24194    0.17113
  0.33552   -0.15725   -0.38984    0.59337   -0.19388   -0.39864
 -0.47901    1.0835     0.24473    0.41309    0.64952    0.46846
  0.024386  -0.72087   -0.095061   0.10095   -0.025229   0.29435
 -0.57696    0.53166   -0.0058338 -0.3304     0.19661   -0.085206
  0.34225    0.56262    0.19924   -0.027111  -0.44567    0.17266
  0.20887   -0.40702    0.63954    0.50708   -0.31862   -0.39602
 -0.1714    -0.040006  -0.45077   -0.32482   -0.0316     0.54908
 -0.1121     0.12951   -0.33577   -0.52768   -0.44592   -0.45388
  0.66145    0.33023   -1.9089     0.5318     0.21626   -0.13152
  0.48258    0.68028   -0.84115   -0.51165    0.40017    0.17233
 -0.033749   0.045275   0.37398   -0.18252    0.19877    0.1511
  0.029803   0.16657   -0.12987   -0.50489    0.55311   -0.22504
  0.13085   -0.78459    0.36481   -0.27472    0.031805   0.53052
 -0.20078    0.46392   -0.63554    0.040289  -0.19142   -0.0097011
  0.068084  -0.10602    0.25567    0.096125  -0.10046    0.15016
 -0.26733   -0.26494    0.057888   0.062678  -0.11596    0.28115
  0.25375   -0.17954    0.20615    0.24189    0.062696   0.27719
 -0.42601   -0.28619   -0.44697   -0.082253  -0.73415   -0.20675
 -0.60289   -0.06728    0.15666   -0.042614   0.41368   -0.17367
 -0.54012    0.23883    0.23075    0.13608   -0.058634  -0.089705
  0.18469    0.023634   0.16178    0.23384    0.24267    0.091846 ]
======================================================================
Shape of glove vector:  (300,)
```

In [0]:

```
def getWord2VecVectors(texts):
    word2VecTextsVectors = [];
    for preProcessedText in tqdm(texts):
        word2VecTextVector = np.zeros(300);
        numberOfWordsInText = 0;
        for word in preProcessedText.split():
            if word in gloveWords:
                word2VecTextVector += gloveModel[word];
```

```
            numberOfWordsInText += 1;
        if numberOfWordsInText != 0:
            word2VecTextVector = word2VecTextVector / numberOfWordsInTe
xt;
        word2VecTextsVectors.append(word2VecTextVector);
    return word2VecTextsVectors;
```

1. Vectorizing project_essay

In [159]:
```
word2VecEssaysVectors = getWord2VecVectors(preProcessedEssaysWithoutSto
pWords);
```

In [160]:
```
print("Shape of Word2Vec vectorization matrix of essays: {},{}".format(
len(word2VecEssaysVectors), len(word2VecEssaysVectors[0])));
equalsBorder(70);
print("Sample essay: ");
equalsBorder(70);
print(preProcessedEssaysWithoutStopWords[0]);
equalsBorder(70);
print("Word2Vec vector of sample essay: ");
equalsBorder(70);
print(word2VecEssaysVectors[0]);
```

```
Shape of Word2Vec vectorization matrix of essays: 53529,300
======================================================================
Sample essay: 
======================================================================
third grade teacher inner city school students identified risk achievin
g grade level standards also active boys need interactive hands learnin
g experiences many may sound like quite feat honor charged providing ed
ucational learning experiences need therefore work tenaciously ensure r
eceive high quality education considering teachers 21st century countle
ss resources available help us achieve goal students eager learn howeve
r truly embrace experiences allow use technology truly goal facilitate 
opportunities maximize students academic progress support partners educ
ation like students reach stars words benjamin franklin tell forget tea
```

```
ch remember involve learn quote truly describes students students activ
e need interactive technology help grow become career college ready edu
cator atlanta public schools endeavor ensure student ready leaders futu
re support mission atlanta public school caring culture trust collabora
tion every student graduate ready college career want ensure students m
eet goal chromebooks help scaffold concepts risk active students studen
ts use chromebooks become motivated multi faceted global thinkers resou
rces give endless opportunities engage interactive hands experiences th
ank advance taking time read class project importantly partner educatio
n nannan
=======================================================================
Word2Vec vector of sample essay:
=======================================================================
[ 2.04543569e-02  2.07545385e-02  3.97038946e-02 -5.16370090e-02
 -9.58477898e-02 -3.00796796e-02 -2.97243546e+00  5.54538475e-02
  6.24832898e-02  9.82128958e-02 -2.80520186e-02  7.14508762e-02
  6.79008479e-02 -1.74055817e-01 -6.41051453e-02 -6.15239509e-02
  6.73092749e-02 -8.82404551e-02  5.57375383e-02 -2.23718964e-02
  8.42551216e-02  5.39485766e-02  2.04348341e-02 -9.78144850e-03
  6.14343765e-02  3.31668291e-02  1.20466743e-01 -3.68431916e-02
 -7.01159383e-02 -8.96005401e-02 -3.08510261e-01 -8.29343796e-02
  4.33746366e-02  6.54051036e-02  4.11941880e-03 -3.70316668e-02
 -6.56590814e-02 -1.28904024e-03 -1.13302267e-01 -4.82476114e-02
 -6.82111216e-02  6.65962706e-02  1.21791148e-01 -1.54549257e-01
  4.35683138e-02 -7.31888234e-02  8.15812060e-02  4.29595701e-02
  5.91722527e-02 -1.36467096e-01  1.63599551e-02  4.17764072e-04
  5.83761365e-02 -3.80841952e-02  4.68790299e-03 -1.03284434e-01
  1.82574216e-03 -4.54995683e-02 -1.62345655e-01  1.15280077e-01
 -4.54551916e-02 -1.04781402e-02  7.93058980e-02 -7.10872626e-02
 -3.57805078e-02  9.04414677e-02  3.52333198e-02 -8.62842365e-02
  1.84669286e-01 -1.54362054e-01 -1.07727418e-01  6.27495587e-02
 -9.91135329e-04 -1.54905075e-01 -1.00464093e-01 -1.01985782e-01
  6.14864886e-03 -2.66078725e-02 -4.60630096e-02 -3.50799714e-02
  6.41047480e-02 -4.42517444e-01 -5.59305156e-02 -4.38270317e-02
 -1.26194318e-01 -3.02254144e-02  1.19733359e-01 -1.04827645e-01
  1.29428951e-01  2.39440838e-03  9.64035240e-02  1.33687814e-02
 -3.30521156e-02  2.84618640e-02  9.93352317e-03 -1.51602778e-01
 -2.32553707e+00  1.15290568e-01  1.54815023e-01  1.97641831e-01
 -9.32669623e-02 -6.51458922e-03  1.34759735e-01 -9.78091491e-02
```

```
 1.42680931e-01  6.36353563e-02  9.28927960e-02 -1.68964296e-01
 5.76116354e-02  3.42074383e-02 -1.19521625e-01  1.42357665e-03
 1.72342241e-02  1.89282992e-01  2.71800599e-02 -2.36099305e-02
-3.19515849e-01  9.47798946e-02  8.99973749e-02  6.61103171e-02
-3.29540437e-02  3.08461952e-02 -6.45036890e-02 -9.43297431e-02
 1.38943126e-02  1.57894850e-02  9.70188257e-02 -2.65705281e-02
-3.39774695e-02  1.53546069e-01  8.09209695e-02 -1.98608344e-02
 9.26842515e-04 -4.37390653e-02  1.48740153e-01 -1.13663485e-01
 8.17597192e-02 -1.58441407e-02 -8.08748323e-03  2.47789760e-01
 4.87463892e-03  3.08445838e-03  3.55631168e-03 -3.63122455e-02
-6.16885478e-02  5.88615078e-02  3.80264563e-02 -1.08470359e-02
 8.08136240e-02 -3.63697982e-02  5.98065389e-03  2.81163030e-02
 1.49099294e-01 -4.99627120e-02 -8.02123946e-02  5.42734637e-02
 6.98784347e-02 -5.89527006e-02 -2.08781964e-02 -6.82661210e-02
 4.76494663e-02  4.01232335e-02 -4.15263874e-02 -8.92321760e-02
-5.82419323e-02  9.30203485e-02 -1.26124279e-01  1.22072986e-01
 8.77868774e-02 -4.74334018e-02 -9.09554545e-02  2.88386754e-02
-2.00670413e-02 -1.16855354e-01  1.42940589e-02 -2.32596503e-02
-3.53198251e-02  4.02083202e-02 -8.43523940e-02  8.10093293e-03
-2.09915498e-02  2.78207462e-01  4.06699419e-02  4.38479168e-02
-6.31782293e-02 -7.76219964e-02 -3.00407755e-02  1.69985132e-02
 1.39398743e-02 -2.51963066e-02 -6.50045884e-02 -1.30424251e-02
-7.58389934e-02  4.09315766e-02  2.25448407e-02 -9.33499120e-02
-8.84215485e-02  5.12995042e-02  7.29593503e-02  7.11705816e-02
 1.18154841e-01  1.25269473e-02 -5.72915940e-02  1.06128814e-01
-1.06368640e-01  2.13750713e-02  9.18381750e-02  3.09608544e-02
 8.39342387e-02 -6.84883293e-03 -6.86295509e-02 -5.93254749e-03
-1.37709590e-02 -1.89647329e-01 -8.17742299e-02  2.10635892e-02
-4.52247515e-02 -4.14746988e-02 -5.52902580e-02  5.76008880e-02
-1.50778684e-01 -7.53806174e-02 -1.40932065e-01 -5.41017251e-02
-1.84971951e+00  8.01600659e-02 -8.96777832e-02  9.00903473e-03
-2.51310371e-02 -1.38849304e-01  1.95983683e-02  1.43482611e-02
-3.48197743e-02  2.98574890e-02 -1.07686437e-01  3.06637883e-02
 4.56830030e-02 -3.46740653e-02  5.13837982e-02  1.51798896e-01
-1.65363725e-02  3.20647559e-02 -2.23584287e-01  6.96195585e-02
 8.37502874e-03 -1.63984796e-03 -6.95668189e-02 -7.84659751e-02
 4.52903359e-02 -8.38755360e-02  1.41688024e-02  1.12331315e-01
 8.49013832e-03 -7.36611365e-02  8.94757926e-02  3.90677246e-03
 1.55559977e-01  2.58652623e-02  1.33693125e-01 -1.21997134e-01
```

```
  4.64458381e-02  4.22075713e-02 -7.94508982e-03  5.39547503e-02
 -1.39836032e-02 -6.89185497e-02 -1.60573070e-01 -3.99081145e-02
  1.51480287e-02  7.79384965e-02 -7.51498749e-02 -1.29864305e-02
 -1.25145272e-01  1.19537827e-01 -5.90359898e-02  8.26312814e-02
  6.33621497e-03 -9.46705665e-02 -1.50488756e-02  8.54141011e-02
 -1.00302599e-01 -6.09363974e-02 -4.36927216e-02 -1.35023355e-02
  2.77190874e-02  1.14352888e-01  5.58733353e-03  4.04366551e-02
 -1.54433407e-02  3.46840228e-02  2.46053778e-02 -3.86298629e-02
 -1.16255043e-01 -4.41248295e-02  7.76928719e-02  2.32383605e-02
  1.23024677e-02  1.92424659e-01  1.81368320e-01  3.50246186e-02]
```

2. Vectorizing project_title

In [161]:

```
word2VecTitlesVectors = getWord2VecVectors(preProcessedProjectTitlesWit
houtStopWords);
```

In [162]:

```
print("Shape of Word2Vec vectorization matrix of project titles: {}, {}
".format(len(word2VecTitlesVectors), len(word2VecTitlesVectors[0])));
equalsBorder(70);
print("Sample title: ");
equalsBorder(70);
print(preProcessedProjectTitlesWithoutStopWords[0]);
equalsBorder(70);
print("Word2Vec vector of sample title: ");
equalsBorder(70);
print(word2VecTitlesVectors[0]);
```

```
Shape of Word2Vec vectorization matrix of project titles: 53529, 300
======================================================================
Sample title:
======================================================================
steaming chromebooks
======================================================================
Word2Vec vector of sample title:
======================================================================
[ 1.86585500e-01 -2.06677000e-01  1.69029400e-01 -9.09015000e-01
```

```
  6.89250000e-02  1.34635000e-01 -6.30880500e-01  9.68625000e-02
  3.24390000e-01 -5.98255000e-01  1.93035000e-01  2.73000000e-02
  6.40480000e-01 -2.96480000e-01 -1.43084500e-01  1.17845000e-01
 -2.51900000e-01 -1.28365500e-01 -3.59500950e-01  1.56255000e-01
 -4.87780000e-01 -2.21435000e-01  2.48296500e-01  1.12586500e-01
 -8.73900000e-02  1.46641000e-01  4.00420000e-01 -2.13070000e-02
 -1.16589000e-01 -3.88050000e-01 -5.23230000e-01  4.19790000e-01
  3.94555000e-01 -3.37480000e-02  6.45310000e-01 -6.50940000e-02
  5.29050000e-02  1.40730000e-01 -2.24210000e-01 -7.35465000e-02
  2.65775000e-01 -1.08428000e-01  3.86240000e-01 -4.73719500e-01
 -4.64885000e-01  2.58855000e-01 -3.70475000e-01  1.48070000e-01
 -5.75000000e-04  5.89500000e-02  2.41550000e-01 -1.22307500e-01
  1.43700000e-01 -1.05265000e-02 -2.34276500e-01 -1.87985000e-01
 -4.68195000e-01  9.27800000e-03 -2.36700000e-02  4.10425000e-01
  1.84120000e-01  1.81925000e-01  2.41420000e-01  4.19525000e-01
 -2.45390000e-01 -1.04420000e-01 -1.64397000e-01  2.55921000e-01
  6.82875000e-01  1.65986000e-01  1.79350000e-01  2.41350000e-01
 -1.95050000e-02  1.87040000e-01 -5.19580000e-02 -3.35400000e-02
  3.09955000e-01 -2.87745000e-01 -8.49350000e-02  2.87249500e-01
 -5.21810000e-02  4.30045000e-01  2.89104500e-01 -7.02400000e-02
 -1.77425000e-01 -5.75500000e-02 -9.49915000e-02 -1.50665000e-01
 -3.17471500e-01  4.67890000e-02 -5.73500000e-03  2.81490000e-01
 -1.53865000e-01  1.57073500e-01 -1.57965500e-01 -5.77880000e-01
 -9.49600000e-02 -3.28560000e-01 -8.05440000e-02  3.31145000e-01
  5.92160000e-02  1.34388000e-01  3.81404000e-01 -3.33900000e-02
 -1.62870000e-01  8.30545000e-02 -3.97865000e-01  1.55565000e-01
  1.36857000e-01 -1.01540750e-01  1.52500000e-02  2.48575000e-01
 -1.30378500e-01 -3.25766000e-01  3.83300000e-02 -3.82645700e-01
 -1.66295000e-01  2.03658500e-01 -6.80880000e-02  6.64050000e-02
 -4.38685000e-01  5.76740000e-01 -2.58265000e-01 -1.25510000e-01
  4.32100000e-02 -6.89650000e-02  3.91870000e-01 -3.19540000e-01
  1.23730350e-01  2.65620000e-01 -2.49780000e-01 -4.49670000e-01
  2.95964000e-01  4.38505000e-01 -2.00695000e-01  1.55770000e-01
  1.85280000e-01 -2.40720000e-01  4.46535000e-01 -2.03000000e-03
  6.16790000e-01  4.90425000e-01 -9.54850000e-03 -2.46805000e-01
  1.43775000e-01 -1.69950000e-02  3.16015000e-01  1.17790000e-01
  4.76000000e-03  2.08670000e-01  1.56264500e-01 -8.04800000e-02
 -3.00239500e-01 -1.88925500e-01  4.41285000e-01  1.14455000e-01
  2.66504000e-01 -1.48365000e-01  2.52200000e-02 -1.01510000e-01
```

```
 2.97710000e-01 -1.71469500e-01 -5.18860000e-01  1.81110000e-01
-3.25840000e-01  1.59665000e-01  7.19735000e-02  1.50125000e-01
-2.77280000e-01 -1.65619500e-01 -6.21040000e-01 -2.58886000e-01
-3.79105000e-01 -2.10685000e-02  5.98025000e-01  3.06193000e-01
-1.69880000e-01  6.80130000e-01 -2.80886500e-01  2.50200000e-02
 2.18475000e-01  3.18585000e-01  3.49562217e-01  1.97168300e-01
 3.49152000e-01 -2.66875000e-01 -6.95060000e-01  1.97935000e-01
 4.30525000e-01 -2.63055000e-01  2.90440000e-01  3.47225000e-02
-5.96100000e-02  3.92950000e-01  1.83865000e-01 -3.06000000e-02
 3.21983000e-01  4.25040000e-01 -3.84500000e-03 -2.04733700e-01
-4.89285000e-01 -9.56430000e-02  6.54550000e-02  6.47805000e-01
 8.51125000e-02 -1.22184500e-01 -7.21945000e-02  1.64900000e-02
 3.32350000e-02 -1.98638000e-01  1.53947000e-01  1.70985000e-01
-4.57315000e-01 -2.30720000e-01 -1.63711000e-01  2.85650000e-01
-7.62805000e-01  1.67435000e-01  1.34942000e-01  1.39250000e-02
-3.69410000e-01  8.26995000e-02 -3.27250000e-02 -4.71351850e-01
-2.94960000e-01  3.41900000e-01  2.61770000e-01  1.46935500e-01
 6.01490000e-02 -1.51680000e-01  1.42745000e-01 -1.79153500e-01
 3.23905000e-01  2.10705000e-01  2.39425000e-01  1.15730000e-01
 1.33711000e-01 -3.24600000e-01 -7.66790000e-01  2.98906500e-01
-4.32800000e-02 -4.45615000e-01  1.90379750e-01 -1.73793500e-01
-3.75000000e-03 -2.72780000e-01  1.77280000e-01 -3.94273600e-02
-9.34400000e-02 -5.34040000e-02  2.07742000e-01  8.34750000e-02
 3.23775000e-01  1.78810000e-01  2.59927900e-01 -3.93445000e-01
 4.14195000e-01 -6.83150000e-02 -3.51115000e-01 -1.62710500e-01
 8.56050000e-02  2.52135000e-01 -2.64650000e-01  4.77300000e-02
 3.90600000e-02 -1.07145000e-01 -9.90395000e-02  4.25135000e-01
-6.84350000e-02 -9.98900000e-02  3.15045500e-01 -1.21643000e-01
-3.10290000e-01  4.65215000e-01 -8.38220000e-02  1.87770000e-01
-5.03900000e-02  6.69125000e-02 -3.87745000e-01 -1.31255000e-01
 4.52585000e-01 -3.95270000e-01 -3.77054000e-01 -5.32945000e-01
-2.05935000e-01 -2.74801500e-01 -6.15330000e-02 -1.22235000e-01
 1.87785000e-01 -6.70300000e-02 -2.65410000e-01  5.43800000e-02
 2.00986500e-01 -1.50010000e-01  4.32850000e-01  1.01997500e-01
 1.11460500e-01 -1.16722000e-01  6.64000000e-02 -1.14385000e-01]
```

Tf-Idf Weighted Word2Vec Vectorization

1. Vectorizing project_essay

In [0]:

```
# Initializing tfidf vectorizer
tfIdfEssayTempVectorizer = TfidfVectorizer();
# Vectorizing preprocessed essays using tfidf vectorizer initialized ab
ove
tfIdfEssayTempVectorizer.fit(preProcessedEssaysWithoutStopWords);
# Saving dictionary in which each word is key and it's idf is value
tfIdfEssayDictionary = dict(zip(tfIdfEssayTempVectorizer.get_feature_na
mes(), list(tfIdfEssayTempVectorizer.idf_)));
# Creating set of all unique words used by tfidf vectorizer
tfIdfEssayWords = set(tfIdfEssayTempVectorizer.get_feature_names());
```

In [164]:

```
# Creating list to save tf-idf weighted vectors of essays
tfIdfWeightedWord2VecEssaysVectors = [];
# Iterating over each essay
for essay in tqdm(preProcessedEssaysWithoutStopWords):
    # Sum of tf-idf values of all words in a particular essay
    cumulativeSumTfIdfWeightOfEssay = 0;
    # Tf-Idf weighted word2vec vector of a particular essay
    tfIdfWeightedWord2VecEssayVector = np.zeros(300);
    # Splitting essay into list of words
    splittedEssay = essay.split();
    # Iterating over each word
    for word in splittedEssay:
        # Checking if word is in glove words and set of words used by t
fIdf essay vectorizer
        if (word in gloveWords) and (word in tfIdfEssayWords):
            # Tf-Idf value of particular word in essay
            tfIdfValueWord = tfIdfEssayDictionary[word] * (essay.count(
word) / len(splittedEssay));
            # Making tf-idf weighted word2vec
            tfIdfWeightedWord2VecEssayVector += tfIdfValueWord * gloveM
odel[word];
            # Summing tf-idf weight of word to cumulative sum
            cumulativeSumTfIdfWeightOfEssay += tfIdfValueWord;
    if cumulativeSumTfIdfWeightOfEssay != 0:
        # Taking average of sum of vectors with tf-idf cumulative sum
        tfIdfWeightedWord2VecEssayVector = tfIdfWeightedWord2VecEssayVe
```

```
ctor / cumulativeSumTfIdfWeightOfEssay;
    # Appending the above calculated tf-idf weighted vector of particul
ar essay to list of vectors of essays
    tfIdfWeightedWord2VecEssaysVectors.append(tfIdfWeightedWord2VecEssa
yVector);
```

In [165]:

```
print("Shape of Tf-Idf weighted Word2Vec vectorization matrix of projec
t essays: {}, {}".format(len(tfIdfWeightedWord2VecEssaysVectors), len(t
fIdfWeightedWord2VecEssaysVectors[0])));
equalsBorder(70);
print("Sample Essay: ");
equalsBorder(70);
print(preProcessedEssaysWithoutStopWords[0]);
equalsBorder(70);
print("Tf-Idf Weighted Word2Vec vector of sample essay: ");
equalsBorder(70);
print(tfIdfWeightedWord2VecEssaysVectors[0]);
```

```
Shape of Tf-Idf weighted Word2Vec vectorization matrix of project essay
s: 53529, 300
======================================================================
Sample Essay:
======================================================================
third grade teacher inner city school students identified risk achievin
g grade level standards also active boys need interactive hands learnin
g experiences many may sound like quite feat honor charged providing ed
ucational learning experiences need therefore work tenaciously ensure r
eceive high quality education considering teachers 21st century countle
ss resources available help us achieve goal students eager learn howeve
r truly embrace experiences allow use technology truly goal facilitate 
opportunities maximize students academic progress support partners educ
ation like students reach stars words benjamin franklin tell forget tea
ch remember involve learn quote truly describes students students activ
e need interactive technology help grow become career college ready edu
cator atlanta public schools endeavor ensure student ready leaders futu
re support mission atlanta public school caring culture trust collabora
tion every student graduate ready college career want ensure students m

eet goal chromebooks help scaffold concepts risk active students studen
```

```
cet goal chromebooks help scaffold concepts risk active students studen
ts use chromebooks become motivated multi faceted global thinkers resou
rces give endless opportunities engage interactive hands experiences th
ank advance taking time read class project importantly partner educatio
n nannan
======================================================================
Tf-Idf Weighted Word2Vec vector of sample essay:
======================================================================
[ 3.77641104e-02  2.87515761e-02  3.64392397e-02 -2.51074409e-02
 -8.38105647e-02 -7.57442248e-02 -2.91080120e+00  5.46845993e-02
  1.06096961e-01  1.50433207e-01 -1.99995724e-02  5.53792123e-02
  2.78238259e-02 -1.83537569e-01 -6.43857895e-02 -7.35416375e-02
  9.15225833e-02 -7.73431568e-02  7.41488140e-02 -5.15891642e-03
  7.44805198e-02  5.80347270e-02  4.23444145e-02 -2.77460050e-02
  4.64730976e-02  5.38944439e-02  1.37153667e-01 -4.99762514e-02
 -7.61492541e-02 -1.13714235e-01 -3.01039515e-01 -7.73390401e-02
  2.85406827e-02  7.39748176e-02 -8.68365742e-03 -2.59379358e-02
 -4.84487721e-02  1.07423908e-02 -1.41962565e-01 -3.15083842e-02
 -8.21734979e-02  2.23690113e-02  1.54454193e-01 -1.74442916e-01
  5.02397179e-02 -1.12234991e-01  8.36296258e-02  5.48051985e-02
  3.98414982e-02 -1.51839962e-01  1.65648589e-02 -1.18798127e-02
  6.75889576e-02 -3.51683301e-02 -2.68851269e-02 -1.14156546e-01
  9.25676183e-03 -2.83452598e-02 -1.84962935e-01  9.11067138e-02
 -3.57143793e-02  2.07288618e-02  1.10715987e-01 -6.89107763e-02
 -4.53091914e-02  8.60475201e-02  3.43334147e-02 -8.60162298e-02
  2.16029373e-01 -1.76320125e-01 -1.22441100e-01  5.31714212e-02
 -3.12787806e-02 -1.49376687e-01 -9.86316751e-02 -9.38878907e-02
 -1.81109116e-02 -4.36637357e-02 -3.71285674e-02 -4.00130979e-02
  5.71872732e-02 -4.58292774e-01 -4.81915168e-02 -5.53614531e-02
 -1.51323969e-01 -3.76933689e-03  1.10322568e-01 -9.98910876e-02
  1.49105244e-01 -6.65136262e-03  8.74105359e-02 -2.78258988e-03
 -4.61411111e-02  3.70600814e-02  1.54886174e-02 -1.36074347e-01
 -2.35410769e+00  1.32119112e-01  1.66644155e-01  2.24406886e-01
 -1.12997163e-01 -1.03596407e-02  1.53747010e-01 -1.06668550e-01
  1.45614052e-01  8.11199312e-02  1.24443353e-01 -1.68013649e-01
  9.90313790e-02  3.95880158e-02 -1.36353220e-01  1.61782951e-02
  1.68100146e-02  1.57974529e-01  4.43057533e-02  1.87201763e-03
 -3.57689934e-01  1.27564190e-01  1.10946379e-01  3.70642033e-02
 -4.05146380e-02  1.60133603e-02 -6.83403261e-02 -9.01827366e-02

  3.09529317e-02  2.93637013e-02  1.35949524e-01 -1.35330941e-02
```

```
-7.98175176e-02  1.73529614e-01  9.83789088e-02 -6.85322356e-02
 2.53100335e-02 -2.62774520e-02  1.44533192e-01 -1.04439262e-01
 7.26944643e-02 -2.30838115e-02 -1.78907909e-02  2.23639542e-01
-4.54535742e-02 -2.59199711e-02 -2.63285447e-02 -3.07072453e-02
-8.73425356e-02  9.16445058e-02  6.81553778e-02  1.32519042e-02
 3.96351228e-02 -3.58350390e-02  3.79403746e-02  2.04613802e-02
 1.60662902e-01 -3.81425208e-02 -7.37404476e-02  5.26489886e-02
 7.00554586e-02 -6.16114425e-02 -3.83823884e-02 -9.59688499e-02
 3.71668949e-02  5.85580837e-02 -5.73204184e-02 -1.20474288e-01
-7.57629435e-02  1.18421585e-01 -1.42510387e-01  1.25505141e-01
 6.81299570e-02 -2.30357855e-02 -1.18569820e-01  2.39310672e-02
-3.59106142e-02 -1.28817308e-01 -2.21306511e-02 -3.53565780e-02
-1.91513115e-02  9.47076984e-04 -1.00322364e-01  2.01071793e-02
-3.78355622e-02  2.88203290e-01  4.95652108e-02  7.50121569e-02
-6.35455979e-02 -8.21260315e-02 -1.70649618e-02  2.27908233e-02
 5.54309130e-03 -6.02317747e-02 -8.42523142e-02  1.78637269e-02
-9.79342095e-02  3.72043345e-02  1.12024623e-02 -9.79850842e-02
-1.24969475e-01  1.52205532e-02  7.99246419e-02  1.02878256e-01
 8.76765001e-02  4.00742138e-02 -8.04469214e-04  1.04950892e-01
-1.15030741e-01  1.49436816e-02  8.91523400e-02  3.05647939e-02
 9.65676308e-02 -7.80059336e-03 -6.64379237e-02 -9.61813812e-03
-3.28535252e-02 -2.23854636e-01 -9.65448977e-02  2.76816961e-02
-6.58345031e-02 -4.00928439e-02 -4.65936944e-02  8.14956567e-02
-1.55129853e-01 -8.21316313e-02 -1.66145237e-01 -4.98109543e-02
-1.82642489e+00  6.18341486e-02 -9.52408815e-02 -4.77902954e-03
-4.04834061e-02 -1.52767720e-01  1.04157218e-02  6.97960082e-02
-5.40962772e-02  5.26969986e-02 -1.01829555e-01  1.04159419e-02
 6.17427150e-02 -2.02531158e-02  8.11001660e-02  1.92616219e-01
-5.86853992e-02  2.47528846e-02 -2.07882415e-01  8.45935654e-02
 2.55343770e-03 -1.35930991e-02 -7.74395771e-02 -9.68503237e-02
 4.63826083e-02 -1.10941909e-01  1.92488792e-02  1.34928671e-01
 3.32381325e-02 -1.03461806e-01  9.38714008e-02  1.87974335e-02
 1.65987410e-01  2.73435111e-03  1.46750394e-01 -1.28205169e-01
 3.52778500e-02  7.90625271e-02 -1.78828506e-02  1.03116168e-01
 7.21360600e-03 -6.41967874e-02 -2.07532136e-01 -8.60026099e-02
 2.80042820e-02  8.88864853e-02 -7.32027927e-02 -3.60664544e-03
-9.79051733e-02  1.61808407e-01 -7.03135902e-02  8.38246119e-02
-2.73289864e-02 -1.34080163e-01 -1.23953261e-02  9.57355185e-02

-1.93847198e-01 -3.11262816e-02 -2.36442024e-02 -5.46218077e-02
```

```
  3.11275315e-02  1.24065613e-01  5.48052495e-03  1.79827773e-02
  3.08831061e-03  4.26440885e-02  2.55502931e-02 -4.69813528e-02
 -1.43106149e-01 -5.51938360e-02  7.47254555e-02  2.96492111e-02
  1.53614615e-02  2.05773569e-01  1.99488694e-01  2.94221155e-03]
```

2. Vectorizing project_title

In [0]:

```
# Initializing tfidf vectorizer
tfIdfTitleTempVectorizer = TfidfVectorizer();
# Vectorizing preprocessed titles using tfidf vectorizer initialized ab
ove
tfIdfTitleTempVectorizer.fit(preProcessedProjectTitlesWithoutStopWords
);
# Saving dictionary in which each word is key and it's idf is value
tfIdfTitleDictionary = dict(zip(tfIdfTitleTempVectorizer.get_feature_na
mes(), list(tfIdfTitleTempVectorizer.idf_)));
# Creating set of all unique words used by tfidf vectorizer
tfIdfTitleWords = set(tfIdfTitleTempVectorizer.get_feature_names());
```

In [167]:

```
# Creating list to save tf-idf weighted vectors of project titles
tfIdfWeightedWord2VecTitlesVectors = [];
# Iterating over each title
for title in tqdm(preProcessedProjectTitlesWithoutStopWords):
    # Sum of tf-idf values of all words in a particular project title
    cumulativeSumTfIdfWeightOfTitle = 0;
    # Tf-Idf weighted word2vec vector of a particular project title
    tfIdfWeightedWord2VecTitleVector = np.zeros(300);
    # Splitting title into list of words
    splittedTitle = title.split();
    # Iterating over each word
    for word in splittedTitle:
        # Checking if word is in glove words and set of words used by t
fIdf title vectorizer
        if (word in gloveWords) and (word in tfIdfTitleWords):
            # Tf-Idf value of particular word in title
            tfIdfValueWord = tfIdfTitleDictionary[word] * (title.count(
word) / len(splittedTitle));
```

```
            # Making tf-idf weighted word2vec
            tfIdfWeightedWord2VecTitleVector += tfIdfValueWord * gloveM
odel[word];
            # Summing tf-idf weight of word to cumulative sum
            cumulativeSumTfIdfWeightOfTitle += tfIdfValueWord;
    if cumulativeSumTfIdfWeightOfTitle != 0:
        # Taking average of sum of vectors with tf-idf cumulative sum
        tfIdfWeightedWord2VecTitleVector = tfIdfWeightedWord2VecTitleVe
ctor / cumulativeSumTfIdfWeightOfTitle;
    # Appending the above calculated tf-idf weighted vector of particul
ar title to list of vectors of project titles
    tfIdfWeightedWord2VecTitlesVectors.append(tfIdfWeightedWord2VecTitl
eVector);
```

In [168]:

```
print("Shape of Tf-Idf weighted Word2Vec vectorization matrix of projec
t titles: {}, {}".format(len(tfIdfWeightedWord2VecTitlesVectors), len(t
fIdfWeightedWord2VecTitlesVectors[0])));
equalsBorder(70);
print("Sample Title: ");
equalsBorder(70);
print(preProcessedProjectTitlesWithoutStopWords[0]);
equalsBorder(70);
print("Tf-Idf Weighted Word2Vec vector of sample title: ");
equalsBorder(70);
print(tfIdfWeightedWord2VecTitlesVectors[0]);
```

```
Shape of Tf-Idf weighted Word2Vec vectorization matrix of project title
s: 53529, 300
======================================================================
Sample Title:
======================================================================
steaming chromebooks
======================================================================
Tf-Idf Weighted Word2Vec vector of sample title:
======================================================================
[ 1.23127012e-01 -1.26576136e-01  1.22900151e-01 -9.04520968e-01
  1.56239341e-01  3.44128678e-02 -7.78678727e-01  1.00572720e-01
  2.85222176e-01 -7.02731536e-01  1.04655050e-01 -9.30549132e-02
```

```
 5.77443070e-01 -1.78197402e-01 -9.17383449e-02  3.83058564e-02
-2.83177498e-01 -1.38251701e-01 -2.61149243e-01  1.60411678e-01
-4.46285507e-01 -2.18785703e-01  3.19401402e-01  7.35623190e-02
-3.57882861e-02  2.07040414e-01  3.20659970e-01 -2.41541573e-02
-1.41526660e-01 -3.92863982e-01 -3.56167678e-01  3.46091625e-01
 3.33050887e-01 -6.77436043e-02  5.69857922e-01 -7.40742993e-02
 4.94591739e-02  1.34360443e-01 -2.08415961e-01 -6.07368357e-02
 3.66337163e-01 -9.39560660e-02  2.55788575e-01 -3.71281678e-01
-4.54559570e-01  2.21565379e-01 -2.37676826e-01  5.58091450e-02
-1.79092619e-01  1.54078361e-01  2.56348042e-01 -1.33965610e-01
 1.44760254e-01 -2.45317318e-03 -1.89431631e-01 -1.92701257e-01
-4.53553586e-01 -1.72101480e-02  1.05794227e-02  4.56426912e-01
 1.67335320e-01  2.75049320e-01  2.30630040e-01  4.07038901e-01
-2.59403561e-01  7.44752496e-03 -1.23664766e-01  3.06812933e-01
 6.39668304e-01  1.42466564e-01  1.00698418e-01  2.35339216e-01
-9.85327584e-02  1.91588919e-01 -8.24381656e-02  8.15832549e-02
 3.55587429e-01 -2.59826312e-01  9.91089387e-03  1.89438241e-01
-5.14848735e-02  5.38751843e-01  2.05728495e-01 -5.71769189e-02
-2.48897109e-01  6.37875225e-02 -5.21426395e-02  6.66054446e-02
-3.96328841e-01  4.90348111e-02  2.77821512e-02  9.58839392e-02
-7.26230239e-02  1.21847760e-01 -2.13177567e-01 -6.06769249e-01
-1.94714924e-01 -3.05116602e-01 -8.99593784e-02  3.52131875e-01
 2.43550570e-02  1.75674190e-01  2.65877527e-01  2.43242398e-02
-2.83412332e-01  7.00193978e-02 -4.10530486e-01  9.36639135e-03
 1.52073522e-01 -7.35459568e-02 -2.58187848e-02  3.70194989e-01
-1.15422089e-01 -4.00806828e-01 -3.79949135e-02 -4.85526312e-01
-2.37791205e-01  1.60940029e-01 -3.65823866e-02 -3.39590349e-02
-4.36070510e-01  5.13978843e-01 -2.65155313e-01 -3.37658851e-02
-3.38056851e-02 -2.69365000e-04  3.50667345e-01 -1.74245017e-01
 8.85707018e-02  2.88854560e-01 -2.67640999e-01 -4.77175028e-01
 2.24288140e-01  4.29887757e-01 -1.98356283e-01  1.91872190e-01
 2.70282398e-01 -2.33378543e-01  4.19601599e-01  6.37780488e-02
 6.20257245e-01  4.18299603e-01  1.57052283e-02 -9.48258642e-02
 1.37409459e-01 -4.68200035e-02  2.74495072e-01  4.59604571e-02
 2.14995551e-01  1.84120832e-01  1.27764680e-01 -1.88002089e-01
-3.76104033e-01 -2.13692743e-01  2.90730246e-01  1.14726757e-01
 3.22624110e-01  8.35080362e-02  1.02575716e-01 -1.57979244e-01
 2.98839867e-01 -2.01073751e-01 -4.56484390e-01  2.62013284e-01
-1.50686546e-01  4.92285504e-02  5.64213651e-02  6.63180656e-02
```

```
 -2.66021493e-01 -1.90034931e-01 -4.15827269e-01 -3.02218267e-01
 -3.06138897e-01 -6.71557687e-03  5.59080740e-01  2.42634778e-01
 -1.78485194e-01  6.36555160e-01 -2.17465763e-01 -5.20144270e-02
  2.17922115e-01  2.81260572e-01  2.55944510e-01  2.50414398e-01
  2.60163475e-01 -2.33657719e-01 -6.39589429e-01  2.86309595e-01
  3.77687662e-01 -2.95415510e-01  3.20919630e-01  4.09939302e-02
 -2.04971918e-01  3.90615299e-01  1.98696510e-01 -1.05106952e-01
  4.14842791e-01  2.96205731e-01  8.20128301e-02 -1.51609531e-01
 -5.80867132e-01 -7.91107994e-02  1.88720903e-02  5.66865571e-01
  1.17291616e-01 -9.37623253e-02 -6.55444841e-02 -1.38087562e-01
  7.34001591e-02 -1.61948922e-01  1.03985735e-01  2.84551341e-01
 -3.86552405e-01 -3.51131139e-01 -2.04310970e-01  2.87633960e-01
 -7.46873075e-01  1.58568758e-01  1.17060117e-01 -8.35072715e-02
 -1.82984652e-01  8.83168391e-02 -4.75765907e-02 -3.45674742e-01
 -4.42340685e-01  1.48537484e-01  2.60243877e-01  1.75381504e-01
  9.39917242e-02  7.61247995e-03  3.78909493e-02 -2.04114052e-01
  2.88363727e-01  1.11821573e-01  6.64763927e-02  1.96832880e-02
  1.20744573e-01 -2.51662010e-01 -8.28426645e-01  2.30155041e-01
 -1.13784225e-01 -5.15477986e-01  2.43344332e-01 -1.06947553e-01
 -2.15739254e-01 -1.46403593e-01  2.83080515e-01 -5.01684952e-02
  3.01492239e-02 -5.97328606e-02  1.47942594e-01  1.45565466e-01
  2.89800012e-01  1.85158138e-01  1.92387005e-01 -4.72108631e-01
  3.50800636e-01 -2.15075865e-01 -3.23892439e-01 -1.97647883e-01
  3.11719256e-02  7.40939598e-02 -1.35722022e-01 -3.19845139e-02
  7.95424321e-02 -2.65188281e-02 -4.86076802e-02  4.31104285e-01
 -1.42456003e-01 -1.57432886e-01  2.49412955e-01 -1.32547286e-01
 -3.63838191e-01  5.08941114e-01 -4.30616528e-02  2.72239593e-01
  2.81973245e-02  1.04663305e-01 -4.64352380e-01 -5.01308300e-02
  6.79285281e-01 -3.70870767e-01 -2.98475779e-01 -5.28630355e-01
 -2.27738485e-01 -2.22546900e-01 -6.85389775e-02 -6.32797809e-02
  3.13694199e-01 -1.97941939e-01 -2.48737771e-01 -1.61389757e-01
  2.73812976e-01 -1.14603017e-03  3.86755718e-01  1.45198172e-01
  9.50507909e-02 -8.60420997e-02 -6.12562090e-02 -2.47105529e-01]
```

Method for vectorizing unknown essays using our training data tf-idf weighted model

In [0]:
```
def getAvgTfIdfEssayVectors(arrayOfTexts):
```

```
    # Creating list to save tf-idf weighted vectors of essays
    tfIdfWeightedWord2VecEssaysVectors = [];
    # Iterating over each essay
    for essay in tqdm(arrayOfTexts):
        # Sum of tf-idf values of all words in a particular essay
        cumulativeSumTfIdfWeightOfEssay = 0;
        # Tf-Idf weighted word2vec vector of a particular essay
        tfIdfWeightedWord2VecEssayVector = np.zeros(300);
        # Splitting essay into list of words
        splittedEssay = essay.split();
        # Iterating over each word
        for word in splittedEssay:
            # Checking if word is in glove words and set of words used
 by tfIdf essay vectorizer
            if (word in gloveWords) and (word in tfIdfEssayWords):
                # Tf-Idf value of particular word in essay
                tfIdfValueWord = tfIdfEssayDictionary[word] * (essay.co
unt(word) / len(splittedEssay));
                # Making tf-idf weighted word2vec
                tfIdfWeightedWord2VecEssayVector += tfIdfValueWord * gl
oveModel[word];
                # Summing tf-idf weight of word to cumulative sum
                cumulativeSumTfIdfWeightOfEssay += tfIdfValueWord;
        if cumulativeSumTfIdfWeightOfEssay != 0:
            # Taking average of sum of vectors with tf-idf cumulative s
um
            tfIdfWeightedWord2VecEssayVector = tfIdfWeightedWord2VecEss
ayVector / cumulativeSumTfIdfWeightOfEssay;
        # Appending the above calculated tf-idf weighted vector of part
icular essay to list of vectors of essays
        tfIdfWeightedWord2VecEssaysVectors.append(tfIdfWeightedWord2Vec
EssayVector);
    return tfIdfWeightedWord2VecEssaysVectors;
```

Method for vectorizing unknown titles using our training data tf-idf weighted model

In [0]:
```
def getAvgTfIdfTitleVectors(arrayOfTexts):
    # Creating list to save tf-idf weighted vectors of project titles
    tfIdfWeightedWord2VecTitlesVectors = [];
    # Iterating over each title
    for title in tqdm(arrayOfTexts):
        # Sum of tf-idf values of all words in a particular project tit
le
        cumulativeSumTfIdfWeightOfTitle = 0;
        # Tf-Idf weighted word2vec vector of a particular project title
        tfIdfWeightedWord2VecTitleVector = np.zeros(300);
        # Splitting title into list of words
        splittedTitle = title.split();
        # Iterating over each word
        for word in splittedTitle:
            # Checking if word is in glove words and set of words used
 by tfIdf title vectorizer
            if (word in gloveWords) and (word in tfIdfTitleWords):
                # Tf-Idf value of particular word in title
                tfIdfValueWord = tfIdfTitleDictionary[word] * (title.co
unt(word) / len(splittedTitle));
                # Making tf-idf weighted word2vec
                tfIdfWeightedWord2VecTitleVector += tfIdfValueWord * gl
oveModel[word];
                # Summing tf-idf weight of word to cumulative sum
                cumulativeSumTfIdfWeightOfTitle += tfIdfValueWord;
        if cumulativeSumTfIdfWeightOfTitle != 0:
            # Taking average of sum of vectors with tf-idf cumulative s
um
            tfIdfWeightedWord2VecTitleVector = tfIdfWeightedWord2VecTit
leVector / cumulativeSumTfIdfWeightOfTitle;
        # Appending the above calculated tf-idf weighted vector of part
icular title to list of vectors of project titles
        tfIdfWeightedWord2VecTitlesVectors.append(tfIdfWeightedWord2Vec
TitleVector);
    return tfIdfWeightedWord2VecTitlesVectors;
```

Vectorizing numerical features

1. Vectorizing price

In [0]:
```
# Standardizing the price data using StandardScaler(Uses mean and std f
or standardization)
priceScaler = MinMaxScaler();
priceScaler.fit(trainingData['price'].values.reshape(-1, 1));
priceStandardized = priceScaler.transform(trainingData['price'].values.
reshape(-1, 1));
```

In [172]:
```
print("Shape of standardized matrix of prices: ", priceStandardized.sha
pe);
equalsBorder(70);
print("Sample original prices: ");
equalsBorder(70);
print(trainingData['price'].values[0:5]);
print("Sample standardized prices: ");
equalsBorder(70);
print(priceStandardized[0:5]);
```

```
Shape of standardized matrix of prices:  (53529, 1)
======================================================================
Sample original prices:
======================================================================
[159.   509.85 289.92 190.24 438.99]
Sample standardized prices:
======================================================================
[[0.01583663]
 [0.05092745]
 [0.0289308 ]
 [0.01896115]
 [0.04384028]]
```

2. Vectorizing quantity

In [0]:
```
# Standardizing the quantity data using StandardScaler(Uses mean and st
d for standardization)
quantityScaler = MinMaxScaler();
```

```
quantityScaler.fit(trainingData['quantity'].values.reshape(-1, 1));
quantityStandardized = quantityScaler.transform(trainingData['quantity'
].values.reshape(-1, 1));
```

In [174]:
```
print("Shape of standardized matrix of quantities: ", quantityStandardi
zed.shape);
equalsBorder(70);
print("Sample original quantities: ");
equalsBorder(70);
print(trainingData['quantity'].values[0:5]);
print("Sample standardized quantities: ");
equalsBorder(70);
print(quantityStandardized[0:5]);
```

```
Shape of standardized matrix of quantities:  (53529, 1)
======================================================================
Sample original quantities:
======================================================================
[ 4  1 12 17  2]
Sample standardized quantities:
======================================================================
[[0.00322928]
 [0.        ]
 [0.01184069]
 [0.01722282]
 [0.00107643]]
```

3. Vectorizing teacher_number_of_previously_posted_projects

In [0]:
```
# Standardizing the teacher_number_of_previously_posted_projects data u
sing StandardScaler(Uses mean and std for standardization)
previouslyPostedScaler = MinMaxScaler();
previouslyPostedScaler.fit(trainingData['teacher_number_of_previously_p
osted_projects'].values.reshape(-1, 1));
previouslyPostedStandardized = previouslyPostedScaler.transform(trainin
gData['teacher_number_of_previously_posted_projects'].values.reshape(-1
, 1));
```

In [176]:
```
print("Shape of standardized matrix of teacher_number_of_previously_pos
ted_projects: ", previouslyPostedStandardized.shape);
equalsBorder(70);
print("Sample original quantities: ");
equalsBorder(70);
print(trainingData['teacher_number_of_previously_posted_projects'].valu
es[0:5]);
print("Sample standardized teacher_number_of_previously_posted_project
s: ");
equalsBorder(70);
print(previouslyPostedStandardized[0:5]);
```

```
Shape of standardized matrix of teacher_number_of_previously_posted_pro
jects:  (53529, 1)
======================================================================
Sample original quantities:
======================================================================
[ 1  9 25  0  0]
Sample standardized teacher_number_of_previously_posted_projects:
======================================================================
[[0.00221729]
 [0.01995565]
 [0.05543237]
 [0.        ]
 [0.        ]]
```

In [0]:
```
numberOfPoints = previouslyPostedStandardized.shape[0];
# Categorical data
categoriesVectorsSub = categoriesVectors[0:numberOfPoints];
subCategoriesVectorsSub = subCategoriesVectors[0:numberOfPoints];
teacherPrefixVectorsSub = teacherPrefixVectors[0:numberOfPoints];
schoolStateVectorsSub = schoolStateVectors[0:numberOfPoints];
projectGradeVectorsSub = projectGradeVectors[0:numberOfPoints];

# Text data
bowEssayModelSub = bowEssayModel[0:numberOfPoints];
bowTitleModelSub = bowTitleModel[0:numberOfPoints];
tfIdfEssayModelSub = tfIdfEssayModel[0:numberOfPoints];
tfIdfTitleModelSub = tfIdfTitleModel[0:numberOfPoints];
```

```
# Numerical data
priceStandardizedSub = priceStandardized[0:numberOfPoints];
quantityStandardizedSub = quantityStandardized[0:numberOfPoints];
previouslyPostedStandardizedSub = previouslyPostedStandardized[0:number
OfPoints];

# Classes
classesTrainingSub = classesTraining;
```

In [2]:
```
supportVectorMachineResultsDataFrame = pd.DataFrame(columns =  ['Vector
izer', 'Model', 'Hyper Parameter - alpha', 'AUC', 'Data']);
supportVectorMachineResultsDataFrame
```

Out[2]:

	Vectorizer	Model	Hyper Parameter - alpha	AUC	Data

Preparing cross validate data for analysis

In [179]:
```
# Test data categorical features transformation
categoriesTransformedCrossValidateData = subjectsCategoriesVectorizer.t
ransform(crossValidateData['cleaned_categories']);
subCategoriesTransformedCrossValidateData = subjectsSubCategoriesVector
izer.transform(crossValidateData['cleaned_sub_categories']);
teacherPrefixTransformedCrossValidateData = teacherPrefixVectorizer.tra
nsform(crossValidateData['teacher_prefix']);
schoolStateTransformedCrossValidateData = schoolStateVectorizer.transfo
rm(crossValidateData['school_state']);
projectGradeTransformedCrossValidateData = projectGradeVectorizer.trans
form(crossValidateData['project_grade_category']);

# Test data text features transformation
preProcessedEssaysTemp = preProcessingWithAndWithoutStopWords(crossVali
dateData['project_essay'])[1];
preProcessedTitlesTemp = preProcessingWithAndWithoutStopWords(crossVali
dateData['project_title'])[1];
bowEssayTransformedCrossValidateData = bowEssayVectorizer.transform(pre
```

```
ProcessedEssaysTemp);
bowTitleTransformedCrossValidateData = bowTitleVectorizer.transform(pre
ProcessedTitlesTemp);
tfIdfEssayTransformedCrossValidateData = tfIdfEssayVectorizer.transform
(preProcessedEssaysTemp);
tfIdfTitleTransformedCrossValidateData = tfIdfTitleVectorizer.transform
(preProcessedTitlesTemp);
avgWord2VecEssayTransformedCrossValidateData = getWord2VecVectors(prePr
ocessedEssaysTemp);
avgWord2VecTitleTransformedCrossValidateData = getWord2VecVectors(prePr
ocessedTitlesTemp);
tfIdfWeightedWord2VecEssayTransformedCrossValidateData = getAvgTfIdfEss
ayVectors(preProcessedEssaysTemp);
tfIdfWeightedWord2VecTitleTransformedCrossValidateData = getAvgTfIdfTit
leVectors(preProcessedTitlesTemp);


# Test data numerical features transformation
priceTransformedCrossValidateData = priceScaler.transform(crossValidate
Data['price'].values.reshape(-1, 1));
quantityTransformedCrossValidateData = quantityScaler.transform(crossVa
lidateData['quantity'].values.reshape(-1, 1));
previouslyPostedTransformedCrossValidateData = previouslyPostedScaler.t
ransform(crossValidateData['teacher_number_of_previously_posted_project
s'].values.reshape(-1, 1));
```

Preparing Test data for analysis

In [180]:

```
# Test data categorical features transformation
categoriesTransformedTestData = subjectsCategoriesVectorizer.transform(
```

```
testData['cleaned_categories']);
subCategoriesTransformedTestData = subjectsSubCategoriesVectorizer.tran
sform(testData['cleaned_sub_categories']);
teacherPrefixTransformedTestData = teacherPrefixVectorizer.transform(te
stData['teacher_prefix']);
schoolStateTransformedTestData = schoolStateVectorizer.transform(testDa
ta['school_state']);
projectGradeTransformedTestData = projectGradeVectorizer.transform(test
Data['project_grade_category']);

# Test data text features transformation
preProcessedEssaysTemp = preProcessingWithAndWithoutStopWords(testData[
'project_essay'])[1];
preProcessedTitlesTemp = preProcessingWithAndWithoutStopWords(testData[
'project_title'])[1];
bowEssayTransformedTestData = bowEssayVectorizer.transform(preProcessed
EssaysTemp);
bowTitleTransformedTestData = bowTitleVectorizer.transform(preProcessed
TitlesTemp);
tfIdfEssayTransformedTestData = tfIdfEssayVectorizer.transform(preProce
ssedEssaysTemp);
tfIdfTitleTransformedTestData = tfIdfTitleVectorizer.transform(preProce
ssedTitlesTemp);
avgWord2VecEssayTransformedTestData = getWord2VecVectors(preProcessedEs
saysTemp);
avgWord2VecTitleTransformedTestData = getWord2VecVectors(preProcessedTi
tlesTemp);
tfIdfWeightedWord2VecEssayTransformedTestData = getAvgTfIdfEssayVectors
(preProcessedEssaysTemp);
tfIdfWeightedWord2VecTitleTransformedTestData = getAvgTfIdfTitleVectors
(preProcessedTitlesTemp);

# Test data numerical features transformation
priceTransformedTestData = priceScaler.transform(testData['price'].valu
es.reshape(-1, 1));
quantityTransformedTestData = quantityScaler.transform(testData['quanti
ty'].values.reshape(-1, 1));
previouslyPostedTransformedTestData = previouslyPostedScaler.transform(
```

```
testData['teacher_number_of_previously_posted_projects'].values.reshape
(-1, 1));
```

Classification using original data support vector machine

In [181]:
```
techniques = ['Bag of words', 'Tf-Idf', 'Average Word2Vector', 'Tf-Idf
 Weighted Word2Vector'];
for index, technique in enumerate(techniques):
    trainingMergedData = hstack((categoriesVectorsSub,\
                                 subCategoriesVectorsSub,\
                                 teacherPrefixVectorsSub,\
                                 schoolStateVectorsSub,\
                                 projectGradeVectorsSub,\
                                 priceStandardizedSub,\
                                 previouslyPostedStandardizedSub));
    crossValidateMergedData = hstack((categoriesTransformedCrossValidat
eData,\
                                      subCategoriesTransformedCross
ValidateData,\
                                      teacherPrefixTransformedCross
ValidateData,\
                                      schoolStateTransformedCrossVa
lidateData,\
                                      projectGradeTransformedCrossV
alidateData,\
                                      priceTransformedCrossValidate
Data,\
                                      previouslyPostedTransformedCr
ossValidateData));
```

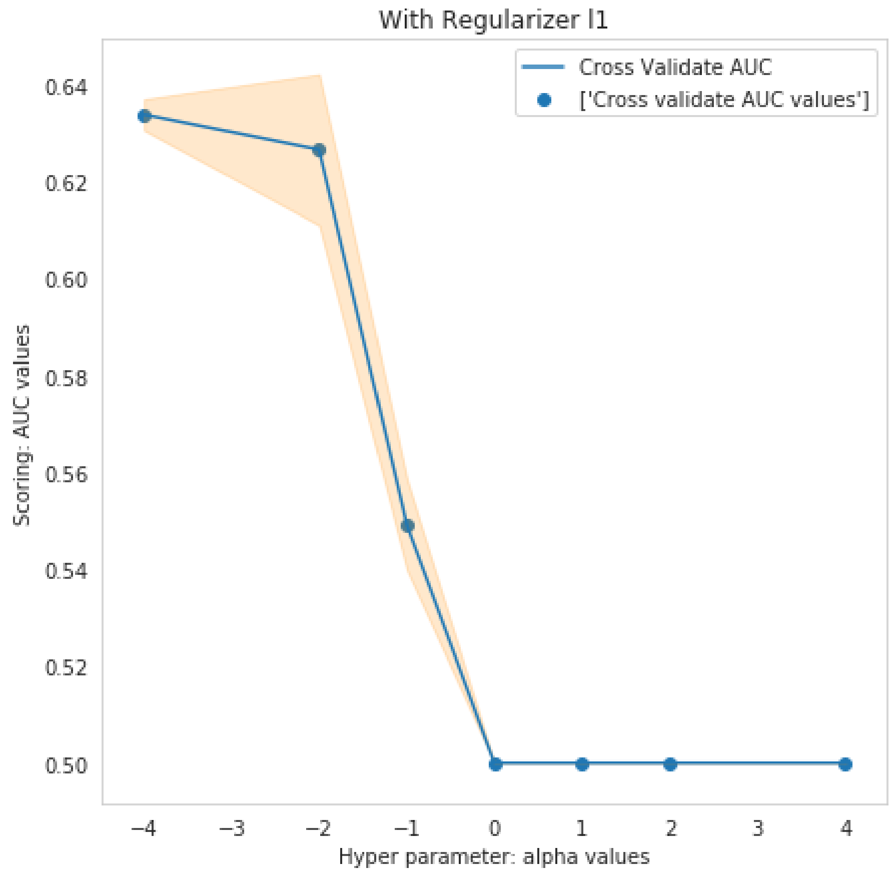
```
    testMergedData = hstack((categoriesTransformedTestData,\
                                        subCategoriesTransformedTestD
ata,\
                                        teacherPrefixTransformedTestD
ata,\
                                        schoolStateTransformedTestDat
a,\
                                        projectGradeTransformedTestDa
ta,\
                                        priceTransformedTestData,\
                                        previouslyPostedTransformedTe
stData));
    if(index == 0):
        trainingMergedData = hstack((trainingMergedData,\
                                    bowTitleModelSub,\
                                    bowEssayModelSub));
        crossValidateMergedData = hstack((crossValidateMergedData,\
                                    bowTitleTransformedCrossValidateData,\
                                    bowEssayTransformedCrossValidateData
));
        testMergedData = hstack((testMergedData,\
                                    bowTitleTransformedTestData,\
                                    bowEssayTransformedTestData));
    elif(index == 1):
        trainingMergedData = hstack((trainingMergedData,\
                                    tfIdfTitleModelSub,\
                                    tfIdfEssayModelSub));
        crossValidateMergedData = hstack((crossValidateMergedData,\
                                    tfIdfTitleTransformedCrossValidateData
,\
                                    tfIdfEssayTransformedCrossValidateData
));
        testMergedData = hstack((testMergedData,\
                                    tfIdfTitleTransformedTestData,\
                                    tfIdfEssayTransformedTestData));
    elif(index == 2):
        trainingMergedData = hstack((trainingMergedData,\
                                    word2VecTitlesVectors,\
                                    word2VecEssaysVectors));
```

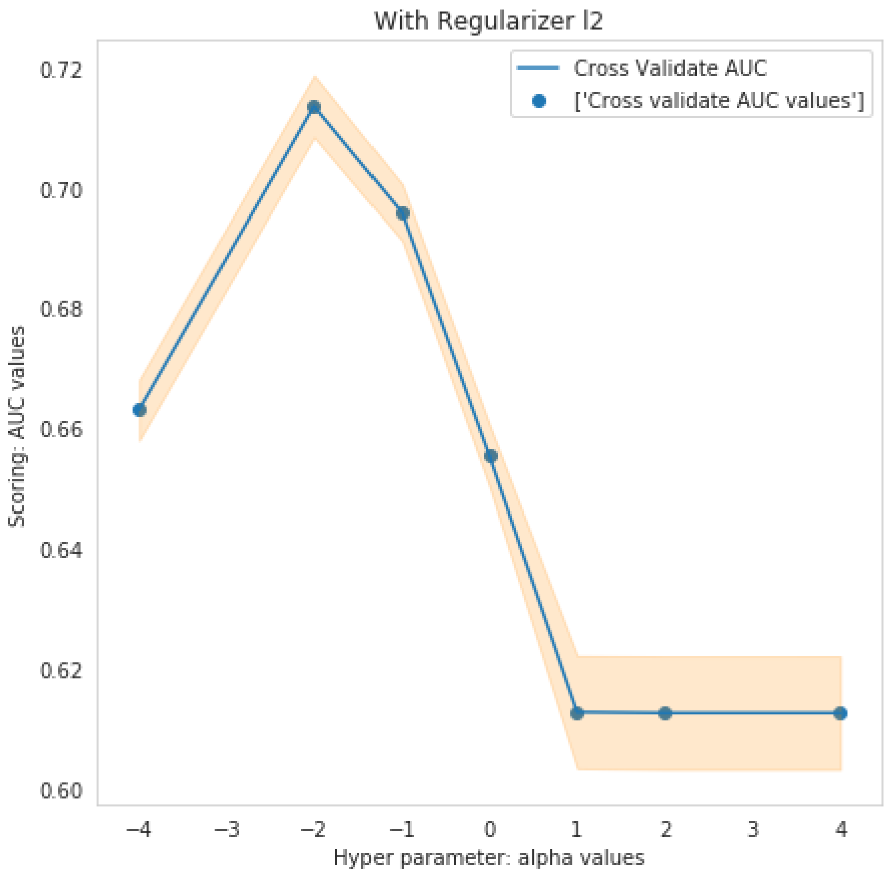
```
        crossValidateMergedData = hstack((crossValidateMergedData,\
                                         avgWord2VecTitleTransformedCrossValida
teData,\
                                         avgWord2VecEssayTransformedCrossValida
teData));
        testMergedData = hstack((testMergedData,\
                                         avgWord2VecTitleTransformedTestData,\
                                         avgWord2VecEssayTransformedTestData));
    elif(index == 3):
        trainingMergedData = hstack((trainingMergedData,\
                                              tfIdfWeightedWord2VecTitlesVectors
,\
                                              tfIdfWeightedWord2VecEssaysVectors
));
        crossValidateMergedData = hstack((crossValidateMergedData,\
                                         tfIdfWeightedWord2VecTitleTransformedC
rossValidateData,\
                                         tfIdfWeightedWord2VecEssayTransformedC
rossValidateData));
        testMergedData = hstack((testMergedData,\
                                         tfIdfWeightedWord2VecTitleTransformedT
estData,\
                                         tfIdfWeightedWord2VecEssayTransformedT
estData));

    svmClassifier = linear_model.SGDClassifier(loss = 'hinge', class_we
ight = 'balanced');
    tunedParameters = {'alpha': [0.0001, 0.01, 0.1, 1, 10, 100, 10000],
 'penalty': ['l1', 'l2']};
    classifier = GridSearchCV(svmClassifier, tunedParameters, cv = 5, s
coring = 'roc_auc');
    classifier.fit(trainingMergedData, classesTrainingSub);

    testScoresDataFrame = pd.DataFrame(data = np.hstack((classifier.cv_
results_['param_alpha'].data[:, None], classifier.cv_results_['param_pe
nalty'].data[:, None], classifier.cv_results_['mean_test_score'][:, Non
e], classifier.cv_results_['std_test_score'][:, None])), columns = ['al
pha', 'penalty', 'mts', 'stdts']);
    testScoresDataFrame
```

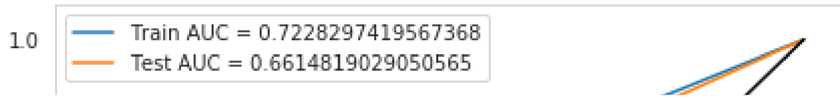
```
    crossValidateAucMeanValues = classifier.cv_results_['mean_test_scor
e'];
    crossValidateAucStdValues = classifier.cv_results_['std_test_score'
];

    testScoresDataFrame['logAlphaValues'] = [math.log10(x) for x in tes
tScoresDataFrame['alpha']];

    plt.plot(testScoresDataFrame[testScoresDataFrame['penalty'] == 'l1'
]['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penalty']
 == 'l1']['mts'], label = "Cross Validate AUC");
    plt.scatter(testScoresDataFrame[testScoresDataFrame['penalty'] ==
'l1']['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penal
ty'] == 'l1']['mts'], label = ['Cross validate AUC values']);
    plt.gca().fill_between(testScoresDataFrame[testScoresDataFrame['pen
alty'] == 'l1']['logAlphaValues'].values, np.array(testScoresDataFrame[
testScoresDataFrame['penalty'] == 'l1']['mts'].values - testScoresDataF
rame[testScoresDataFrame['penalty'] == 'l1']['stdts'].values, dtype = f
loat),\
                           np.array(testScoresDataFrame[testScoresDataF
rame['penalty'] == 'l1']['mts'].values + testScoresDataFrame[testScores
DataFrame['penalty'] == 'l1']['stdts'].values, dtype = float), alpha =
0.2, color = 'darkorange');
    plt.xlabel('Hyper parameter: alpha values');
    plt.ylabel('Scoring: AUC values');
    plt.title("With Regularizer l1");
    plt.grid();
    plt.legend();
    plt.show();

    plt.plot(testScoresDataFrame[testScoresDataFrame['penalty'] == 'l2'
]['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penalty']
 == 'l2']['mts'], label = "Cross Validate AUC");
    plt.scatter(testScoresDataFrame[testScoresDataFrame['penalty'] ==
'l2']['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penal
ty'] == 'l2']['mts'], label = ['Cross validate AUC values']);
    plt.gca().fill_between(testScoresDataFrame[testScoresDataFrame['pen
alty'] == 'l2']['logAlphaValues'].values, np.array(testScoresDataFrame[
```

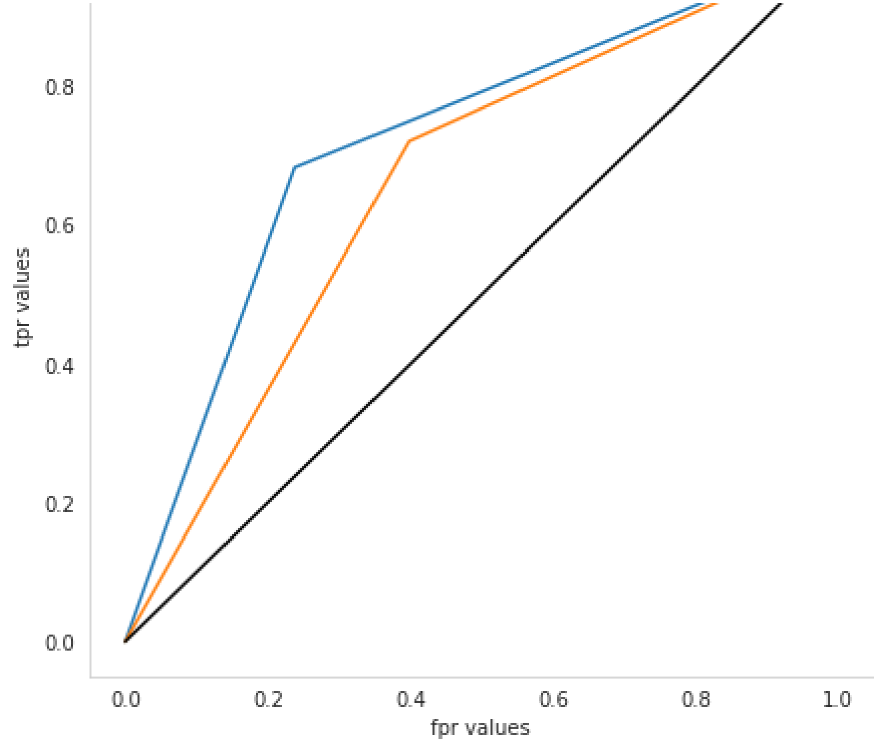
```
testScoresDataFrame['penalty'] == 'l2']['mts'].values - testScoresDataF
rame[testScoresDataFrame['penalty'] == 'l2']['stdts'].values, dtype = f
loat),\
                                   np.array(testScoresDataFrame[testScoresDataF
rame['penalty'] == 'l2']['mts'].values + testScoresDataFrame[testScores
DataFrame['penalty'] == 'l2']['stdts'].values, dtype = float), alpha =
0.2, color = 'darkorange');
    plt.xlabel('Hyper parameter: alpha values');
    plt.ylabel('Scoring: AUC values');
    plt.title("With Regularizer l2");
    plt.grid();
    plt.legend();
    plt.show();

    optimalHypParamValue = classifier.best_params_['alpha'];
    optimalHypParam2Value = classifier.best_params_['penalty'];
    svmClassifier = linear_model.SGDClassifier(loss = 'hinge', class_we
ight = 'balanced', alpha = optimalHypParamValue, penalty = optimalHypPa
ram2Value);
    svmClassifier.fit(trainingMergedData, classesTrainingSub);
    predScoresTraining = svmClassifier.predict(trainingMergedData);
    fprTrain, tprTrain, thresholdTrain = roc_curve(classesTraining, pre
dScoresTraining);
    predScoresTest = svmClassifier.predict(testMergedData);
    fprTest, tprTest, thresholdTest = roc_curve(classesTest, predScores
Test);

    plt.plot(fprTrain, tprTrain, label = "Train AUC = " + str(auc(fprTr
ain, tprTrain)));
    plt.plot(fprTest, tprTest, label = "Test AUC = " + str(auc(fprTest,
 tprTest)));
    plt.plot([0, 1], [0, 1], 'k-');
    plt.xlabel("fpr values");
    plt.ylabel("tpr values");
    plt.grid();
    plt.legend();
    plt.show();

    areaUnderRocValueTest = auc(fprTest, tprTest);
```

```
    print("Results of analysis using {} vectorized text features merged
 with other features using support vector machine classifier: ".format(
technique));
    print("Optimal Alpha value: ", optimalHypParamValue);
    equalsBorder(40);
    print("Optimal Regularizer: ", optimalHypParam2Value);
    equalsBorder(40);
    print("AUC value of test data: ", str(areaUnderRocValueTest));
    # Predicting classes of test data projects
    predictionClassesTest = svmClassifier.predict(testMergedData);
    equalsBorder(40);
    # Printing confusion matrix
    confusionMatrix = confusion_matrix(classesTest, predictionClassesTe
st);
    # Creating dataframe for generated confusion matrix
    confusionMatrixDataFrame = pd.DataFrame(data = confusionMatrix, ind
ex = ['Actual: NO', 'Actual: YES'], columns = ['Predicted: NO', 'Predic
ted: YES']);
    print("Confusion Matrix : ");
    equalsBorder(60);
    sbrn.heatmap(confusionMatrixDataFrame, annot = True, fmt = 'd', cma
p="YlGnBu");
    plt.show();
    # Adding results to results dataframe
    supportVectorMachineResultsDataFrame = supportVectorMachineResultsD
ataFrame.append({'Vectorizer': technique, 'Model': 'SVM(SGD - hinge los
s)', 'Hyper Parameter - alpha': optimalHypParamValue, 'AUC': areaUnderR
ocValueTest}, ignore_index = True);
```

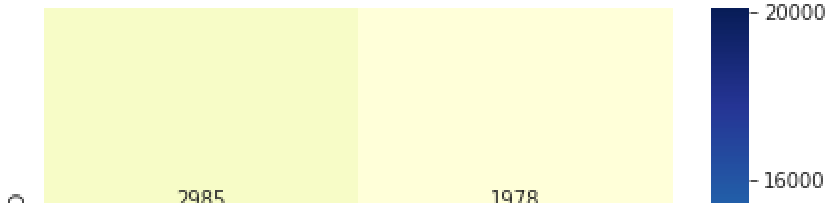

With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
Scoring: AUC values
0.64
0.62
0.60
0.58
0.56
0.54
0.52
0.50
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

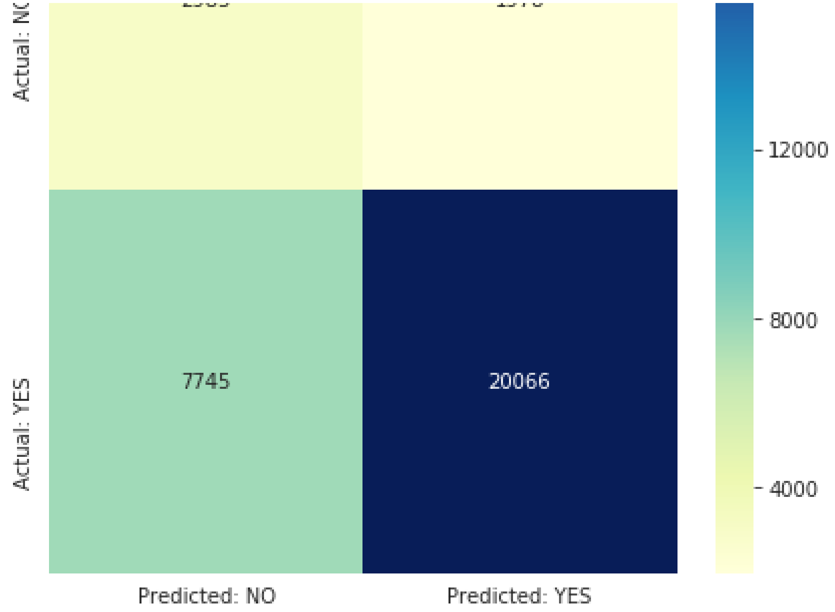

With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
0.72
0.70
0.68
0.66
0.64
0.62
0.60
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values


1.0
Train AUC = 0.7228297419567368
Test AUC = 0.6614819029050565

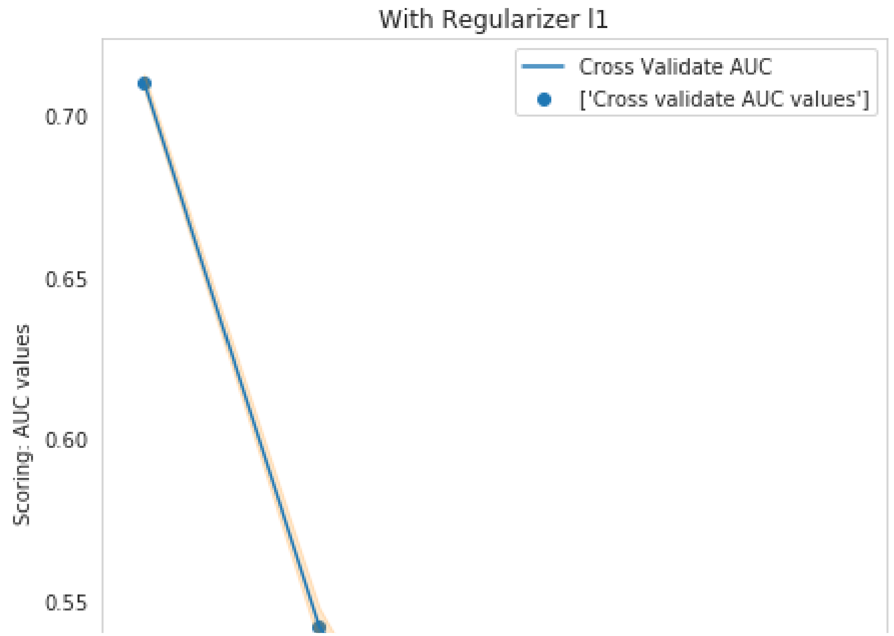



```
Results of analysis using Bag of words vectorized text features merged
with other features using support vector machine classifier:
Optimal Alpha value:  0.01
========================================
Optimal Regularizer:  l2
========================================
AUC value of test data:  0.6614819029050565
========================================
Confusion Matrix :
============================================================
```

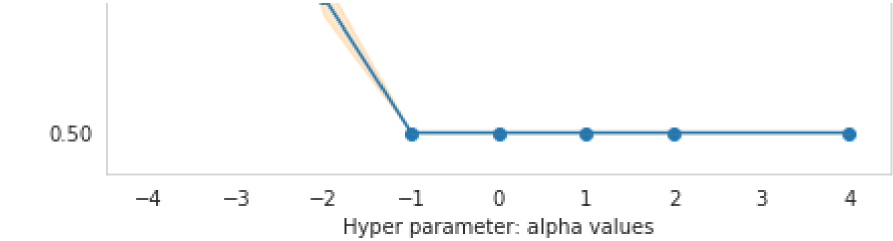


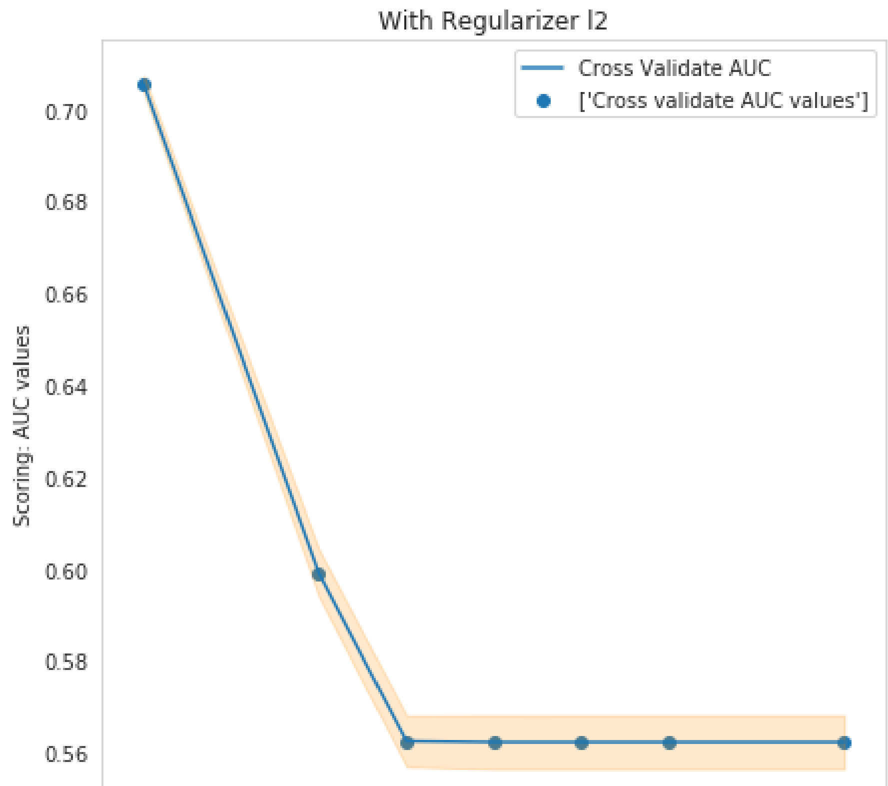


Actual: NO
Actual: YES
7745
20066
Predicted: NO
Predicted: YES
12000
8000
4000

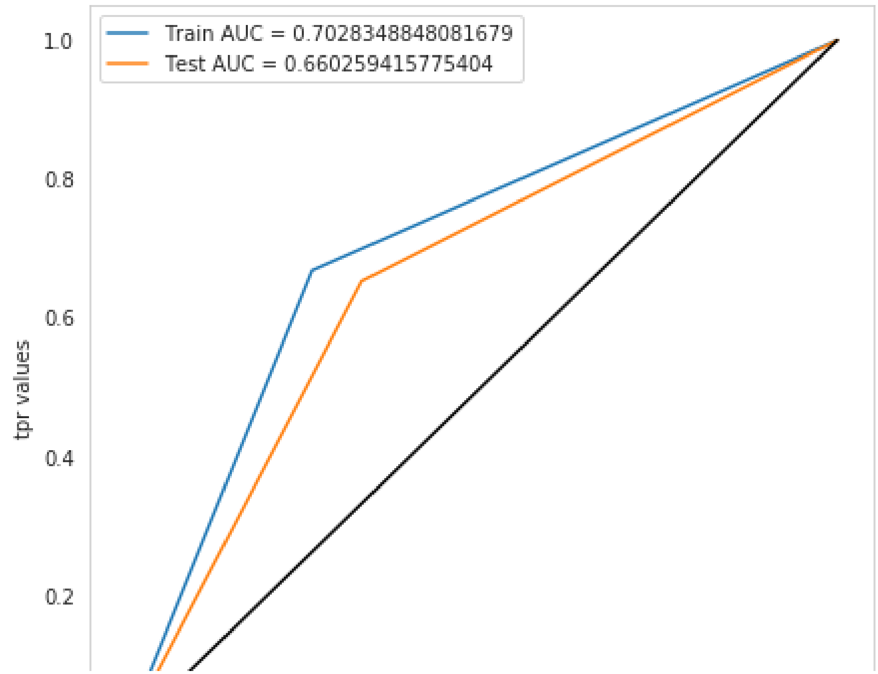


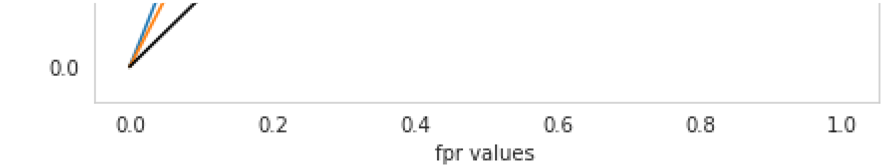
With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
Scoring: AUC values
0.70
0.65
0.60
0.55


0.50
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

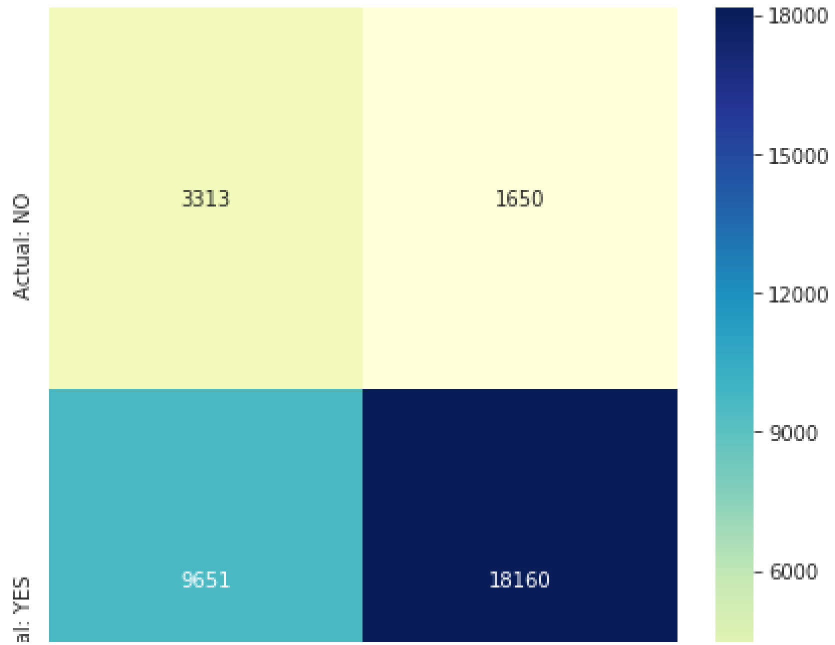

With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
Scoring: AUC values
0.70
0.68
0.66
0.64
0.62
0.60
0.58
0.56


−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

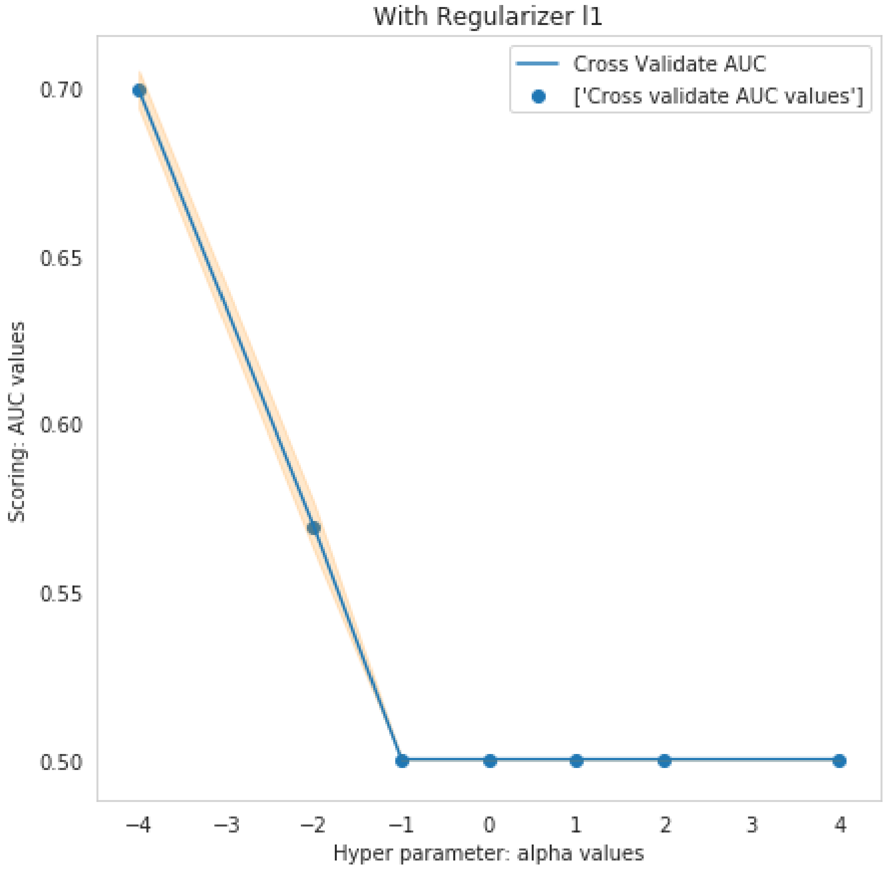

Train AUC = 0.7028348848081679
Test AUC = 0.660259415775404
1.0
0.8
0.6
0.4
0.2
tpr values

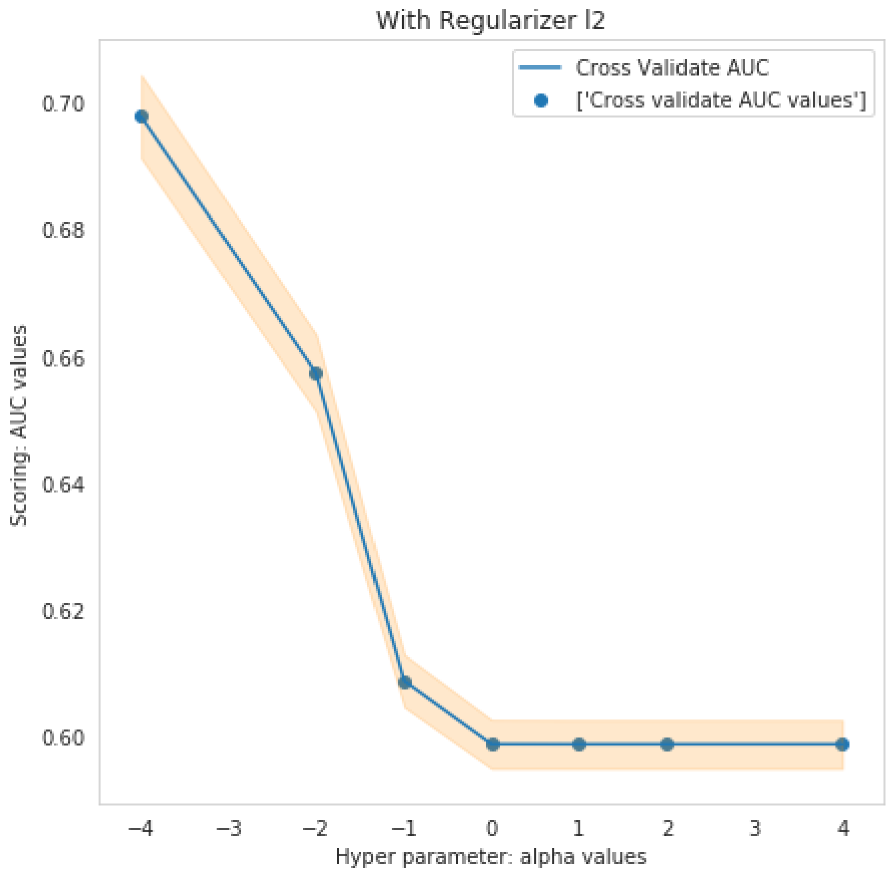



```
Results of analysis using Tf-Idf vectorized text features merged with o
ther features using support vector machine classifier:
Optimal Alpha value:  0.0001
========================================
Optimal Regularizer:  l1
========================================
AUC value of test data:  0.660259415775404
========================================
Confusion Matrix :
============================================================
```

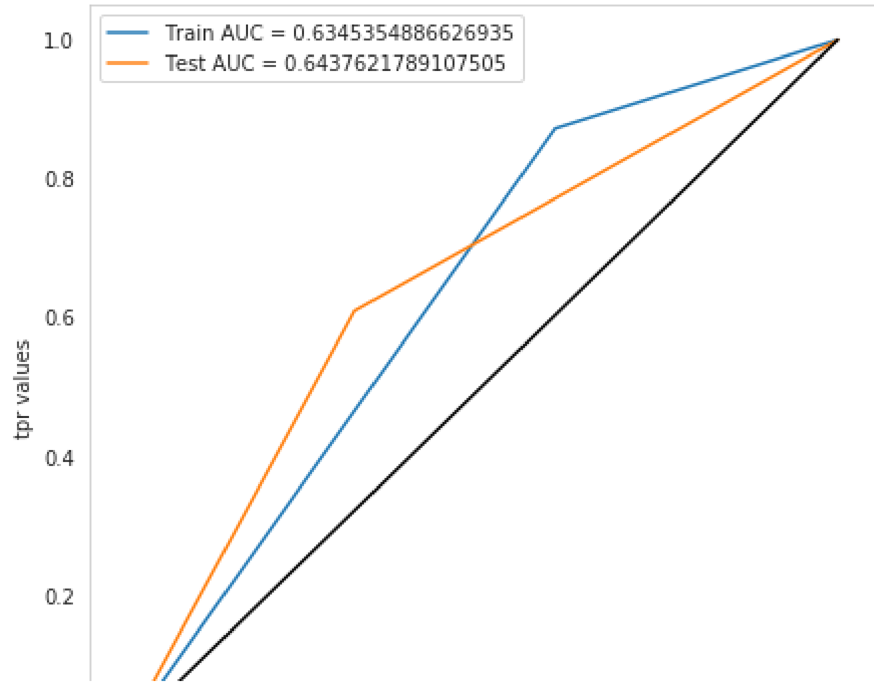



Actu
3000
Predicted: NO
Predicted: YES

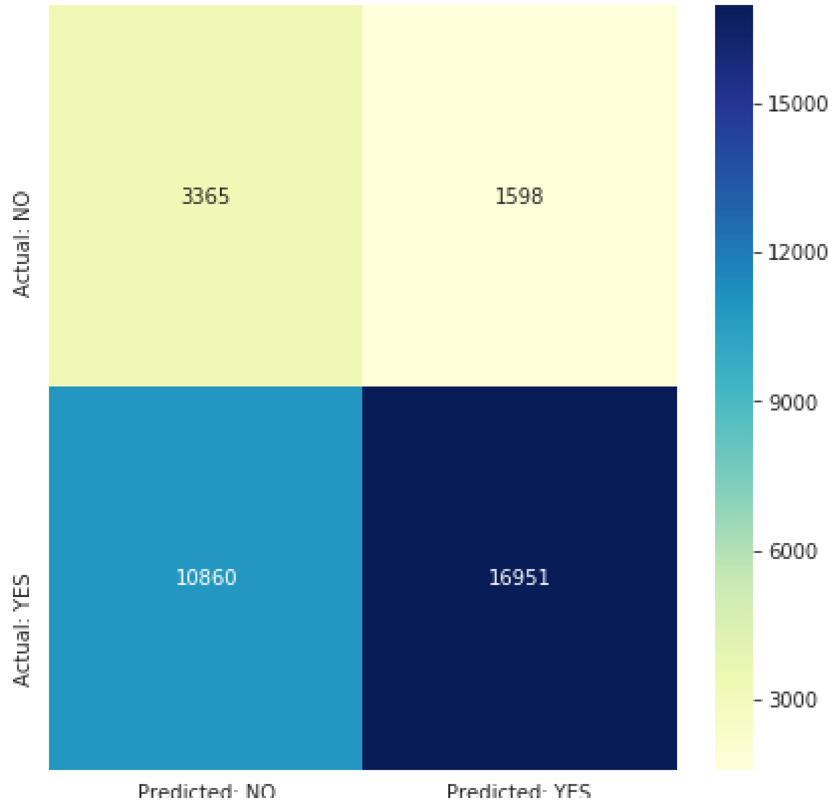

With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
0.70
0.65
0.60
0.55
0.50
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

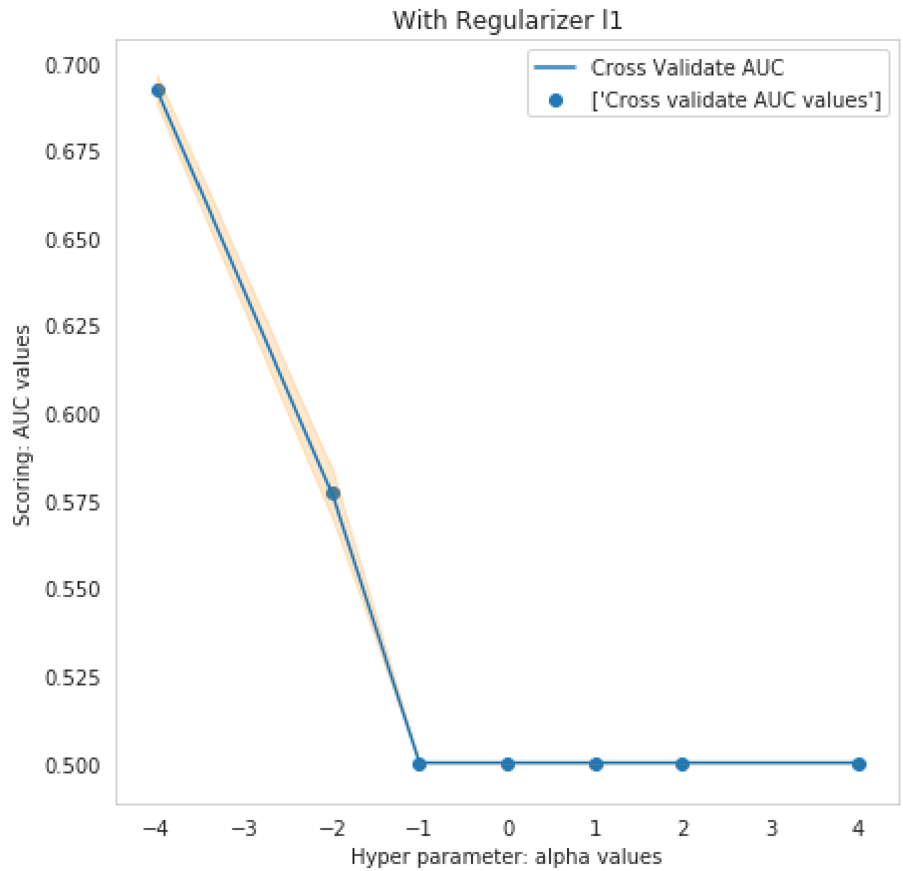


With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
0.70
0.68
0.66
0.64
0.62
0.60
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

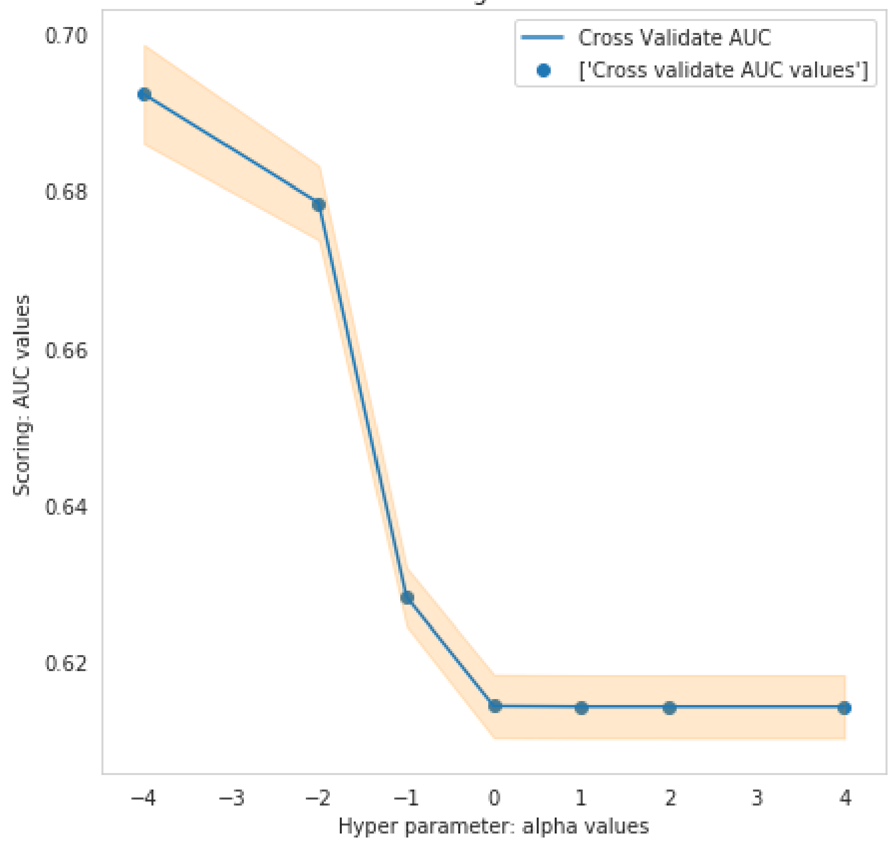

Train AUC = 0.6345354886626935
Test AUC = 0.6437621789107505
1.0
0.8
0.6
0.4
0.2
tpr values

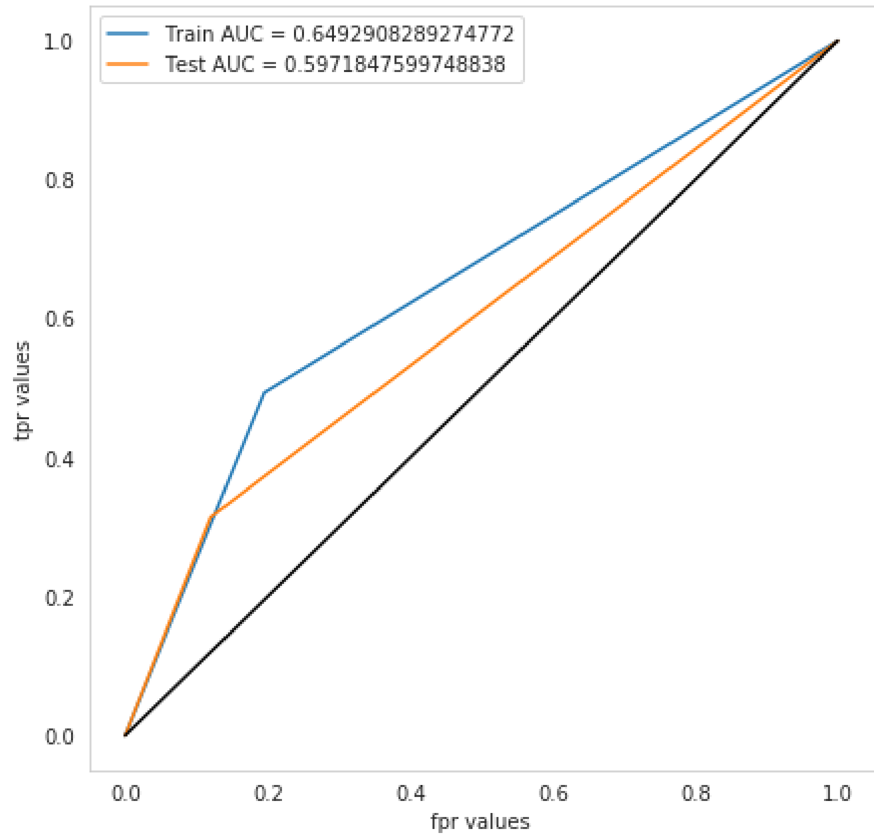

```
Results of analysis using Average Word2Vector vectorized text features
merged with other features using support vector machine classifier:
Optimal Alpha value:  0.0001
========================================
Optimal Regularizer:  l1
========================================
AUC value of test data:  0.6437621789107505
========================================
Confusion Matrix :
============================================================
```




With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

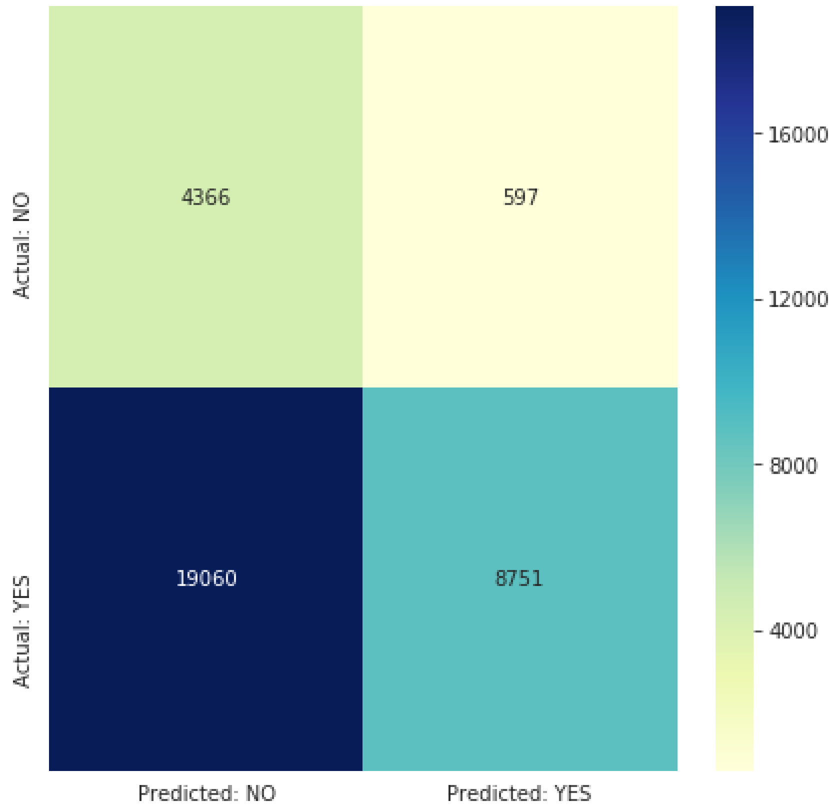
with Regularizer l2






```
Results of analysis using Tf-Idf Weighted Word2Vector vectorized text f
eatures merged with other features using support vector machine classif
ier:
Optimal Alpha value:  0.0001
========================================
```

```
Optimal Regularizer:  l1
========================================
AUC value of test data:  0.5971847599748838
========================================
Confusion Matrix :
============================================================
```



Classification using data with reduced dimensions by support vector machine

In [63]:
```
projectsData.shape
```

Out[63]: (109248, 24)

Calculating number of words of title and essay

In [0]:
```
number_of_words_in_title = [len(title.split()) for title in projectsDat
a['preprocessed_titles'].values]
number_of_words_in_essay = [len(essay.split()) for essay in projectsDat
a['preprocessed_essays'].values]
projectsData['number_of_words_in_title'] = number_of_words_in_title;
projectsData['number_of_words_in_essay'] = number_of_words_in_essay;
```

Calculating sentiment score of each essay

In [74]:
```
sentimentAnalyzer = SentimentIntensityAnalyzer();
positiveSentimentScores = [];
negativeSentimentScores = [];
neutralSentimentScores = [];
compoundSentimentScores = [];
for projectEssay in tqdm(projectsData['preprocessed_essays'].values):
  sentimentScore = sentimentAnalyzer.polarity_scores(projectEssay);
  positiveSentimentScores.append(sentimentScore['pos']);
  negativeSentimentScores.append(sentimentScore['neg']);
  neutralSentimentScores.append(sentimentScore['neu']);
  compoundSentimentScores.append(sentimentScore['compound']);
print(len(positiveSentimentScores), len(negativeSentimentScores), len(n
eutralSentimentScores), len(compoundSentimentScores));
print(positiveSentimentScores[0:5])
```

```
109245 109245 109245 109245
[0.154, 0.305, 0.23, 0.256, 0.151]
```

In [75]:
```
projectsData['positive_sentiment_score'] = positiveSentimentScores;
projectsData['negative_sentiment_score'] = negativeSentimentScores;
projectsData['neutral_sentiment_score'] = neutralSentimentScores;
```

```
projectsData['compound_sentiment_score'] = compoundSentimentScores;
projectsData.shape
```

Out[75]: (109245, 30)

Splitting Data(Only training and test)

In [76]:
```
projectsData = projectsData.dropna(subset = ['teacher_prefix']);
projectsData.shape
```

Out[76]: (109245, 30)

In [77]:
```
classesData = projectsData['project_is_approved']
print(classesData.shape)
```

```
(109245,)
```

In [0]:
```
trainingData, testData, classesTraining, classesTest = model_selection.train_test_split(projectsData, classesData, test_size =  0.3, random_state = 0, stratify = classesData);
trainingData, crossValidateData, classesTraining, classesCrossValidate = model_selection.train_test_split(trainingData, classesTraining, test_size = 0.3, random_state = 0, stratify = classesTraining);
```

In [79]:
```
print("Shapes of splitted data: ");
equalsBorder(70);

print("testData shape: ", testData.shape);
print("classesTest: ", classesTest.shape);
print("trainingData shape: ", trainingData.shape);
print("classesTraining shape: ", classesTraining.shape);
```

```
Shapes of splitted data: 
======================================================================
testData shape:  (32774, 30)
classesTest:  (32774,)
```

```
trainingData shape:  (53529, 30)
classesTraining shape:  (53529,)
```

In [80]:
```
print("Number of negative points: ", trainingData[trainingData['project_is_approved'] == 0].shape);
print("Number of positive points: ", trainingData[trainingData['project_is_approved'] == 1].shape);
```

```
Number of negative points:  (8105, 30)
Number of positive points:  (45424, 30)
```

In [0]:
```
vectorizedFeatureNames = [];
```

Vectorizing categorical data

1. Vectorizing cleaned_categories(project_subject_categories cleaned) - One Hot Encoding

In [0]:
```
# Using CountVectorizer for performing one-hot-encoding by setting vocabulary as list of all unique cleaned_categories
subjectsCategoriesVectorizer = CountVectorizer(vocabulary = list(sortedCategoriesDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with cleaned_categories values
subjectsCategoriesVectorizer.fit(trainingData['cleaned_categories'].values);
# Vectorizing categories using one-hot-encoding
categoriesVectors = subjectsCategoriesVectorizer.transform(trainingData['cleaned_categories'].values);
```

In [83]:
```
print("Features used in vectorizing categories: ");
equalsBorder(70);
print(subjectsCategoriesVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of cleaned_categories matrix after vectorization(one-hot-encoding): ", categoriesVectors.shape);
```

```
equalsBorder(70);
print("Sample vectors of categories: ");
equalsBorder(70);
print(categoriesVectors[0:4])
```

```
Features used in vectorizing categories:
======================================================================
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearn
ing', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Langua
ge']
======================================================================
Shape of cleaned_categories matrix after vectorization(one-hot-encodin
g):  (53529, 9)
======================================================================
Sample vectors of categories:
======================================================================
  (0, 3)	1
  (0, 7)	1
  (1, 7)	1
  (1, 8)	1
  (2, 6)	1
  (2, 7)	1
  (3, 7)	1
```

2. Vectorizing cleaned_sub_categories(project_subject_sub_categories cleaned) - One Hot Encoding

In [0]:

```
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique cleaned_sub_categories
subjectsSubCategoriesVectorizer = CountVectorizer(vocabulary = list(sor
tedDictionarySubCategories.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with cleaned_sub_categories values
subjectsSubCategoriesVectorizer.fit(trainingData['cleaned_sub_categorie
s'].values);
# Vectorizing sub categories using one-hot-encoding
subCategoriesVectors = subjectsSubCategoriesVectorizer.transform(traini
ngData['cleaned_sub_categories'].values);
```

In [85]:
```
print("Features used in vectorizing subject sub categories: ");
equalsBorder(70);
print(subjectsSubCategoriesVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of cleaned_categories matrix after vectorization(one-hot-e
ncoding): ", subCategoriesVectors.shape);
equalsBorder(70);
print("Sample vectors of categories: ");
equalsBorder(70);
print(subCategoriesVectors[0:4])
```

```
Features used in vectorizing subject sub categories:
======================================================================
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolveme
nt', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutri
tionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingA
rts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPre
p', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopme
nt', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Healt
h_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing',
'Mathematics', 'Literacy']
======================================================================
Shape of cleaned_categories matrix after vectorization(one-hot-encodin
g):  (53529, 30)
======================================================================
Sample vectors of categories: 
======================================================================
  (0, 23)	1
  (0, 25)	1
  (1, 28)	1
  (1, 29)	1
  (2, 24)	1
  (2, 25)	1
  (3, 28)	1
```

3. Vectorizing teacher_prefix - One Hot Encoding

In [0]:
```
def giveCounter(data):
    counter = Counter();
    for dataValue in data:
        counter.update(str(dataValue).split());
    return counter
```

In [87]:
```
giveCounter(trainingData['teacher_prefix'].values)
```

Out[87]:
```
Counter({'Dr.': 4, 'Mr.': 5206, 'Mrs.': 28216, 'Ms.': 18934, 'Teacher':
1169})
```

In [0]:
```
teacherPrefixDictionary = dict(giveCounter(trainingData['teacher_prefi
x'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique teacher_prefix
teacherPrefixVectorizer = CountVectorizer(vocabulary = list(teacherPref
ixDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with teacher_prefix values
teacherPrefixVectorizer.fit(trainingData['teacher_prefix'].values);
# Vectorizing teacher_prefix using one-hot-encoding
teacherPrefixVectors = teacherPrefixVectorizer.transform(trainingData[
'teacher_prefix'].values);
```

In [89]:
```
print("Features used in vectorizing teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of teacher_prefix matrix after vectorization(one-hot-encod
ing): ", teacherPrefixVectors.shape);
equalsBorder(70);
print("Sample vectors of teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectors[0:100]);
```

```
Features used in vectorizing teacher_prefix:
======================================================================
['Ms.', 'Mrs.', 'Teacher', 'Mr.', 'Dr.']
======================================================================
Shape of teacher_prefix matrix after vectorization(one-hot-encoding):
```

```
(53529, 5)
==============================================================================
Sample vectors of teacher_prefix:
==============================================================================
  (21, 2)	1
```

In [90]:
```
teacherPrefixes = [prefix.replace('.', '') for prefix in trainingData[
'teacher_prefix'].values];
teacherPrefixes[0:5]
```

Out[90]:
```
['Ms', 'Ms', 'Mrs', 'Mrs', 'Mrs']
```

In [91]:
```
trainingData['teacher_prefix'] = teacherPrefixes;
trainingData.head(3)
```

Out[91]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
66637	174395	p233512	c9e73f31af5ad4c7d3a140e81554da3b	Ms	GA
76424	11981	p088047	e1aa00913e0009364b5c7c3c4ab9a6f5	Ms	WA
34433	11994	p210041	a6c5d41f4e18aca1530159f7cee84084	Mrs	NC

In [0]:
```
teacherPrefixDictionary = dict(giveCounter(trainingData['teacher_prefi
x'].values));
```

```
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique teacher_prefix
teacherPrefixVectorizer = CountVectorizer(vocabulary = list(teacherPref
ixDictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with teacher_prefix values
teacherPrefixVectorizer.fit(trainingData['teacher_prefix'].values);
# Vectorizing teacher_prefix using one-hot-encoding
teacherPrefixVectors = teacherPrefixVectorizer.transform(trainingData[
'teacher_prefix'].values);
```

In [93]:

```
print("Features used in vectorizing teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of teacher_prefix matrix after vectorization(one-hot-encod
ing): ", teacherPrefixVectors.shape);
equalsBorder(70);
print("Sample vectors of teacher_prefix: ");
equalsBorder(70);
print(teacherPrefixVectors[0:4]);
```

```
Features used in vectorizing teacher_prefix:
======================================================================
['Ms', 'Mrs', 'Teacher', 'Mr', 'Dr']
======================================================================
Shape of teacher_prefix matrix after vectorization(one-hot-encoding):
(53529, 5)
======================================================================
Sample vectors of teacher_prefix:
======================================================================
  (0, 0)	1
  (1, 0)	1
  (2, 1)	1
  (3, 1)	1
```

4. Vectorizing school_state - One Hot Encoding

In [0]:
```
schoolStateDictionary = dict(giveCounter(trainingData['school_state'].v
alues));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique school states
schoolStateVectorizer = CountVectorizer(vocabulary = list(schoolStateDi
ctionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with school_state values
schoolStateVectorizer.fit(trainingData['school_state'].values);
# Vectorizing school_state using one-hot-encoding
schoolStateVectors = schoolStateVectorizer.transform(trainingData['scho
ol_state'].values);
```

In [95]:
```
print("Features used in vectorizing school_state: ");
equalsBorder(70);
print(schoolStateVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of school_state matrix after vectorization(one-hot-encodin
g): ", schoolStateVectors.shape);
equalsBorder(70);
print("Sample vectors of school_state: ");
equalsBorder(70);
print(schoolStateVectors[0:4]);
```

```
Features used in vectorizing school_state:
======================================================================
['GA', 'WA', 'NC', 'MI', 'NV', 'KY', 'CA', 'CT', 'PA', 'SC', 'WV', 'C
O', 'FL', 'AZ', 'MS', 'OH', 'LA', 'TX', 'NY', 'IN', 'MO', 'KS', 'IA',
'NJ', 'AR', 'MA', 'WI', 'OK', 'UT', 'MN', 'OR', 'DC', 'VA', 'AL', 'NM',
'TN', 'IL', 'HI', 'DE', 'MD', 'ID', 'SD', 'NH', 'NE', 'ME', 'MT', 'AK',
'ND', 'VT', 'WY', 'RI']
======================================================================
Shape of school_state matrix after vectorization(one-hot-encoding):  (5
3529, 51)
======================================================================
Sample vectors of school_state:
======================================================================
  (0, 0)        1
  (1, 1)        1
```

```
(2, 2)        1
(3, 3)        1
```

5. Vectorizing project_grade_category - One Hot Encoding

In [96]:
```
giveCounter(trainingData['project_grade_category'])
```

Out[96]:
```
Counter({'3-5': 18193,
         '6-8': 8300,
         '9-12': 5289,
         'Grades': 53529,
         'PreK-2': 21747})
```

In [97]:
```
cleanedGrades = []
for grade in trainingData['project_grade_category'].values:
    grade = grade.replace(' ', '');
    grade = grade.replace('-', 'to');
    cleanedGrades.append(grade);
cleanedGrades[0:4]
```

Out[97]:
```
['Grades3to5', 'GradesPreKto2', 'Grades3to5', 'Grades3to5']
```

In [98]:
```
trainingData['project_grade_category'] = cleanedGrades
trainingData.head(4)
```

Out[98]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
66637	174395	p233512	c9e73f31af5ad4c7d3a140e81554da3b	Ms	GA

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
76424	11981	p088047	e1aa00913e0009364b5c7c3c4ab9a6f5	Ms	WA
34433	11994	p210041	a6c5d41f4e18aca1530159f7cee84084	Mrs	NC
84559	145506	p030629	8c9462aaf17c6a5869fe54c62af8b23c	Mrs	MI

In [0]:

```
projectGradeDictionary = dict(giveCounter(trainingData['project_grade_c
ategory'].values));
# Using CountVectorizer for performing one-hot-encoding by setting voca
bulary as list of all unique project grade categories
projectGradeVectorizer = CountVectorizer(vocabulary = list(projectGrade
Dictionary.keys()), lowercase = False, binary = True);
# Fitting CountVectorizer with project_grade_category values
projectGradeVectorizer.fit(trainingData['project_grade_category'].value
s);
# Vectorizing project_grade_category using one-hot-encoding
projectGradeVectors = projectGradeVectorizer.transform(trainingData['pr
oject_grade_category'].values);
```

In [100]:

```
print("Features used in vectorizing project_grade_category: ");
equalsBorder(70);
print(projectGradeVectorizer.get_feature_names());
equalsBorder(70);
print("Shape of school_state matrix after vectorization(one-hot-encodin
```

```
g): ", projectGradeVectors.shape);
equalsBorder(70);
print("Sample vectors of school_state: ");
equalsBorder(70);
print(projectGradeVectors[0:4]);
```

```
Features used in vectorizing project_grade_category:
======================================================================
['Grades3to5', 'GradesPreKto2', 'Grades6to8', 'Grades9to12']
======================================================================
Shape of school_state matrix after vectorization(one-hot-encoding):  (5
3529, 4)
======================================================================
Sample vectors of school_state:
======================================================================
  (0, 0)	1
  (1, 1)	1
  (2, 0)	1
  (3, 0)	1
```

Tf-Idf Vectorization

1. Vectorizing project_essay

In [0]:

```
# Intializing tfidf vectorizer for tf-idf vectorization of preprocessed
 project essays
tfIdfEssayVectorizer = TfidfVectorizer(min_df = 10, max_features = 5000
);
# Transforming the preprocessed project essays to tf-idf vectors
tfIdfEssayModel = tfIdfEssayVectorizer.fit_transform(preProcessedEssays
WithoutStopWords);
```

In [102]:

```
print("Some of the Features used in tf-idf vectorizing preprocessed ess
ays: ");
equalsBorder(70);
print(tfIdfEssayVectorizer.get_feature_names()[-40:]);
```

```
equalsBorder(70);
print("Shape of preprocessed title matrix after tf-idf vectorization: "
, tfIdfEssayModel.shape);
equalsBorder(70);
print("Sample Tf-Idf vector of preprocessed essay: ");
equalsBorder(70);
print(tfIdfEssayModel[0])
```

```
Some of the Features used in tf-idf vectorizing preprocessed essays:
======================================================================
['worrying', 'worst', 'worth', 'worthwhile', 'worthy', 'would', 'wow',
'write', 'writer', 'writers', 'writing', 'writings', 'written', 'wron
g', 'wrote', 'xylophones', 'yard', 'year', 'yearbook', 'yearly', 'year
n', 'yearning', 'years', 'yes', 'yesterday', 'yet', 'yoga', 'york', 'yo
ung', 'younger', 'youngest', 'youngsters', 'youth', 'youtube', 'zero',
'zest', 'zip', 'zone', 'zones', 'zoo']
======================================================================
Shape of preprocessed title matrix after tf-idf vectorization:  (10924
8, 5000)
======================================================================
Sample Tf-Idf vector of preprocessed essay:
======================================================================
  (0, 3013)     0.015965240695453155
  (0, 1488)     0.10227077629951559
  (0, 900)      0.026463005286219803
  (0, 4982)     0.04582647393654424
  (0, 3375)     0.0625444219876457
  (0, 4977)     0.0306752753684296
  (0, 4752)     0.05440679396599839
  (0, 780)      0.07662037632839458
  (0, 3169)     0.03781481331251044
  (0, 3390)     0.18321040329163196
  (0, 108)      0.040000071711429254
  (0, 3091)     0.022124698215231303
  (0, 1433)     0.061920444892322624
  (0, 1246)     0.05054577035223512
  (0, 4851)     0.06986541769014476
  (0, 3730)     0.09018028989080147
  (0, 4021)     0.09616413965528452
  (0, 4479)     0.03754788787263396
```

```
  (0, 805)	0.07832945191677794
  (0, 4224)	0.10949652575968567
  (0, 4795)	0.20453745972832738
  (0, 1468)	0.10442911392080413
  (0, 3256)	0.045907362380733514
  (0, 3137)	0.08665307470908791
  (0, 4290)	0.0668235584225867
  :	:
  (0, 2819)	0.18267578272530896
  (0, 2657)	0.08246396864201584
  (0, 1645)	0.03290277997628051
  (0, 3540)	0.09266878292993941
  (0, 2629)	0.23032992958601448
  (0, 3788)	0.18439470661000118
  (0, 31)	0.08003880477371976
  (0, 3958)	0.03546008600799206
  (0, 2591)	0.1200129277454165
  (0, 2032)	0.08302595052382315
  (0, 622)	0.0797021927845829
  (0, 262)	0.09317692266510744
  (0, 579)	0.08993661029788748
  (0, 3019)	0.07694246325409376
  (0, 2316)	0.09044713807559518
  (0, 3723)	0.09806756817480866
  (0, 3439)	0.09819479507450754
  (0, 2861)	0.09995313270720342
  (0, 2592)	0.13131596546189067
  (0, 4546)	0.058915439503183835
  (0, 3986)	0.04932114628372647
  (0, 4944)	0.03804356418624494
  (0, 2630)	0.03600810586474103
  (0, 1564)	0.29767755886622843
  (0, 4364)	0.07692419628496143
```

Vectorizing numerical features

1. Vectorizing price

In [0]:
```
# Standardizing the price data using StandardScaler(Uses mean and std f
or standardization)
priceScaler = MinMaxScaler();
priceScaler.fit(trainingData['price'].values.reshape(-1, 1));
priceStandardized = priceScaler.transform(trainingData['price'].values.
reshape(-1, 1));
```

In [104]:
```
print("Shape of standardized matrix of prices: ", priceStandardized.sha
pe);
equalsBorder(70);
print("Sample original prices: ");
equalsBorder(70);
print(trainingData['price'].values[0:5]);
print("Sample standardized prices: ");
equalsBorder(70);
print(priceStandardized[0:5]);
```

```
Shape of standardized matrix of prices:  (53529, 1)
======================================================================
Sample original prices: 
======================================================================
[159.   509.85 289.92 190.24 438.99]
Sample standardized prices: 
======================================================================
[[0.01583663]
 [0.05092745]
 [0.0289308 ]
 [0.01896115]
 [0.04384028]]
```

2. Vectorizing quantity

In [0]:
```
# Standardizing the quantity data using StandardScaler(Uses mean and st
d for standardization)
quantityScaler = MinMaxScaler();
```

```
quantityScaler.fit(trainingData['quantity'].values.reshape(-1, 1));
quantityStandardized = quantityScaler.transform(trainingData['quantity'
].values.reshape(-1, 1));
```

In [106]:
```
print("Shape of standardized matrix of quantities: ", quantityStandardi
zed.shape);
equalsBorder(70);
print("Sample original quantities: ");
equalsBorder(70);
print(trainingData['quantity'].values[0:5]);
print("Sample standardized quantities: ");
equalsBorder(70);
print(quantityStandardized[0:5]);
```

```
Shape of standardized matrix of quantities:  (53529, 1)
======================================================================
Sample original quantities: 
======================================================================
[ 4  1 12 17  2]
Sample standardized quantities: 
======================================================================
[[0.00322928]
 [0.        ]
 [0.01184069]
 [0.01722282]
 [0.00107643]]
```

3. Vectorizing teacher_number_of_previously_posted_projects

In [0]:
```
# Standardizing the teacher_number_of_previously_posted_projects data u
sing StandardScaler(Uses mean and std for standardization)
previouslyPostedScaler = MinMaxScaler();
previouslyPostedScaler.fit(trainingData['teacher_number_of_previously_p
osted_projects'].values.reshape(-1, 1));
previouslyPostedStandardized = previouslyPostedScaler.transform(trainin
gData['teacher_number_of_previously_posted_projects'].values.reshape(-1
, 1));
```

In [108]:
```
print("Shape of standardized matrix of teacher_number_of_previously_pos
ted_projects: ", previouslyPostedStandardized.shape);
equalsBorder(70);
print("Sample original number of previously posted projects: ");
equalsBorder(70);
print(trainingData['teacher_number_of_previously_posted_projects'].valu
es[0:5]);
print("Sample standardized teacher_number_of_previously_posted_project
s: ");
equalsBorder(70);
print(previouslyPostedStandardized[0:5]);
```

```
Shape of standardized matrix of teacher_number_of_previously_posted_pro
jects:  (53529, 1)
======================================================================
Sample original number of previously posted projects:
======================================================================
[ 1  9 25  0  0]
Sample standardized teacher_number_of_previously_posted_projects:
======================================================================
[[0.00221729]
 [0.01995565]
 [0.05543237]
 [0.        ]
 [0.        ]]
```

4. Vectorizing number_of_words_in_title

In [0]:
```
numberOfWordsInTitleScaler = MinMaxScaler();
numberOfWordsInTitleScaler.fit(trainingData['number_of_words_in_title']
.values.reshape(-1, 1));
numberOfWordsInTitleStandardized = numberOfWordsInTitleScaler.transform
(trainingData['number_of_words_in_title'].values.reshape(-1, 1));
```

In [110]:
```
print("Shape of standardized matrix of number_of_words_in_title: ", num
berOfWordsInTitleStandardized.shape);
equalsBorder(70);
```

```
print("Sample original number of words in title: ");
equalsBorder(70);
print(trainingData['number_of_words_in_title'].values[0:5]);
print("Sample standardized number_of_words_in_title: ");
equalsBorder(70);
print(numberOfWordsInTitleStandardized[0:5]);
```

```
Shape of standardized matrix of number_of_words_in_title:  (53529, 1)
======================================================================
Sample original number of words in title:
======================================================================
[2 4 2 4 3]
Sample standardized number_of_words_in_title:
======================================================================
[[0.18181818]
 [0.36363636]
 [0.18181818]
 [0.36363636]
 [0.27272727]]
```

5. Vectorizing number_of_words_in_essay

In [0]:

```
numberOfWordsInEssayScaler = MinMaxScaler();
numberOfWordsInEssayScaler.fit(trainingData['number_of_words_in_essay']
.values.reshape(-1, 1));
numberOfWordsInEssayStandardized = numberOfWordsInEssayScaler.transform
(trainingData['number_of_words_in_essay'].values.reshape(-1, 1));
```

In [112]:

```
print("Shape of standardized matrix of number_of_words_in_essay: ", num
berOfWordsInEssayStandardized.shape);
equalsBorder(70);
print("Sample original number of words in essay: ");
equalsBorder(70);
print(trainingData['number_of_words_in_essay'].values[0:5]);
print("Sample standardized number_of_words_in_essay: ");
equalsBorder(70);
print(numberOfWordsInEssayStandardized[0:5]);
```

```
Shape of standardized matrix of number_of_words_in_essay:  (53529, 1)
```

```
Shape of standardized matrix of number_of_words_in_essay:  (53529, 1)
==========================================================================
Sample original number of words in essay:
==========================================================================
[167 139 210  97 117]
Sample standardized number_of_words_in_essay:
==========================================================================
[[0.39917695]
 [0.28395062]
 [0.57613169]
 [0.11111111]
 [0.19341564]]
```

In [0]:

```
numberOfPoints = previouslyPostedStandardized.shape[0];
# Categorical data
categoriesVectorsSub = categoriesVectors[0:numberOfPoints];
subCategoriesVectorsSub = subCategoriesVectors[0:numberOfPoints];
teacherPrefixVectorsSub = teacherPrefixVectors[0:numberOfPoints];
schoolStateVectorsSub = schoolStateVectors[0:numberOfPoints];
projectGradeVectorsSub = projectGradeVectors[0:numberOfPoints];

# Text data
tfIdfEssayModelSub = tfIdfEssayModel[0:numberOfPoints];

# Numerical data
priceStandardizedSub = priceStandardized[0:numberOfPoints];
quantityStandardizedSub = quantityStandardized[0:numberOfPoints];
previouslyPostedStandardizedSub = previouslyPostedStandardized[0:number
OfPoints];
numberOfWordsInTitleStandardizedSub = numberOfWordsInTitleStandardized[
0:numberOfPoints];
numberOfWordsInEssayStandardizedSub = numberOfWordsInEssayStandardized[
0:numberOfPoints];
positiveSentimentScoreSub = trainingData['positive_sentiment_score'].va
lues[0:numberOfPoints].reshape(-1, 1);
negativeSentimentScoreSub = trainingData['negative_sentiment_score'].va
lues[0:numberOfPoints].reshape(-1, 1);
neutralSentimentScoreSub = trainingData['neutral_sentiment_score'].valu
es[0:numberOfPoints].reshape(-1, 1);
```

```
compoundSentimentScoreSub = trainingData['compound_sentiment_score'].va
lues[0:numberOfPoints].reshape(-1, 1);

# Classes
classesTrainingSub = classesTraining;
```

In [5]:

```
supportVectorMachineResultsDataFrame = pd.DataFrame(columns =  ['Vector
izer', 'Model', 'Hyper Parameter - alpha', 'AUC', 'Data']);
supportVectorMachineResultsDataFrame
```

Out[5]:

	Vectorizer	Model	Hyper Parameter - alpha	AUC	Data

Preparing cross validate data for analysis

In [115]:

```
# Test data categorical features transformation
categoriesTransformedCrossValidateData = subjectsCategoriesVectorizer.t
ransform(crossValidateData['cleaned_categories']);
subCategoriesTransformedCrossValidateData = subjectsSubCategoriesVector
izer.transform(crossValidateData['cleaned_sub_categories']);
teacherPrefixTransformedCrossValidateData = teacherPrefixVectorizer.tra
nsform(crossValidateData['teacher_prefix']);
schoolStateTransformedCrossValidateData = schoolStateVectorizer.transfo
rm(crossValidateData['school_state']);
projectGradeTransformedCrossValidateData = projectGradeVectorizer.trans
form(crossValidateData['project_grade_category']);

# Test data text features transformation
preProcessedEssaysTemp = preProcessingWithAndWithoutStopWords(crossVali
dateData['project_essay'])[1];
tfIdfEssayTransformedCrossValidateData = tfIdfEssayVectorizer.transform
(preProcessedEssaysTemp);

# Test data numerical features transformation
priceTransformedCrossValidateData = priceScaler.transform(crossValidate
Data['price'].values.reshape(-1, 1));
quantityTransformedCrossValidateData = quantityScaler.transform(crossVa
```

```
lidateData['quantity'].values.reshape(-1, 1));
previouslyPostedTransformedCrossValidateData = previouslyPostedScaler.t
ransform(crossValidateData['teacher_number_of_previously_posted_project
s'].values.reshape(-1, 1));
numberOfWordsInTitleTransformedCrossValidateData = numberOfWordsInTitle
Scaler.transform(crossValidateData['number_of_words_in_title'].values.r
eshape(-1, 1));
numberOfWordsInEssayTransformedCrossValidateData = numberOfWordsInEssay
Scaler.transform(crossValidateData['number_of_words_in_essay'].values.r
eshape(-1, 1));
positiveSentimentScoreCrossValidateData = crossValidateData['positive_s
entiment_score'].values.reshape(-1, 1);
negativeSentimentScoreCrossValidateData = crossValidateData['negative_s
entiment_score'].values.reshape(-1, 1);
neutralSentimentScoreCrossValidateData = crossValidateData['neutral_sen
timent_score'].values.reshape(-1, 1);
compoundSentimentScoreCrossValidateData = crossValidateData['compound_s
entiment_score'].values.reshape(-1, 1);
```

Preparing Test data for analysis

In [116]:

```
# Test data categorical features transformation
categoriesTransformedTestData = subjectsCategoriesVectorizer.transform(
testData['cleaned_categories']);
subCategoriesTransformedTestData = subjectsSubCategoriesVectorizer.tran
sform(testData['cleaned_sub_categories']);
teacherPrefixTransformedTestData = teacherPrefixVectorizer.transform(te
stData['teacher_prefix']);
schoolStateTransformedTestData = schoolStateVectorizer.transform(testDa
ta['school_state']);
projectGradeTransformedTestData = projectGradeVectorizer.transform(test
Data['project_grade_category']);

# Test data text features transformation
preProcessedEssaysTemp = preProcessingWithAndWithoutStopWords(testData[
'project_essay'])[1];
```

```
tfIdfEssayTransformedTestData = tfIdfEssayVectorizer.transform(preProce
ssedEssaysTemp);


# Test data numerical features transformation
priceTransformedTestData = priceScaler.transform(testData['price'].valu
es.reshape(-1, 1));
quantityTransformedTestData = quantityScaler.transform(testData['quanti
ty'].values.reshape(-1, 1));
previouslyPostedTransformedTestData = previouslyPostedScaler.transform(
testData['teacher_number_of_previously_posted_projects'].values.reshape
(-1, 1));
numberOfWordsInTitleTransformedTestData = numberOfWordsInTitleScaler.tr
ansform(testData['number_of_words_in_title'].values.reshape(-1, 1));
numberOfWordsInEssayTransformedTestData = numberOfWordsInEssayScaler.tr
ansform(testData['number_of_words_in_essay'].values.reshape(-1, 1));
positiveSentimentScoreTestData = testData['positive_sentiment_score'].v
alues.reshape(-1, 1);
negativeSentimentScoreTestData = testData['negative_sentiment_score'].v
alues.reshape(-1, 1);
neutralSentimentScoreTestData = testData['neutral_sentiment_score'].val
ues.reshape(-1, 1);
compoundSentimentScoreTestData = testData['compound_sentiment_score'].v
alues.reshape(-1, 1);
```

Finding appropriate dimensions(less) using elbow method

In [0]:

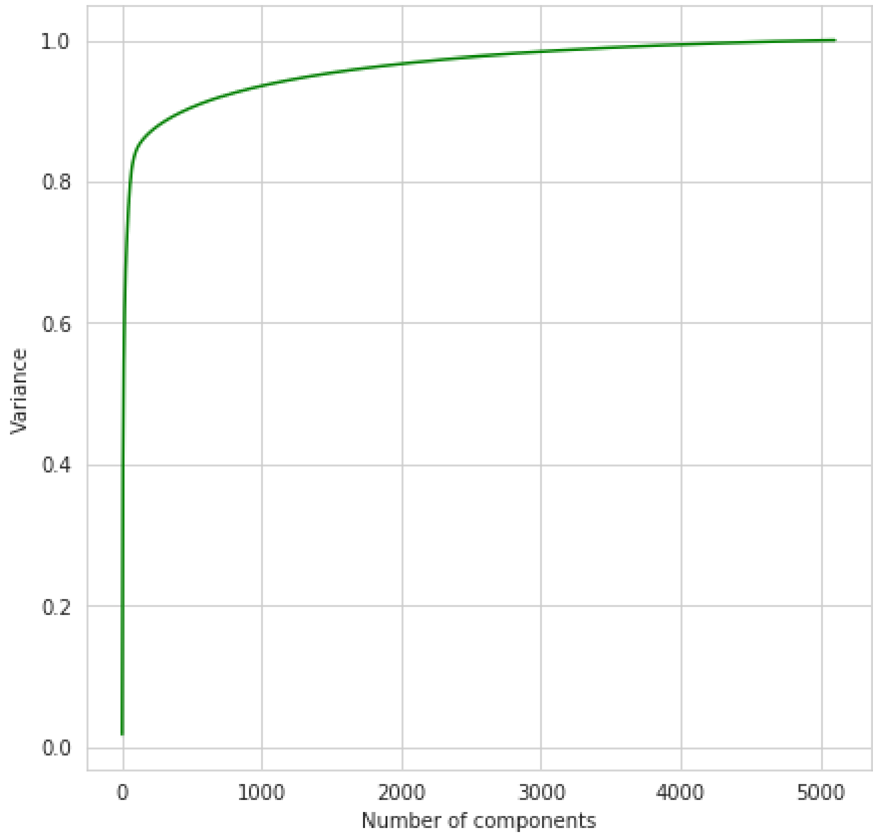
```
trainingMergedData = hstack((categoriesVectorsSub,\
                             subCategoriesVectorsSub,\
                             teacherPrefixVectorsSub,\
                             schoolStateVectorsSub,\
                             projectGradeVectorsSub,\
                             priceStandardizedSub,\
                             previouslyPostedStandardizedSub,\
                             numberOfWordsInTitleStandardizedSub,\
                             numberOfWordsInEssayStandardizedSub,\
```

```
                                     positiveSentimentScoreSub,\
                                     negativeSentimentScoreSub,\
                                     neutralSentimentScoreSub,\
                                     compoundSentimentScoreSub, \
                                    tfIdfEssayModelSub));
svd = TruncatedSVD(n_components = trainingMergedData.shape[1] - 1, rand
om_state = 42);
svd.fit(trainingMergedData);

componentsRatio = svd.explained_variance_ratio_;
```

In [118]:

```
components = np.arange(1, trainingMergedData.shape[1]);
componentsRatio =svd.explained_variance_ratio_.cumsum()

print(componentsRatio);

plt.xlabel('Number of components');
plt.ylabel('Variance');
plt.plot(components, componentsRatio, color = 'green');
```

```
[0.01792355 0.10202375 0.17900866 ... 1.         1.         1.        ]
```



Observations:

1. As you can see from above plot that with dimensions more than 450 90% of variance is retained. So the less number of dimensions to start with inorder to get good results would be 450.
2. At dimensions more than 1400 more than 95% of variance is retained.

Classification using data with reduced dimensions by support vector machine

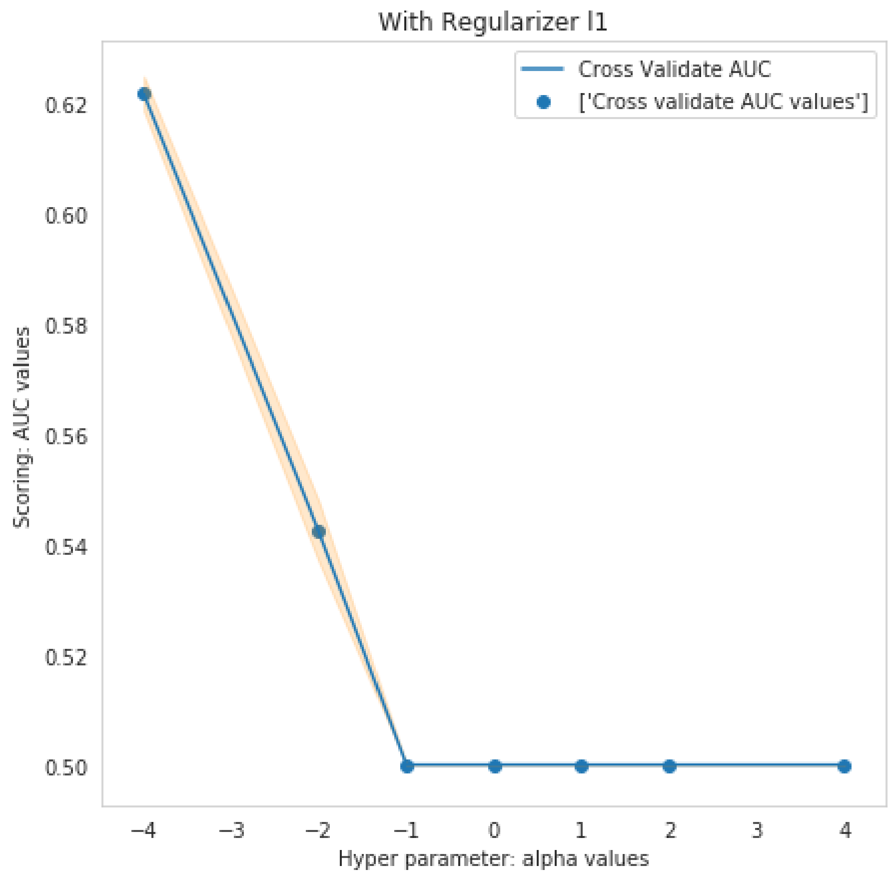
```
In [120]: techniques = ['With Reduced Dimensions'];
n_componentsValues = [450, 600, 900, 1200, 1400];
for index, technique in enumerate(techniques):
  for n_components in n_componentsValues:
    trainingMergedData = hstack((categoriesVectorsSub,\
                                 subCategoriesVectorsSub,\
                                 teacherPrefixVectorsSub,\
                                 schoolStateVectorsSub,\
                                 projectGradeVectorsSub,\
                                 priceStandardizedSub,\
                                 previouslyPostedStandardizedSub,\
                                 numberOfWordsInTitleStandardizedSub,\
                                 numberOfWordsInEssayStandardizedSub,\
                                 positiveSentimentScoreSub,\
                                 negativeSentimentScoreSub,\
                                 neutralSentimentScoreSub,\
                                 compoundSentimentScoreSub,\
                                 tfIdfEssayModelSub));
    svd = TruncatedSVD(n_components = n_components, random_state = 42);
    svd.fit(trainingMergedData);
    trainingMergedData = svd.transform(trainingMergedData);
    crossValidateMergedData = hstack((categoriesTransformedCrossValidat
eData,\
                                      subCategoriesTransformedCrossVali
dateData,\
                                      teacherPrefixTransformedCrossVali
dateData,\
                                      schoolStateTransformedCrossValida
teData,\
                                      projectGradeTransformedCrossValid
ateData,\
                                      priceTransformedCrossValidateData
,\
                                      previouslyPostedTransformedCrossV
alidateData,\
                                      numberOfWordsInTitleTransformedCr
ossValidateData,\
                                      numberOfWordsInEssayTransformedCr
```

```
ossValidateData,\
                                        positiveSentimentScoreCrossValida
teData,\
                                        negativeSentimentScoreCrossValida
teData,\
                                        neutralSentimentScoreCrossValidat
eData,\
                                        compoundSentimentScoreCrossValida
teData,\
                                        tfIdfEssayTransformedCrossValidat
eData));
    crossValidateMergedData = svd.transform(crossValidateMergedData);

    testMergedData = hstack((categoriesTransformedTestData,\
                             subCategoriesTransformedTestData,\
                             teacherPrefixTransformedTestData,\
                             schoolStateTransformedTestData,\
                             projectGradeTransformedTestData,\
                             priceTransformedTestData,\
                             previouslyPostedTransformedTestData,\
                             numberOfWordsInTitleTransformedTestData,\
                             numberOfWordsInEssayTransformedTestData,\
                             positiveSentimentScoreTestData,\
                             negativeSentimentScoreTestData,\
                             neutralSentimentScoreTestData,\
                             compoundSentimentScoreTestData,\
                            tfIdfEssayTransformedTestData));
    testMergedData = svd.transform(testMergedData);

    svmClassifier = linear_model.SGDClassifier(loss = 'hinge', class_we
ight = 'balanced');
    tunedParameters = {'alpha': [0.0001, 0.01, 0.1, 1, 10, 100, 10000],
 'penalty': ['l1', 'l2']};
    classifier = GridSearchCV(svmClassifier, tunedParameters, cv = 5, s
coring = 'roc_auc');
    classifier.fit(trainingMergedData, classesTrainingSub);

    testScoresDataFrame = pd.DataFrame(data = np.hstack((classifier.cv_
results_['param_alpha'].data[:, None], classifier.cv_results_['param_pe
```

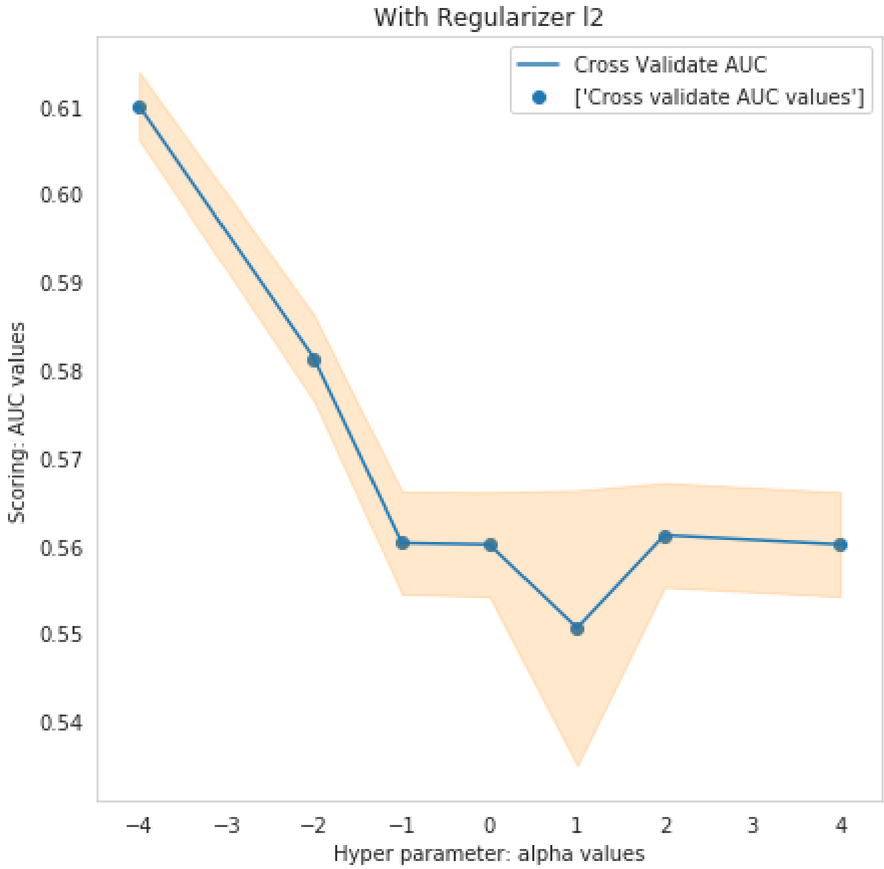
```
nalty'].data[:, None], classifier.cv_results_['mean_test_score'][:, Non
e], classifier.cv_results_['std_test_score'][:, None])), columns = ['al
pha', 'penalty', 'mts', 'stdts']);
    testScoresDataFrame

    crossValidateAucMeanValues = classifier.cv_results_['mean_test_scor
e'];
    crossValidateAucStdValues = classifier.cv_results_['std_test_score'
];

    testScoresDataFrame['logAlphaValues'] = [math.log10(x) for x in tes
tScoresDataFrame['alpha']];

    plt.plot(testScoresDataFrame[testScoresDataFrame['penalty'] == 'l1'
]['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penalty']
 == 'l1']['mts'], label = "Cross Validate AUC");
    plt.scatter(testScoresDataFrame[testScoresDataFrame['penalty'] ==
'l1']['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penal
ty'] == 'l1']['mts'], label = ['Cross validate AUC values']);
    plt.gca().fill_between(testScoresDataFrame[testScoresDataFrame['pen
alty'] == 'l1']['logAlphaValues'].values, np.array(testScoresDataFrame[
testScoresDataFrame['penalty'] == 'l1']['mts'].values - testScoresDataF
rame[testScoresDataFrame['penalty'] == 'l1']['stdts'].values, dtype = f
loat),\
                           np.array(testScoresDataFrame[testScoresDataF
rame['penalty'] == 'l1']['mts'].values + testScoresDataFrame[testScores
DataFrame['penalty'] == 'l1']['stdts'].values, dtype = float), alpha =
0.2, color = 'darkorange');
    plt.xlabel('Hyper parameter: alpha values');
    plt.ylabel('Scoring: AUC values');
    plt.title("With Regularizer l1");
    plt.grid();
    plt.legend();
    plt.show();

    plt.plot(testScoresDataFrame[testScoresDataFrame['penalty'] == 'l2'
]['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penalty']
 == 'l2']['mts'], label = "Cross Validate AUC");
    plt.scatter(testScoresDataFrame[testScoresDataFrame['penalty'] ==
```

```
'l2']['logAlphaValues'], testScoresDataFrame[testScoresDataFrame['penal
ty'] == 'l2']['mts'], label = ['Cross validate AUC values']);
    plt.gca().fill_between(testScoresDataFrame[testScoresDataFrame['pen
alty'] == 'l2']['logAlphaValues'].values, np.array(testScoresDataFrame[
testScoresDataFrame['penalty'] == 'l2']['mts'].values - testScoresDataF
rame[testScoresDataFrame['penalty'] == 'l2']['stdts'].values, dtype = f
loat),\
                           np.array(testScoresDataFrame[testScoresDataF
rame['penalty'] == 'l2']['mts'].values + testScoresDataFrame[testScores
DataFrame['penalty'] == 'l2']['stdts'].values, dtype = float), alpha =
0.2, color = 'darkorange');
    plt.xlabel('Hyper parameter: alpha values');
    plt.ylabel('Scoring: AUC values');
    plt.title("With Regularizer l2");
    plt.grid();
    plt.legend();

    plt.show();

    optimalHypParamValue = classifier.best_params_['alpha'];
    optimalHypParam2Value = classifier.best_params_['penalty'];
    svmClassifier = linear_model.SGDClassifier(loss = 'hinge', class_we
ight = 'balanced', alpha = optimalHypParamValue, penalty = optimalHypPa
ram2Value);
    svmClassifier.fit(trainingMergedData, classesTrainingSub);
    predScoresTraining = svmClassifier.predict(trainingMergedData);
    fprTrain, tprTrain, thresholdTrain = roc_curve(classesTraining, pre
dScoresTraining);
    predScoresTest = svmClassifier.predict(testMergedData);
    fprTest, tprTest, thresholdTest = roc_curve(classesTest, predScores
Test);

    plt.plot(fprTrain, tprTrain, label = "Train AUC = " + str(auc(fprTr
ain, tprTrain)));
    plt.plot(fprTest, tprTest, label = "Test AUC = " + str(auc(fprTest,
 tprTest)));
    plt.plot([0, 1], [0, 1], 'k-');
    plt.xlabel("fpr values");
    plt.ylabel("tpr values");
```

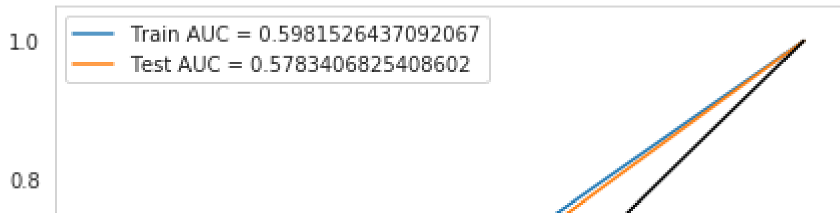
```
    plt.grid();
    plt.legend();
    plt.show();

    areaUnderRocValueTest = auc(fprTest, tprTest);

    print("Results of analysis using data of dimensions {} using suppor
t vector machine classifier: ".format(n_components));
    equalsBorder(40);
    print("Optimal Alpha value: ", optimalHypParamValue);
    equalsBorder(40);
    print("Optimal Regularizer: ", optimalHypParam2Value);
    equalsBorder(40);
    print("AUC value of test data: ", str(areaUnderRocValueTest));
    # Predicting classes of test data projects
    predictionClassesTest = svmClassifier.predict(testMergedData);
    equalsBorder(40);
    # Printing confusion matrix
    confusionMatrix = confusion_matrix(classesTest, predictionClassesTe
st);
    # Creating dataframe for generated confusion matrix
    confusionMatrixDataFrame = pd.DataFrame(data = confusionMatrix, ind
ex = ['Actual: NO', 'Actual: YES'], columns = ['Predicted: NO', 'Predic
ted: YES']);
    print("Confusion Matrix : ");
    equalsBorder(60);
    sbrn.heatmap(confusionMatrixDataFrame, annot = True, fmt = 'd', cma
p="YlGnBu");
    plt.show();
    # Adding results to results dataframe
    supportVectorMachineResultsDataFrame = supportVectorMachineResultsD
ataFrame.append({'Vectorizer': technique, 'Model': 'SVM(SGD - hinge los
s)', 'Hyper Parameter - alpha': optimalHypParamValue, 'AUC': areaUnderR
ocValueTest}, ignore_index = True);
```



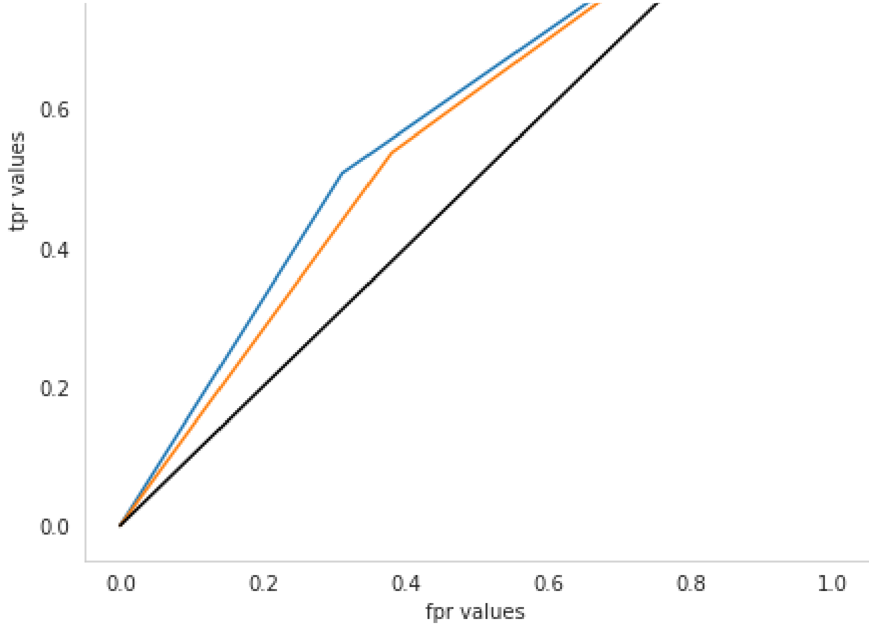
With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
0.62
0.60
0.58
0.56
0.54
0.52
0.50
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values



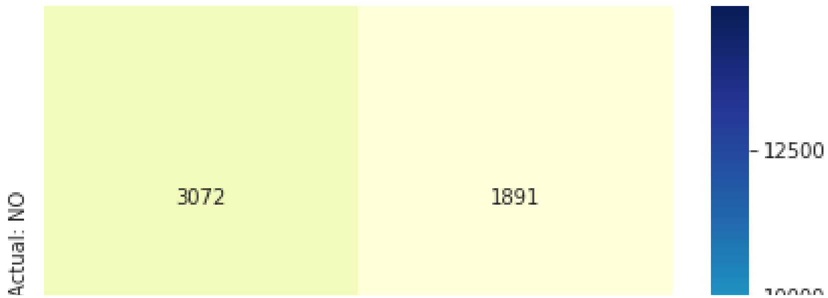
With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
0.61
0.60
0.59
0.58
0.57
0.56
0.55
0.54
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

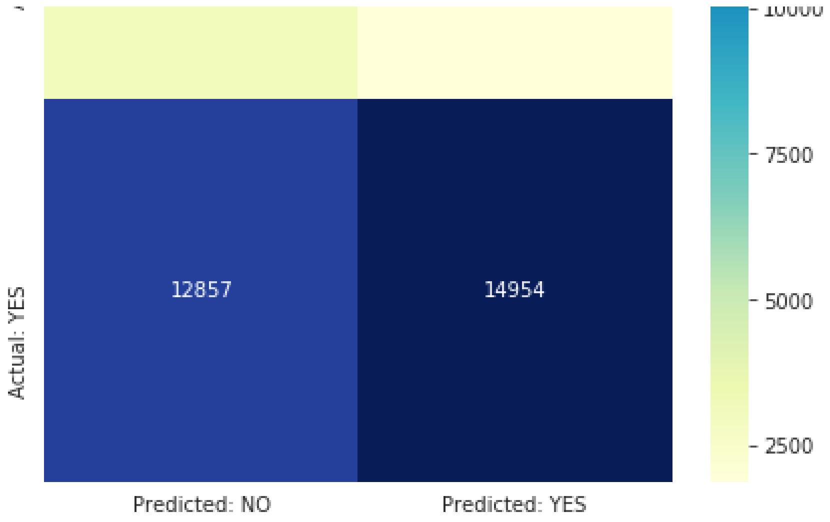


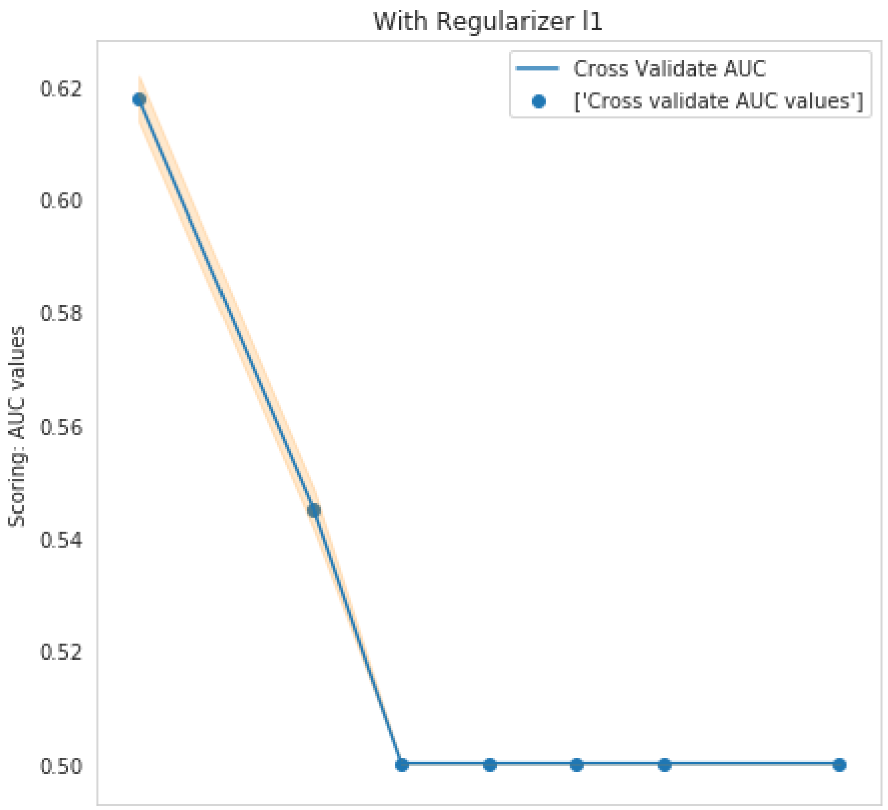
Train AUC = 0.5981526437092067
Test AUC = 0.5783406825408602
1.0
0.8



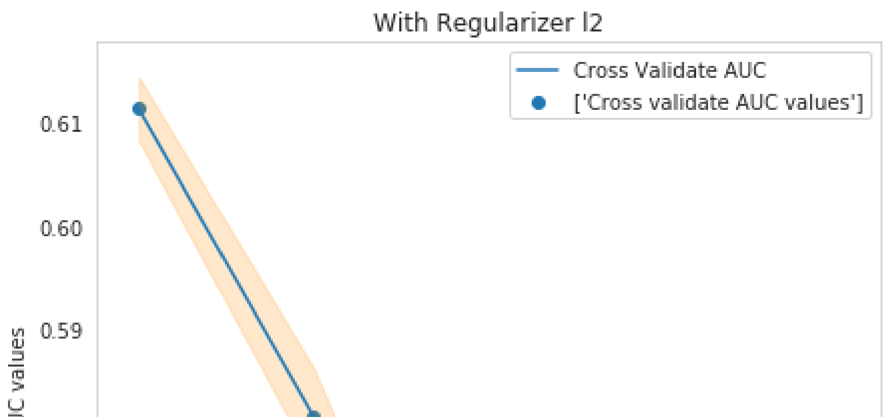

```
Results of analysis using data of dimensions 450 using support vector m
achine classifier:
========================================
Optimal Alpha value:  0.0001
========================================
Optimal Regularizer:  l1
========================================
AUC value of test data:  0.5783406825408602
========================================
Confusion Matrix :
============================================================
```

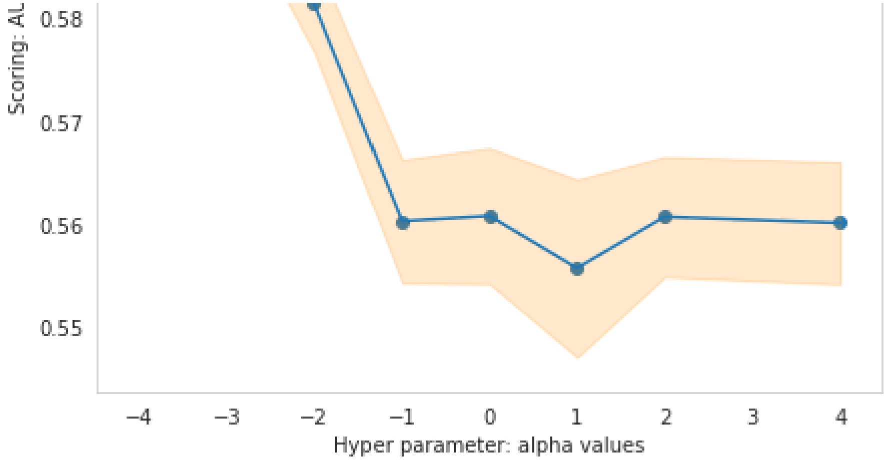



12857
14954
Actual: YES
Predicted: NO
Predicted: YES
10000
7500
5000
2500

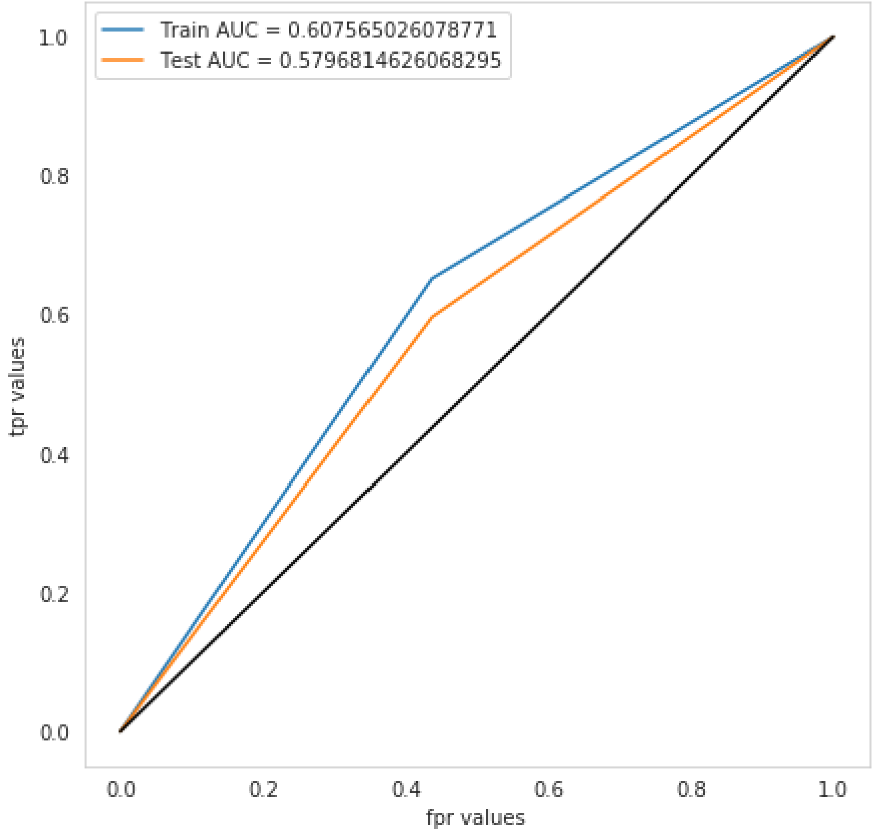

With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
Scoring: AUC values
0.62
0.60
0.58
0.56
0.54
0.52
0.50


−4 −3 −2 −1 0 1 2 3 4
Hyper parameter: alpha values


With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
0.61
0.60
0.59

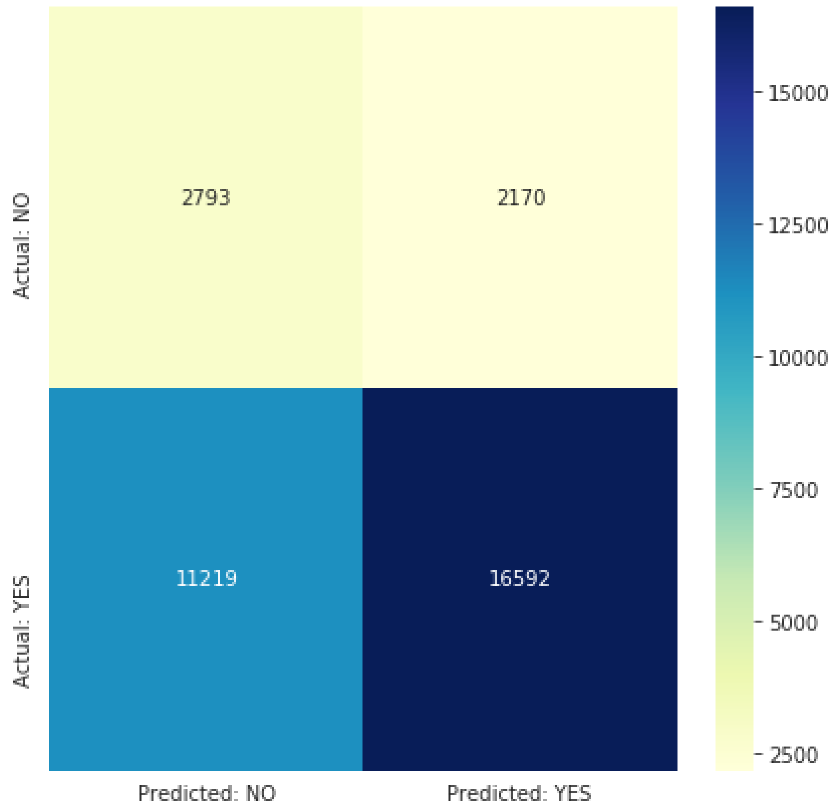


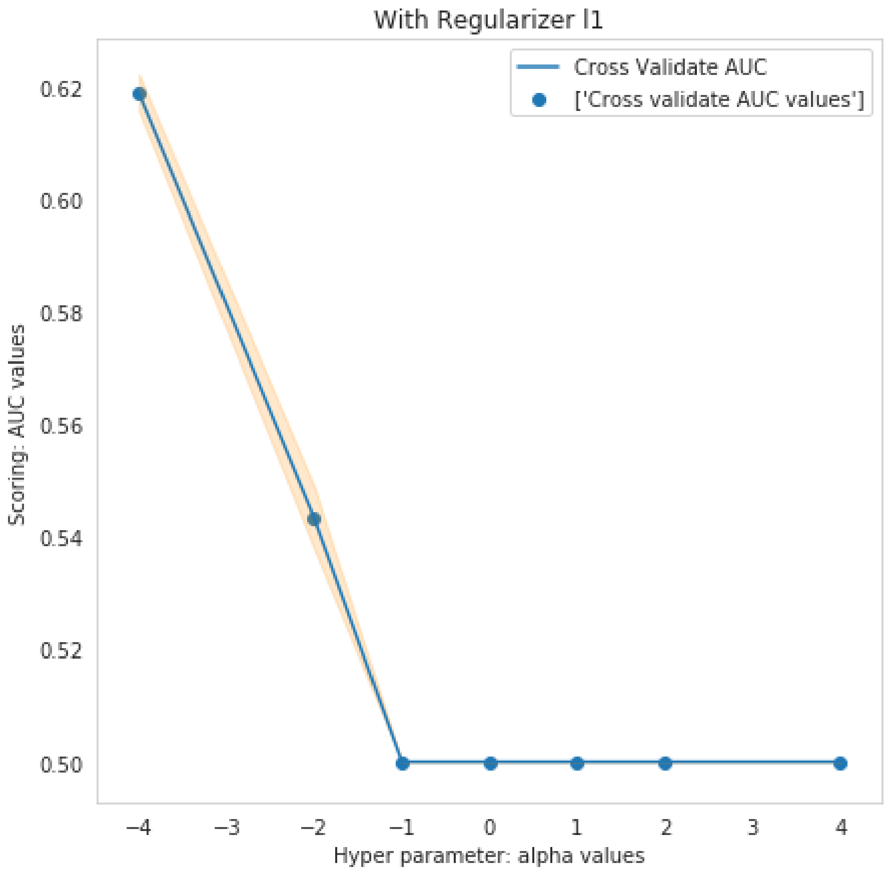
Scoring: AU
0.58
0.57
0.56
0.55
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

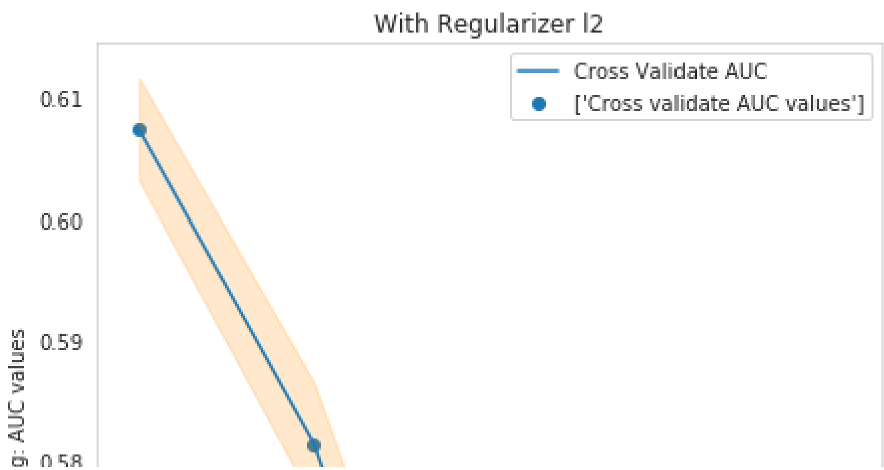


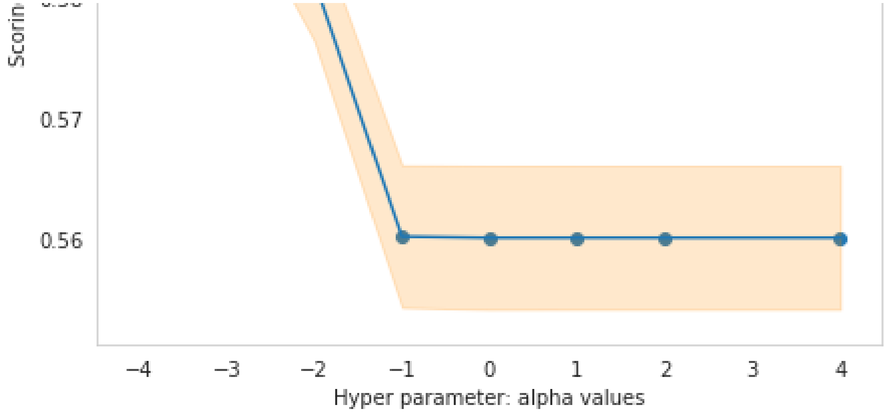
Train AUC = 0.607565026078771
Test AUC = 0.5796814626068295
tpr values
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
fpr values

```
Results of analysis using data of dimensions 600 using support vector m
achine classifier:
========================================
Optimal Alpha value:  0.0001
========================================
Optimal Regularizer:  l1
========================================
AUC value of test data:  0.5796814626068295
========================================
Confusion Matrix :
============================================================
```

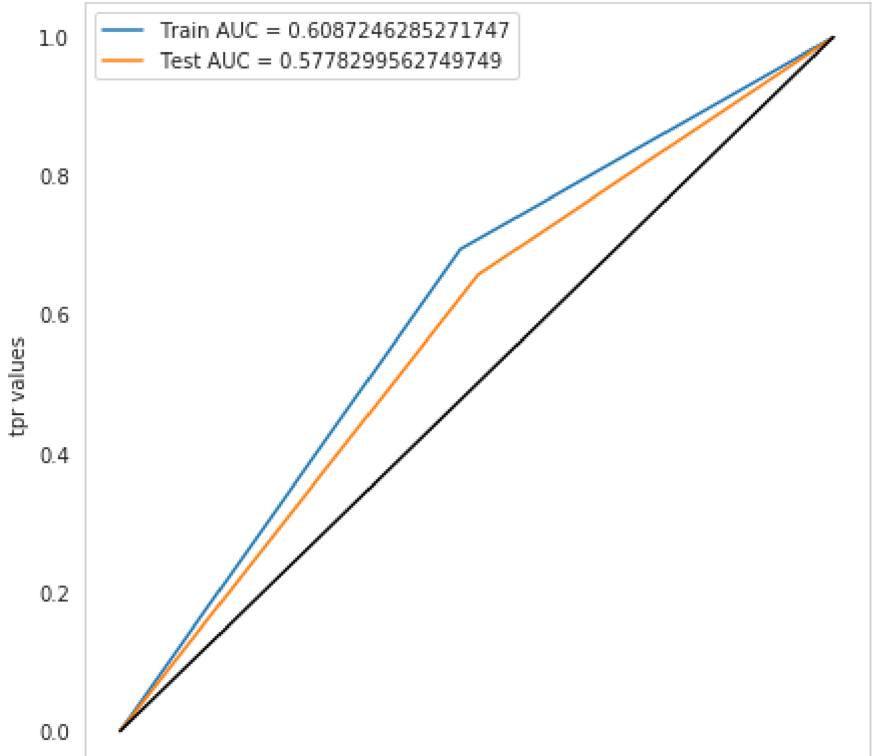



With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
0.62
0.60
0.58
0.56
0.54
0.52
0.50
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

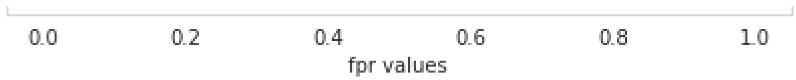

With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
0.61
0.60
0.59
0.58



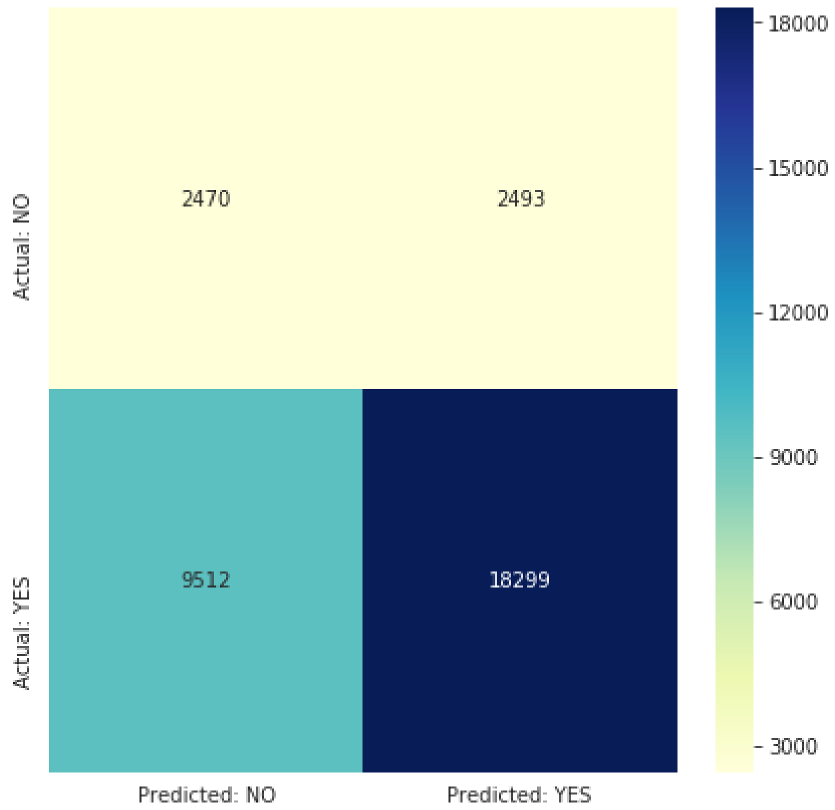
Scorin
0.57
0.56
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

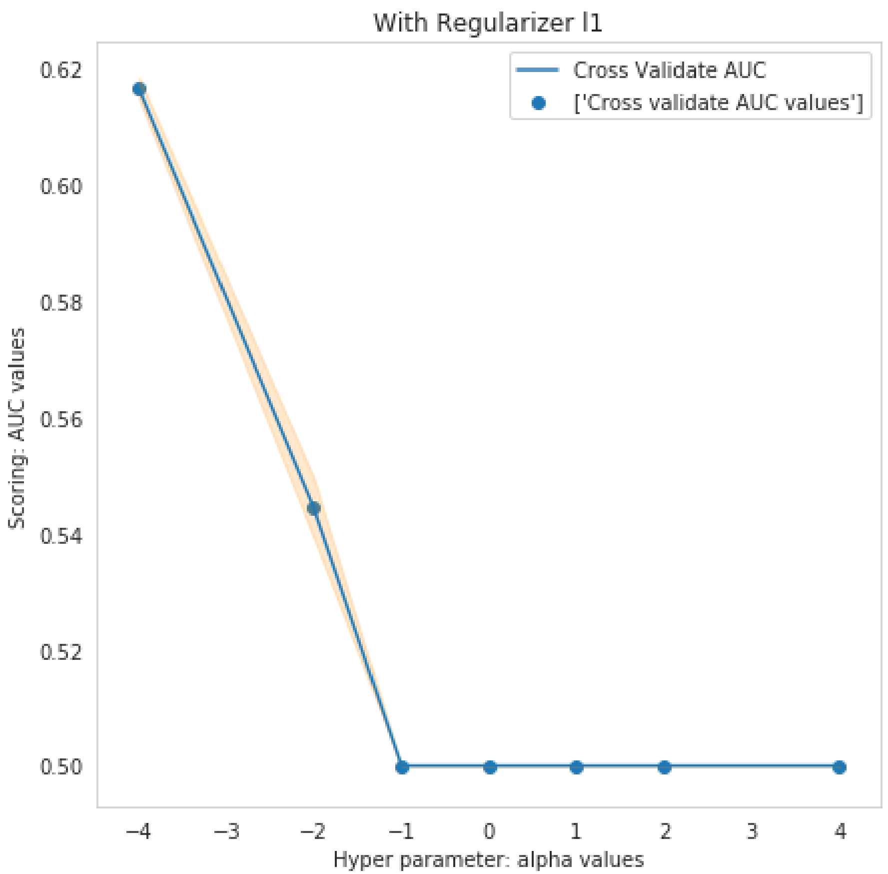


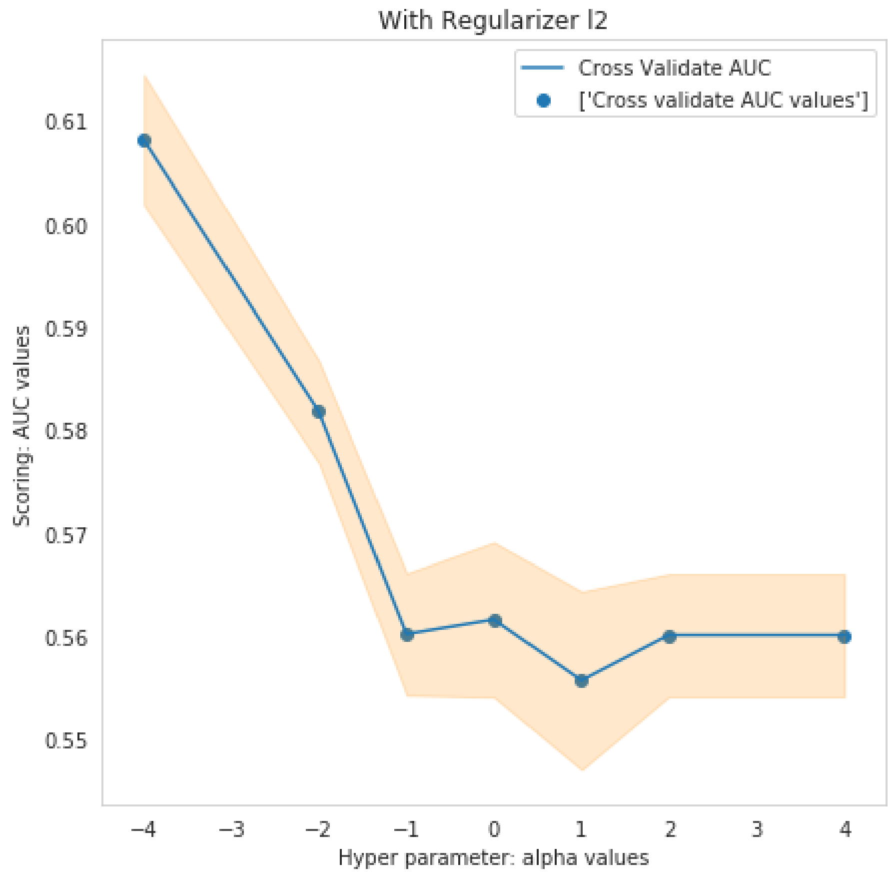
Train AUC = 0.6087246285271747
Test AUC = 0.5778299562749749
1.0
0.8
0.6
0.4
0.2
0.0
tpr values

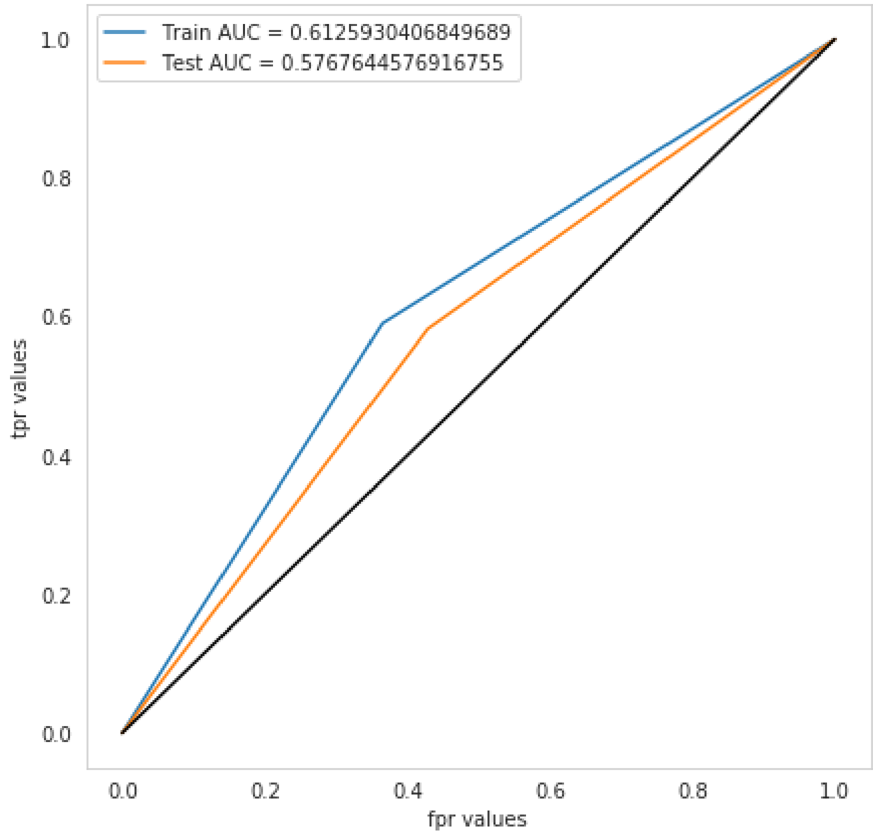



```
Results of analysis using data of dimensions 900 using support vector m
achine classifier:
========================================
Optimal Alpha value:  0.0001
========================================
Optimal Regularizer:  l1
========================================

AUC value of test data:  0.5778299562749749
========================================
Confusion Matrix :
============================================================
```

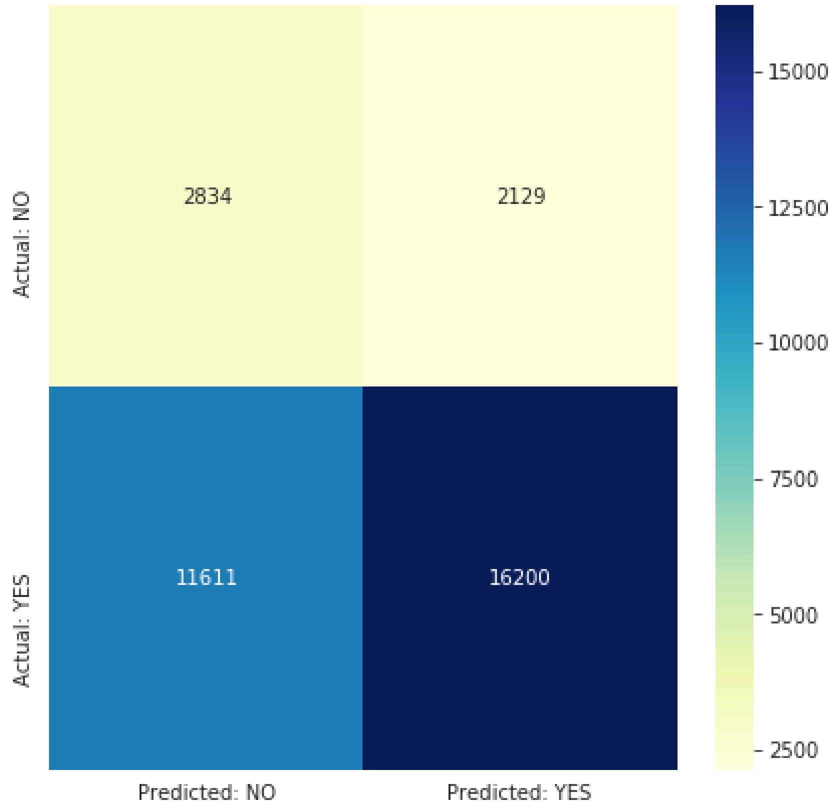



With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
0.62
0.60
0.58
0.56
0.54
0.52
0.50
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

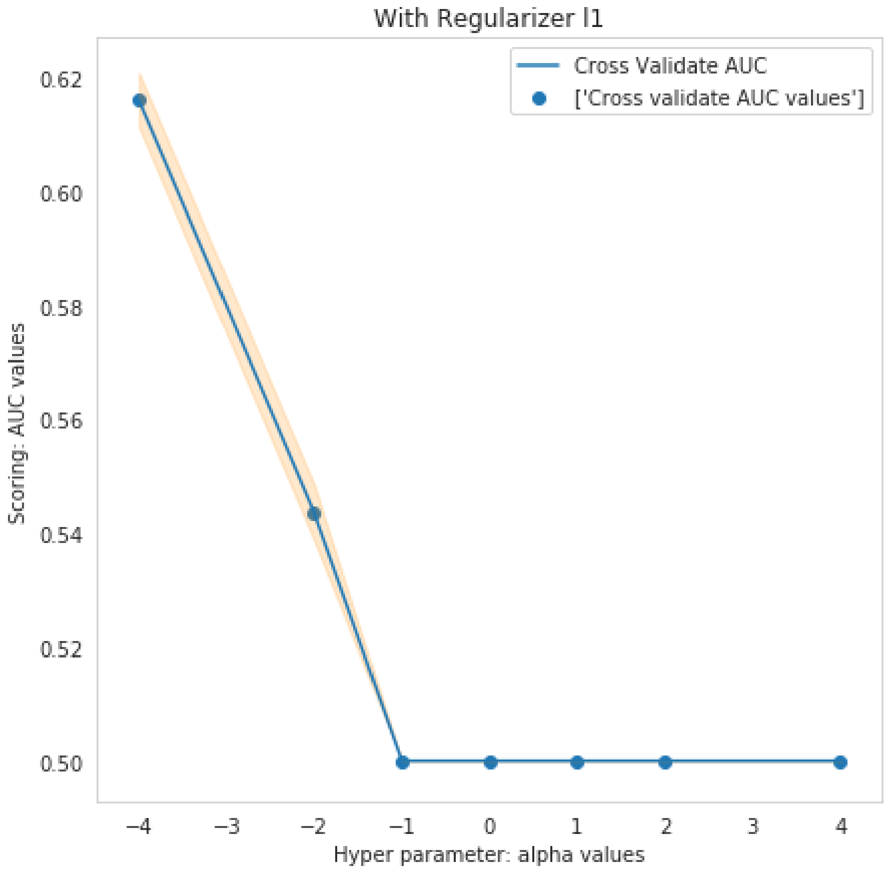

With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
0.61
0.60
0.59
0.58
0.57
0.56
0.55
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values



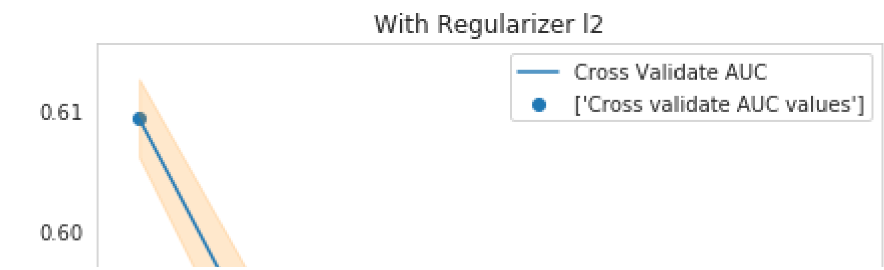

```
Results of analysis using data of dimensions 1200 using support vector
machine classifier:
========================================
Optimal Alpha value:  0.0001
========================================
Optimal Regularizer:  l1
========================================
AUC value of test data:  0.5767644576916755
========================================
Confusion Matrix :
============================================================
```



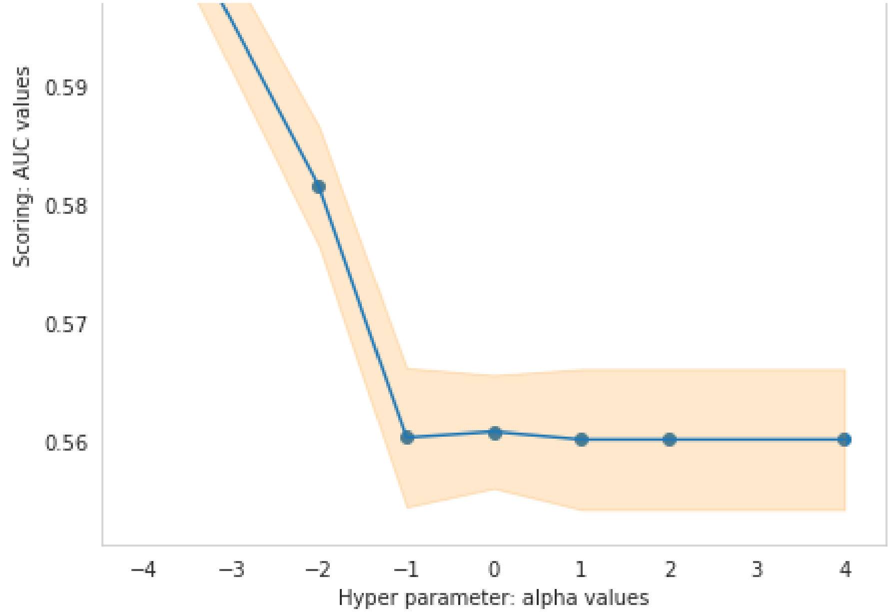
Actual: NO
Actual: YES
2834
2129
11611
16200
Predicted: NO
Predicted: YES
15000
12500
10000
7500
5000
2500

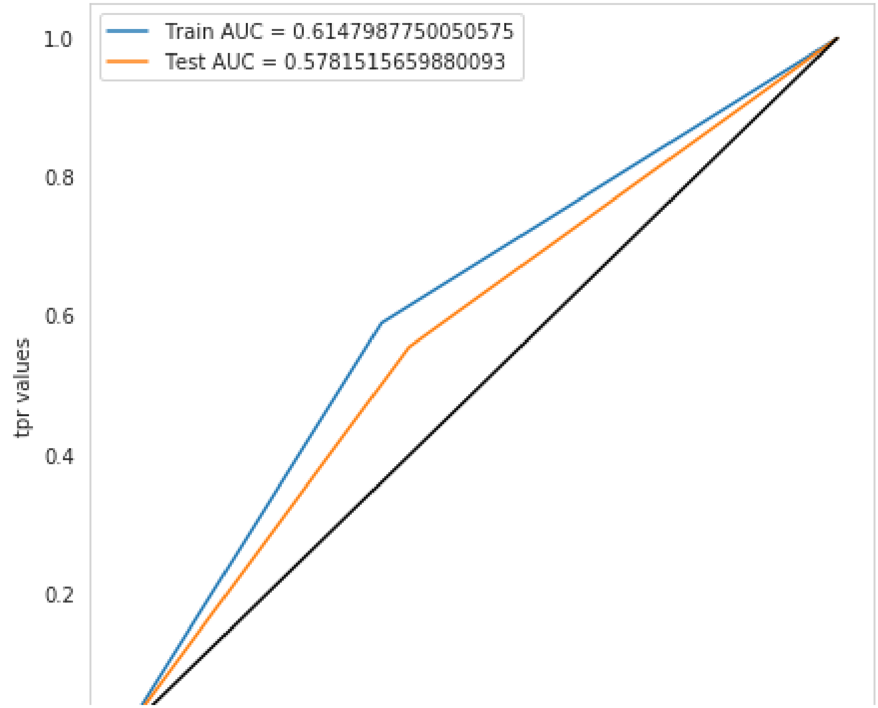


With Regularizer l1
Cross Validate AUC
['Cross validate AUC values']
0.62
0.60
0.58
0.56
0.54
0.52
0.50
Scoring: AUC values
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values

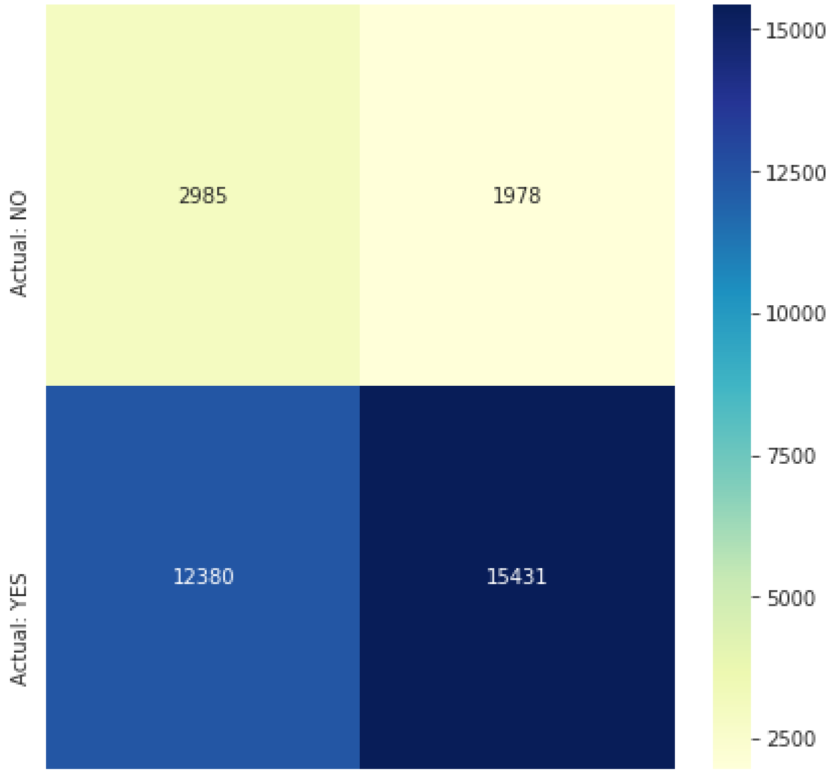


With Regularizer l2
Cross Validate AUC
['Cross validate AUC values']
0.61
0.60


Scoring: AUC values
0.59
0.58
0.57
0.56
−4
−3
−2
−1
0
1
2
3
4
Hyper parameter: alpha values


Train AUC = 0.6147987750050575
Test AUC = 0.5781515659880093
1.0
0.8
0.6
0.4
0.2
tpr values


```
Results of analysis using data of dimensions 1400 using support vector
machine classifier:
========================================
Optimal Alpha value:  0.0001
========================================
Optimal Regularizer:  l1
========================================
AUC value of test data:  0.5781515659880093
========================================
Confusion Matrix :
============================================================
```



Predicted: NO Predicted: YES

Summary of results of above classification using support vector machine

In [9]:
```
supportVectorMachineResultsDataFrame
```

Out[9]:

	Vectorizer	Model	Hyper Parameter - alpha	AUC	Regularizer
0	Bag of Words	SVM(SGD - hinge loss)	0.0100	0.6614	l2
1	Tf-Idf	SVM(SGD - hinge loss)	0.0001	0.6602	l1
2	Word2Vec	SVM(SGD - hinge loss)	0.0001	0.6437	l1
3	Tf-Idf Weighted Word2Vec	SVM(SGD - hinge loss)	0.0001	0.5971	l1
4	Tf-Idf(450 dim)	SVM(SGD - hinge loss)	0.0001	0.5783	l1
5	Tf-Idf(600 dim)	SVM(SGD - hinge loss)	0.0001	0.5796	l1
6	Tf-Idf(900 dim)	SVM(SGD - hinge loss)	0.0001	0.5778	l1
7	Tf-Idf(1200 dim)	SVM(SGD - hinge loss)	0.0001	0.5767	l1
8	Tf-Idf(1400 dim)	SVM(SGD - hinge loss)	0.0001	0.5781	l1

Conclusions of above analysis

1. From above analysis it seems that when data is balanced and text features are vectorized using bag of words the support vector machine is giving best results with auc value of 0.6614. The classification of negative points into negative points are also reasonable with this model but with other models it is kind of biased.
2. When classification is done using imbalanced data with reduced dimensions it is giving less auc values and a biased model(dumb model) that is classifying points incorrectly.
3. At last the good combination would be using balanced data with all categorical features, numerical features and text features vectorized with bag of words technique and hyper parameter value as 0.01.