DonorsChoose

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

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

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

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv',nrows=50000)
resource_data = pd.read_csv('resources.csv')
```

In [3]:

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

```
Number of data points in train data (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

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

Out[4]:

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

In [5]:

```
project_data["project_is_approved"].value_counts()
```

Out[5]:

```
1    42286
0     7714
Name: project_is_approved, dtype: int64
```

1.2 preprocessing of `project_subject_categories`

In [6]:

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

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

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

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

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of `project_subject_subcategories`

In [7]:

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

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

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

```
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

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

In [8]:

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

1.2.8 Univariate Analysis: Cost per project

In [9]:

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

Out[9]:

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

In [10]:

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

Out[10]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [11]:

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

print(project_data)
```

```
   Unnamed: 0       id                         teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.
2       21895  p182444  3465aaf82da834c0582ebd0ef8040ca0            Ms.
3          45  p246581  f3cb9bffbba169bef1a77b243e620b60           Mrs.
```

4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.
5	141660	p154343	a50a390e8327a95b77b9e495b58b9a6e	Mrs.
6	21147	p099819	9b40170bfa65e399981717ee8731efc3	Mrs.
7	94142	p092424	5bfd3d12fae3d2fe88684bbac570c9d2	Ms.
8	112489	p045029	487448f5226005d08d36bdd75f095b31	Mrs.
9	158561	p001713	140eeac1885c820ad5592a409a3a8994	Ms.
10	43184	p040307	363788b51d40d978fe276bcb1f8a2b35	Mrs.
11	127083	p251806	4ba7c721133ef651ca54a03551746708	Ms.
12	19090	p051126	5e52c92b7e3c472aad247a239d345543	Mrs.
13	15126	p003874	178f6ae765cd4e0fb143a77c47fd65e2	Mrs.
14	62232	p233127	424819801de22a60bba7d0f4354d0258	Ms.
15	67303	p132832	bb6d6d054824fa01576ab38dfa2be160	Ms.
16	127215	p174627	4ad7e280fddff889e1355cc9f29c3b89	Mrs.
17	157771	p152491	e39abda057354c979c5b075cffbe5f88	Ms.
18	122186	p196421	fcd9b003fc1891383f340a89da02a1a6	Mrs.
19	146331	p058343	8e07a98deb1bc74c75b97521e05b1691	Ms.
20	75560	p052326	e0c1aad1f71badeff703fadc15f57680	Mrs.
21	132078	p187097	2d4a4d2d774e5c2fdd25b2ba0e7341f8	Mrs.
22	84810	p165540	30f08fbe02eba5453c4ce2e857e88eb4	Ms.
23	8636	p219330	258ef2e6ab5ce007ac6764ce15d261ba	Mr.
24	21478	p126524	74f8690562c44fc88f65f845b9fe61d0	Mrs.
25	20142	p009037	b8bf3507cee960d5fedcb27719df2d59	Mrs.
26	33903	p040091	7a0a5de5ed94e7036946b1ac3eaa99d0	Ms.
27	1156	p161033	efdc3cf14d136473c9f62becc00d4cec	Teacher
28	35430	p085706	22c8184c4660f1c589bea061d14b7f35	Mrs.
29	22088	p032018	45f16a103f1e00b7439861d4e0728a59	Mrs.
...	...	...	...	...
49970	48139	p035589	03c50019548dd9ad9af9071fc76e5eeb	Ms.
49971	165303	p223730	0118a9857c874be87b315397f89e01d5	Mrs.
49972	63169	p104703	ba74528e836831eecac01773dcceddb1	Mrs.
49973	93353	p148480	fe2ad9b264d7a635834f36aa1b649ccd	Mrs.
49974	158902	p179531	6a4129e5310e29c21a6c50f9b6f808d3	Ms.
49975	121072	p086067	39a09b91c4c76ad631a61400f1ad47de	Ms.
49976	3059	p164918	78f09b1c41019e4f0455e3eb50d4dc03	Ms.
49977	164903	p214345	6b293f09676d0fe7d09f509e5cf0edec	Mr.
49978	15950	p258473	4c9a7219cf17ea5ded4819b3c23bd167	Ms.
49979	35813	p059990	a3a6de13f1e65fb1a6de7d0fd94ff9a7	Mrs.
49980	3524	p236931	a0655e02d03a5560f7ce9c627198ca4b	Mrs.
49981	97334	p078864	be29e53ae707eca7a35d0f9295e41691	Ms.
49982	11975	p039851	e46da6793a26b19ee5af40582230e29d	Mrs.
49983	156577	p216585	876fbb0add5e3ce09121bcde2553ed08	Mrs.
49984	127474	p224995	01e2ac2a6e6313d14f1e909e84f5987a	Mrs.
49985	78855	p175446	dc73ab17b5f5967cb282628bd7cd8f28	Ms.
49986	78097	p192812	e7a2e9a3312207fd60577f5edeaf64b9	Mrs.
49987	159360	p147171	49402e5295b1440a3eb361371e21e413	Ms.
49988	69407	p245054	a437264489c55252ff993bcb44f628f9	Mrs.
49989	107823	p049807	ef2b0681c4095ac55b0b39742a18fb2d	Ms.
49990	162953	p127477	72ea5e6bab3e3a509a0b1dba2a6df137	Mrs.
49991	153686	p183870	3a78d6aaa327aa63bbf71c75862fb17e	Mrs.
49992	2971	p236386	f87aba87d69fef2d72c617711e3371d3	Mrs.
49993	71389	p054472	fcd0839e05279f8478801dce254ba647	Ms.
49994	45277	p172774	dad8cf5bd23d36a93e6526737867192f	Ms.
49995	27461	p144673	88a8bdd51dca790df61b3cc2fafdae14	Mrs.
49996	89711	p138289	df7a55562859452b3aa897c3f3a53d19	Mr.
49997	5176	p159292	30f3dd18199ab24d10e6c8fdc1a877f8	Mrs.
49998	48461	p094764	bdf30a7b220e6b90218acbc57cf73440	Mrs.
49999	82189	p188201	56dbd8fbf3338c939a37f384eae0fd72	Mrs.

	school_state	project_submitted_datetime	project_grade_category
0	IN	2016-12-05 13:43:57	Grades PreK-2
1	FL	2016-10-25 09:22:10	Grades 6-8
2	AZ	2016-08-31 12:03:56	Grades 6-8
3	KY	2016-10-06 21:16:17	Grades PreK-2
4	TX	2016-07-11 01:10:09	Grades PreK-2
5	FL	2017-04-08 22:40:43	Grades 3-5
6	CT	2017-02-17 19:58:56	Grades 6-8
7	GA	2016-09-01 00:02:15	Grades 3-5
8	SC	2016-09-25 17:00:26	Grades PreK-2
9	NC	2016-11-17 18:18:56	Grades PreK-2
10	CA	2017-01-04 16:40:30	Grades 3-5
11	CA	2016-11-14 22:57:28	Grades PreK-2
12	NY	2016-05-23 15:46:02	Grades 6-8
13	OK	2016-10-17 09:49:27	Grades PreK-2
14	MA	2017-02-14 16:29:10	Grades PreK-2
15	TX	2016-10-05 21:05:38	Grades 3-5
16	FL	2017-01-18 10:59:05	Grades PreK-2
17	NY	2016-11-23 17:14:17	Grades 3-5

18	GA	2016-08-28 15:04:42	Grades PreK-2
19	OH	2016-08-06 13:05:20	Grades 3-5
20	PA	2016-10-07 18:27:02	Grades PreK-2
21	NC	2016-05-17 19:45:13	Grades 6-8
22	CA	2016-09-01 10:09:15	Grades 9-12
23	AL	2017-01-10 11:41:06	Grades 6-8
24	FL	2017-03-31 12:34:44	Grades PreK-2
25	AL	2017-03-09 15:36:20	Grades 3-5
26	TX	2016-09-18 22:10:40	Grades PreK-2
27	LA	2016-11-06 16:02:31	Grades 3-5
28	GA	2017-01-27 12:34:59	Grades 9-12
29	VA	2016-07-15 12:58:40	Grades PreK-2
...	...	...	...
49970	NY	2016-06-06 15:09:33	Grades PreK-2
49971	IL	2016-12-12 22:41:36	Grades 6-8
49972	UT	2017-01-10 12:31:27	Grades 3-5
49973	NE	2016-09-30 14:17:39	Grades PreK-2
49974	IL	2016-10-14 21:04:42	Grades PreK-2
49975	MO	2017-04-04 12:27:00	Grades 6-8
49976	IA	2017-01-06 12:58:57	Grades PreK-2
49977	OR	2016-09-01 01:25:19	Grades 3-5
49978	LA	2016-09-01 07:57:14	Grades 9-12
49979	IN	2016-07-31 20:26:10	Grades 3-5
49980	KY	2017-02-26 18:19:52	Grades 3-5
49981	CA	2017-01-11 16:40:20	Grades 3-5
49982	CA	2016-11-23 16:40:05	Grades PreK-2
49983	CA	2016-12-29 22:50:04	Grades PreK-2
49984	MA	2017-03-03 15:45:06	Grades 6-8
49985	MI	2017-01-30 07:57:43	Grades PreK-2
49986	LA	2016-10-08 19:11:57	Grades PreK-2
49987	MO	2017-04-07 13:13:24	Grades PreK-2
49988	FL	2016-08-22 18:14:26	Grades 9-12
49989	OH	2016-08-03 17:25:07	Grades PreK-2
49990	AR	2017-02-01 10:34:52	Grades 3-5
49991	AL	2017-01-29 00:00:22	Grades 9-12
49992	GA	2016-12-06 11:43:44	Grades 6-8
49993	IA	2016-08-16 14:12:09	Grades PreK-2
49994	IL	2016-08-12 16:55:48	Grades 3-5
49995	IL	2016-09-05 21:25:39	Grades PreK-2
49996	NV	2017-04-20 01:29:24	Grades 3-5
49997	SD	2016-08-22 16:46:27	Grades 3-5
49998	CT	2017-01-29 12:56:04	Grades PreK-2
49999	KY	2016-08-13 08:47:34	Grades PreK-2

	project_title \
0	Educational Support for English Learners at Home
1	Wanted: Projector for Hungry Learners
2	Soccer Equipment for AWESOME Middle School Stu...
3	Techie Kindergarteners
4	Interactive Math Tools
5	Flexible Seating for Mrs. Jarvis' Terrific Thi...
6	Chromebooks for Special Education Reading Program
7	It's the 21st Century
8	Targeting More Success in Class
9	Just For the Love of Reading--\r\nPure Pleasure
10	Reading Changes Lives
11	Elevating Academics and Parent Rapports Throug...
12	Building Life Science Experiences
13	Everyone deserves to be heard!
14	TABLETS CAN SHOW US THE WORLD
15	Making Recess Active
16	Making Great LEAP's With Leapfrog!
17	Technology Teaches Tomorrow's Talents Today
18	Test Time
19	Wiggling Our Way to Success
20	Magic Carpet Ride in Our Library
21	From Sitting to Standing in the Classroom
22	Books for Budding Intellectuals
23	Instrumental Power: Conquering STEAM!
24	S.T.E.A.M. Challenges(Science Technology Engin...
25	Math Masters!
26	Techy Teaching
27	4th Grade French Immersion Class Ipads
28	Hands-On Language and Literacy
29	Basic Classroom Supplies Needed
...	...
49970	Art Paper: Making Creation a Team Effort

```
49970          Art Paper: Making Creation a Team Effort
49971             Improving Skills Through Game-Playing
49972              STEAM supplies for art room centers!
49973                       Enhancing Math for Students
49974                            We Are Ready To Learn!
49975                  Crime Scene: Who Stole the Gum?
49976              Headphones for First Grade Learners!
49977                     Research and Writing for All!
49978    Shaping Tomorrow's Children, Through Art Today
49979                 Help Us Speak the Language of Art
49980  21st Century Learners Need 21st Century Techno...
49981                        Chromebooks for Success!!!
49982                                For the Love of Art
49983                  STEAM Bins for Future Engineers
49984         Helping Kids Have What They Need to Learn
49985                                    We can code!!!
49986                  Flood Our Class With Technology
49987                   Our Social Skills Aren't Wobbly!
49988  Math teacher in need of class set of graphing ...
49989        We Like to Move it Move it While We Learn!
49990             Flexible Seating for Dynamic Students
49991                          Renovate Roberson's Room
49992                Exploring History With Chromebooks
49993                     Book Buddies for New Readers!
49994                       Keep Calm and Learn On!
49995                iTeach: Using iPads in Instruction
49996                A \"Starbucks\" Classroom Redesign
49997                      Active Bodies = Active Minds
49998                      Can You Read My Writing Now?
49999           Inspiring Young Authors Through Reading

                                          project_essay_1  \
0      My students are English learners that are work...
1      Our students arrive to our school eager to lea...
2      \r\n\"True champions aren't always the ones th...
3      I work at a unique school filled with both ESL...
4      Our second grade classroom next year will be m...
5      I will be moving from 2nd grade to 3rd grade a...
6      My students are a dynamic and very energetic g...
7      Not only do our students struggle with poverty...
8      My students are enthusiastic and inquisitive l...
9      Over 95% of my students are on free or reduced...
10     \"There are many little ways to enlarge your w...
11     All of our students receive free breakfast, lu...
12     My students are always working on new projects...
13     I teach in a small school district in central ...
14     My students are my babies...I want the world f...
15     Located in West Dallas, my students face sever...
16     My Preschool children, ages 3-5 years old with...
17     My students are special because they come from...
18     I teach at a Title I school in a low-income ar...
19     We are apart of an urban district and many of ...
20     The students in our school come from diverse b...
21     My students walk into school every day full of...
22     Every day in my English classroom, we work to ...
23     100% of our musical students eat free breakfas...
24     This year, I am teaching in an EFL (Extended F...
25     My students are highly motivated to succeed. U...
26     I teach 22 bright 5 and 6 year olds. My studen...
27     My students spend most of their day learning f...
28     My students all have a primary diagnosis of au...
29     I have an awesome group of 24 students any tea...
...                                                  ...
49970  My school is a government funded Pre-Kindergar...
49971  I teach in a middle school on the south side o...
49972  My students are a phenomenal group of kids ran...
49973  I work with kids from first and second grade. ...
49974  Teaching in a high poverty/low-income school, ...
49975  Students are sometimes underwhelmed with the i...
49976  My students come from a variety of backgrounds...
49977  Buckman Arts Focus Elementary is a K-5 Arts in...
49978  My students come from various backgrounds, and...
49979  \"What are we doing today?\"  This is the firs...
49980  I teach an extremely talented and unique group...
49981  Yehey! Amazement as our class tried one of the...
49982  My first grade class is a diverse group of lea...
49983  The moment my second grade students walk in th...
49984  The students I work with come from a very poor...
```

```
49984  The students I work with come from a very poor...
49985  My students come from different economic backg...
49986  As a teacher at a Title I school, my students ...
49987  Welcome to my page! I'm the Counselor who work...
49988  Our school is considered \"High poverty\" and ...
49989  My classroom is a fully inclusive, high energy...
49990  I teach in a small neighborhood school which s...
49991  I teach at a very small and rural K-12 school....
49992  My students are being challenged in 6th grade ...
49993  My students come from various backgrounds and ...
49994  My classroom has 13 students with a variety of...
49995  I teach kindergarten in a Title I school in Ch...
49996  The students in our room are enrolled at an el...
49997  Welcome! My students and I are pleased to have...
49998  My school empowers 538 students in grades pre-...
49999  We have GRIT!  If you want to meet tenacious, ...

                                        project_essay_2 project_essay_3  \
0      \"The limits of your language are the limits o...             NaN
1      The projector we need for our school is very c...             NaN
2      The students on the campus come to school know...             NaN
3      My students live in high poverty conditions wi...             NaN
4      For many students, math is a subject that does...             NaN
5      These flexible seating options will allow my s...             NaN
6      My students are an engaging and active group o...             NaN
7      My students need 4 iPads, the latest technolog...             NaN
8      My second graders need extra activity time dur...             NaN
9      Reading is Fundamental! My students will read ...             NaN
10     I've had 8 sets of students enjoy the books in...             NaN
11     With three chromebooks, I can teach the Common...             NaN
12     My Spanish Dual Language students are always r...             NaN
13     My students are smart, creative, and also have...             NaN
14     Having this computer in the classroom would pr...             NaN
15     Due to the size of our school, and the tiny na...             NaN
16     Having a set of Leapfrog iPads and educational...             NaN
17     Classroom ChromebookCar\r\n\r\nMy name is Shan...             NaN
18     My 2nd grade students will benefit from having...             NaN
19     Many of my students struggle to sit still for ...             NaN
20     Each week our students love visiting the schoo...             NaN
21     I want to purchase desks in my classroom that ...             NaN
22     My students need books that interest them so t...             NaN
23     We need classroom instruments for our band pro...             NaN
24     I will use these items to create S.T.E.A.M. bi...             NaN
25     These math games will help reinforce the skill...             NaN
26     The iPads will be effectively used to improve ...             NaN
27     The iPads will also be used to enhance the stu...             NaN
28     Children with autism struggle in core deficit ...             NaN
29     My students need basic school supplies such as...             NaN
...                                                  ...             ...
49970  As a teacher, I can use some of this paper to ...             NaN
49971  The students at my  middle school are hard wor...             NaN
49972  These materials will focus on the creation pro...             NaN
49973  My students in first grade and math need these...             NaN
49974  I am asking for interactive phonics journals t...             NaN
49975  Students will be exploring the career of foren...             NaN
49976  First graders use Ipads and laptops daily. I u...             NaN
49977  Why should students hold back their curiosity ...             NaN
49978  In Fine Arts Survey, I find that my students l...             NaN
49979  Students will be learning about art of other c...             NaN
49980  “We need technology in every classroom and in ...             NaN
49981  Chrome books will help my students be familiar...             NaN
49982  I am requesting a variety of art supplies that...             NaN
49983  My second graders love to learn and explore ev...             NaN
49984  The resources that I selected address the lear...             NaN
49985  Students will be able to navigate their way ar...             NaN
49986  My students need these Dell laptops to be able...             NaN
49987  Our students are the most resilient kids in KC...             NaN
49988  Using graphing utilities is part of the curric...             NaN
49989  Many of my students are frustrated by the amou...             NaN
49990  The wobble cushions and bouncy bands will be a...             NaN
49991  Collaboration is a daily focus in my English c...             NaN
49992  My students are able to dig into social studie...             NaN
49993  My students need leveled readers to read at ho...             NaN
49994  This project will allow us to establish a \"ca...             NaN
49995  Teaching kindergarten is all about differentia...             NaN
49996  he research is clear. Students who engage in c...             NaN
49997  Students in my class currently sit, bounce, wi...             NaN
49998  A 7 year old girl in my class desperately wear...             NaN
```

```
49998  A 7 year old girl in my class desperately year...              NaN
49999  Receiving books written by the same author wil...              NaN

      project_essay_4                          project_resource_summary  \
0                 NaN  My students need opportunities to practice beg...
1                 NaN  My students need a projector to help with view...
2                 NaN  My students need shine guards, athletic socks,...
3                 NaN  My students need to engage in Reading and Math...
4                 NaN  My students need hands on practice in mathemat...
5                 NaN  My students need movement to be successful. Be...
6                 NaN  My students need some dependable laptops for d...
7                 NaN  My students need ipads to help them access a w...
8                 NaN  My students need three devices and three manag...
9                 NaN  My students need great books to use during Ind...
10                NaN  My students need books by their favorite autho...
11                NaN  My students need paper, three chromebooks, and...
12                NaN  My students need 3D and 4D life science activi...
13                NaN  My students need access to technology that wil...
14                NaN  My students need 5 tablets for our classroom t...
15                NaN  My students need activities to play during rec...
16                NaN  My students need 2 LeapPad that will engage th...
17                NaN  My students need Chromebooks to publish writte...
18                NaN  My students need privacy partitions to use whi...
19                NaN  My students need 7 Hokki stools to encourage a...
20                NaN  My students need carpet in our library to brig...
21                NaN  My students need desks to stand at and be able...
22                NaN  My students need books so that they can become...
23                NaN  My students need these instruments to give the...
24                NaN  My students need building materials, such as g...
25                NaN  My students need the learning centers and mult...
26                NaN  My students need 2 ipad minis to enhance learn...
27                NaN  My students need Ipads to work in smaller grou...
28                NaN  My students need to increase language and lite...
29                NaN  My students need basic school supplies such as...
...               ...                                              ...
49970             NaN  My students need Duo-Finish Butcher Paper Roll...
49971             NaN  My students need a variety of math games becau...
49972             NaN  My students need hands-on materials that will ...
49973             NaN  My students need will use money puzzles and ma...
49974             NaN  My students need interactive phonics journals ...
49975             NaN  My students need materials to discover the car...
49976             NaN  My students need headphones to use with their ...
49977             NaN  My students need 6 more Chromebooks to add to ...
49978             NaN  My students need art-time dough and paper mach...
49979             NaN  My students need printing inks, markers, paint...
49980             NaN  My students need 2 Chromebooks and Google Chro...
49981             NaN  My students need chrome books to help them kee...
49982             NaN  My students need creative opportunities for ar...
49983             NaN  My students need materials to help them fall i...
49984             NaN  My students need basic curriculum and books to...
49985             NaN  My students need the osmo coding system and ip...
49986             NaN  My students need laptops to take AR tests, do ...
49987             NaN  My students need a whiteboard table and wobble...
49988             NaN  My students need access to graphing calculator...
49989             NaN  My students need a way to move while being sea...
49990             NaN  My students need Wobble cushions, bouncy bands...
49991             NaN  My students need 10 funtioning tables that wil...
49992             NaN  My students need Chromebooks for research and ...
49993             NaN  My students need grade level appropriate books...
49994             NaN  My students need classroom supplies like expo ...
49995             NaN  My students need iPads to help customize learn...
49996             NaN  My students need Hokki Stools to maximize enga...
49997             NaN  My students need 12 Learniture Active Learning...
49998             NaN  My students need Dimples hand strengthener, th...
49999             NaN  My students need copies of books by the same a...

       teacher_number_of_previously_posted_projects  project_is_approved  \
0                                                 0                    0
1                                                 7                    1
2                                                 1                    0
3                                                 4                    1
4                                                 1                    1
5                                                 1                    1
6                                                 1                    1
7                                                 7                    1
8                                                28                    1
9                                                36                    1
10                                               37                    1
```

10	37	1
11	32	1
12	5	0
13	30	1
14	15	0
15	3	1
16	1	1
17	0	1
18	0	1
19	9	1
20	23	1
21	0	1
22	0	0
23	2	1
24	0	1
25	11	0
26	2	1
27	2	1
28	5	0
29	0	1
...	...	...
49970	7	1
49971	1	1
49972	11	1
49973	2	1
49974	50	1
49975	1	0
49976	0	1
49977	0	1
49978	7	1
49979	21	0
49980	45	1
49981	0	1
49982	6	1
49983	3	1
49984	0	1
49985	6	0
49986	1	1
49987	0	1
49988	0	1
49989	3	1
49990	0	1
49991	0	1
49992	0	1
49993	1	1
49994	6	0
49995	8	1
49996	6	1
49997	9	1
49998	37	1
49999	0	1

	clean_categories \
0	Literacy_Language
1	History_Civics Health_Sports
2	Health_Sports
3	Literacy_Language Math_Science
4	Math_Science
5	Literacy_Language SpecialNeeds
6	Literacy_Language SpecialNeeds
7	Math_Science
8	Health_Sports
9	Literacy_Language
10	Literacy_Language
11	Literacy_Language AppliedLearning
12	Math_Science
13	SpecialNeeds
14	Literacy_Language
15	Health_Sports
16	Literacy_Language SpecialNeeds
17	Math_Science Literacy_Language
18	AppliedLearning
19	Health_Sports
20	Literacy_Language
21	Math_Science SpecialNeeds
22	Literacy_Language
23	Music_Arts

```
24                         Math_Science
25                         Math_Science
26        Literacy_Language Math_Science
27        Literacy_Language Math_Science
28        Literacy_Language SpecialNeeds
29     Literacy_Language AppliedLearning
...                                  ...
49970                        Music_Arts
49971                      Math_Science
49972            Math_Science Music_Arts
49973                      Math_Science
49974     Literacy_Language SpecialNeeds
49975                      Math_Science
49976                 Literacy_Language
49977   Literacy_Language History_Civics
49978                        Music_Arts
49979          History_Civics Music_Arts
49980     Literacy_Language Math_Science
49981     Literacy_Language Math_Science
49982                        Music_Arts
49983                      Math_Science
49984     Literacy_Language Math_Science
49985                      Math_Science
49986                 Literacy_Language
49987                   AppliedLearning
49988                      Math_Science
49989                     Health_Sports
49990     Literacy_Language Math_Science
49991                 Literacy_Language
49992                    History_Civics
49993                 Literacy_Language
49994                      SpecialNeeds
49995     Literacy_Language Math_Science
49996                     Health_Sports
49997                     Health_Sports
49998     Literacy_Language SpecialNeeds
49999                 Literacy_Language

                          clean_subcategories  \
0                                 ESL Literacy
1                  Civics_Government TeamSports
2                    Health_Wellness TeamSports
3                          Literacy Mathematics
4                                  Mathematics
5               Literature_Writing SpecialNeeds
6                         Literacy SpecialNeeds
7                                  Mathematics
8                               Health_Wellness
9                   Literacy Literature_Writing
10                                    Literacy
11                   Literacy ParentInvolvement
12     EnvironmentalScience Health_LifeScience
13                                 SpecialNeeds
14                                    Literacy
15                              Health_Wellness
16                        Literacy SpecialNeeds
17          AppliedSciences Literature_Writing
18                             EarlyDevelopment
19                              Health_Wellness
20                                    Literacy
21              Health_LifeScience SpecialNeeds
22                                    Literacy
23                                       Music
24                  AppliedSciences Mathematics
25                                 Mathematics
26                         Literacy Mathematics
27                    ForeignLanguages Mathematics
28                         Literacy SpecialNeeds
29                                Literacy Other
...                                         ...
49970                               VisualArts
49971                              Mathematics
49972          EnvironmentalScience VisualArts
49973                              Mathematics
49974                     Literacy SpecialNeeds
49975                AppliedSciences Mathematics
49976                             ESL Literacy
```

```
49977       Literature_Writing SocialSciences
49978            PerformingArts VisualArts
49979          History_Geography VisualArts
49980                   Literacy Mathematics
49981        Literature_Writing Mathematics
49982                           VisualArts
49983                       AppliedSciences
49984                   Literacy Mathematics
49985          AppliedSciences Mathematics
49986                     Literature_Writing
49987  CharacterEducation College_CareerPrep
49988                           Mathematics
49989           Gym_Fitness Health_Wellness
49990        Literature_Writing Mathematics
49991           Literacy Literature_Writing
49992       History_Geography SocialSciences
49993                          ESL Literacy
49994                          SpecialNeeds
49995                   Literacy Mathematics
49996                        Health_Wellness
49997                        Health_Wellness
49998       Literature_Writing SpecialNeeds
49999                     Literature_Writing
```

	essay	price	quantity
0	My students are English learners that are work...	154.60	23
1	Our students arrive to our school eager to lea...	299.00	1
2	\r\n\"True champions aren't always the ones th...	516.85	22
3	I work at a unique school filled with both ESL...	232.90	4
4	Our second grade classroom next year will be m...	67.98	4
5	I will be moving from 2nd grade to 3rd grade a...	113.22	11
6	My students are a dynamic and very energetic g...	159.99	3
7	Not only do our students struggle with poverty...	229.00	4
8	My students are enthusiastic and inquisitive l...	241.98	6
9	Over 95% of my students are on free or reduced...	125.36	14
10	\"There are many little ways to enlarge your w...	100.21	10
11	All of our students receive free breakfast, lu...	431.77	8
12	My students are always working on new projects...	219.46	22
13	I teach in a small school district in central ...	399.99	1
14	My students are my babies...I want the world f...	91.94	10
15	Located in West Dallas, my students face sever...	435.84	24
16	My Preschool children, ages 3-5 years old with...	298.43	7
17	My students are special because they come from...	158.63	12
18	I teach at a Title I school in a low-income ar...	59.98	4
19	We are apart of an urban district and many of ...	749.42	7
20	The students in our school come from diverse b...	213.85	1
21	My students walk into school every day full of...	250.91	4
22	Every day in my English classroom, we work to ...	278.09	21
23	100% of our musical students eat free breakfas...	299.98	2
24	This year, I am teaching in an EFL (Extended F...	250.00	6
25	My students are highly motivated to succeed. U...	268.99	2
26	I teach 22 bright 5 and 6 year olds. My studen...	280.83	4
27	My students spend most of their day learning f...	660.84	7
28	My students all have a primary diagnosis of au...	129.98	3
29	I have an awesome group of 24 students any tea...	86.74	53
...	...	...	...
49970	My school is a government funded Pre-Kindergar...	159.98	2
49971	I teach in a middle school on the south side o...	158.14	18
49972	My students are a phenomenal group of kids ran...	102.95	6
49973	I work with kids from first and second grade. ...	287.86	28
49974	Teaching in a high poverty/low-income school, ...	117.98	2
49975	Students are sometimes underwhelmed with the i...	167.05	7
49976	My students come from a variety of backgrounds...	12.44	25
49977	Buckman Arts Focus Elementary is a K-5 Arts in...	149.99	6
49978	My students come from various backgrounds, and...	100.99	16
49979	\"What are we doing today?\" This is the firs...	215.09	486
49980	I teach an extremely talented and unique group...	165.95	4
49981	Yehey! Amazement as our class tried one of the...	157.00	2
49982	My first grade class is a diverse group of lea...	153.30	20
49983	The moment my second grade students walk in th...	100.91	10
49984	The students I work with come from a very poor...	324.87	13
49985	My students come from different economic backg...	719.77	7
49986	As a teacher at a Title I school, my students ...	310.74	1
49987	Welcome to my page! I'm the Counselor who work...	484.36	7
49988	Our school is considered \"High poverty\" and ...	189.62	8
49989	My classroom is a fully inclusive, high energy...	13.95	24
49990	I teach in a small neighborhood school which s...	60.04	19

```
49991  I teach at a very small and rural K-12 school....  159.57        50
49992  My students are being challenged in 6th grade ...  175.42         5
49993  My students come from various backgrounds and ...  381.00        14
49994  My classroom has 13 students with a variety of...  210.92        22
49995  I teach kindergarten in a Title I school in Ch...  539.98         2
49996  The students in our room are enrolled at an el...  214.12         6
49997  Welcome! My students and I are pleased to have...   52.05        14
49998  My school empowers 538 students in grades pre-...  102.25         6
49999  We have GRIT!  If you want to meet tenacious, ...  505.43        41

[50000 rows x 20 columns]
```

In [12]:

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

1.3 Text preprocessing

1.3.1 Essay Text

In [13]:

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

Out[13]:

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

In [14]:

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

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
```

```
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==================================================
```

In [15]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [16]:

```
sent = decontracted(project_data['essay'].values[4000])
print(sent)
print("="*50)
```

```
I teach language arts and social studies to about 50 students each day.  I teach two groups of ama
zing kids each day!\r\n\r\nThe students in my classroom range from advanced or gifted learners to
students with various learning disabilities. My school is located in an urban environment in
Maryland. The school is a Title I (low-income) school, and 99% of the students in the school recei
ve free and reduced price lunch. All students at my school receive free breakfast which is the mos
t important meal of the day!High interest reading supports comprehension and learning. I want to e
ncourage a love of reading by choosing books that interest my third grade students. Many of my
students are classified as \"struggling readers\". There is extensive research to support the
premise that the best way to become a better reader is to read more. In order for my students to b
ecome better or more fluent readers I need to increase both the quantity and quality of their read
ing.  They need reading materials that they can read and will want to read. \r\n\r\nI want to send
my students into summer vacation with a high interest book. If they find success and interest with
one book, research shows that learning will generate more learning! The book I have chosen is read
able, has a convincing plot, and has realistic characters.nannan
==================================================
```

In [17]:

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

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

```
I teach language arts and social studies to about 50 students each day.  I teach two groups of ama
zing kids each day!    The students in my classroom range from advanced or gifted learners to stud
ents with various learning disabilities. My school is located in an urban environment in Maryland.
The school is a Title I (low-income) school, and 99% of the students in the school receive free an
d reduced price lunch. All students at my school receive free breakfast which is the most
important meal of the day!High interest reading supports comprehension and learning. I want to enc
ourage a love of reading by choosing books that interest my third grade students. Many of my
students are classified as  struggling readers . There is extensive research to support the
premise that the best way to become a better reader is to read more. In order for my students to b
ecome better or more fluent readers I need to increase both the quantity and quality of their read
ing.  They need reading materials that they can read and will want to read.     I want to send my
students into summer vacation with a high interest book. If they find success and interest with on
e book, research shows that learning will generate more learning! The book I have chosen is readab
le, has a convincing plot, and has realistic characters.nannan
```

In [18]:

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

```
I teach language arts and social studies to about 50 students each day I teach two groups of amazi
ng kids each day The students in my classroom range from advanced or gifted learners to students w
ith various learning disabilities My school is located in an urban environment in Maryland The sch
ool is a Title I low income school and 99 of the students in the school receive free and reduced p
rice lunch All students at my school receive free breakfast which is the most important meal of th
e day High interest reading supports comprehension and learning I want to encourage a love of read
ing by choosing books that interest my third grade students Many of my students are classified as
struggling readers There is extensive research to support the premise that the best way to become
a better reader is to read more In order for my students to become better or more fluent readers I
need to increase both the quantity and quality of their reading They need reading materials that t
hey can read and will want to read I want to send my students into summer vacation with a high int
erest book If they find success and interest with one book research shows that learning will gener
ate more learning The book I have chosen is readable has a convincing plot and has realistic chara
cters nannan
```

In [19]:

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

In [20]:

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

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000 [01:
18<00:00, 635.47it/s]
```

In [21]:

```
# after preprocesing
preprocessed_essays[2000]

project_data['essay']=pd.DataFrame(preprocessed_essays)
```

1.3.2 Project title Text

In [22]:

```
# similarly you can preprocess the titles also

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

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:03<00:00, 13752.97it/s]
```

In [23]:

```
preprocessed_titles[2000]

project_data['project_title']=pd.DataFrame(preprocessed_titles)
```

Computing Sentiments Score

In [24]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

nltk.download('vader_lexicon')

categories=list(project_data['essay'].values)


sent_pos=[]
sent_neg=[]
```

```
sent_neu=[]
sent_comp=[]


for i in categories:
    sid=SentimentIntensityAnalyzer()
    ss = sid.polarity_scores(i)
    sent_pos.append(ss['pos'])
    sent_neg.append(ss['neg'])
    sent_neu.append(ss['neu'])
    sent_comp.append(ss['compound'])


project_data['sentiment_pos_essay']=sent_pos
project_data['sentiment_neg_essay']=sent_neg
project_data['sentiment_neu_essay']=sent_neu
project_data['sentiment_compound_essay']=sent_comp
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Roshan\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Number of words in Essays

In [25]:

```
print(project_data['essay'].values)
```

```
['my students english learners working english second third languages we melting pot refugees immi
grants native born americans bringing gift language school we 24 languages represented english lea
rner program students every level mastery we also 40 countries represented families within school
each student brings wealth knowledge experiences us open eyes new cultures beliefs respect the lim
its language limits world ludwig wittgenstein our english learner strong support system home begs
resources many times parents learning read speak english along side children sometimes creates bar
riers parents able help child learn phonetics letter recognition reading skills by providing dvd p
layers students able continue mastery english language even no one home able assist all families s
tudents within level 1 proficiency status offered part program these educational videos specially
chosen english learner teacher sent home regularly watch the videos help child develop early
reading skills parents not access dvd player opportunity check dvd player use year the plan use vi
deos educational dvd years come el students nannan'
 'our students arrive school eager learn they polite generous strive best they know education succ
eed life help improve lives our school focuses families low incomes tries give student education d
eserve while not much students use materials given best the projector need school crucial academic
improvement students as technology continues grow many resources internet teachers use growth stud
ents however school limited resources particularly technology without disadvantage one things coul
d really help classrooms projector with projector not crucial instruction also growth students wit
h projector show presentations documentaries photos historical land sites math problems much with
projector make teaching learning easier also targeting different types learners classrooms auditor
y visual kinesthetic etc nannan'
 'true champions not always ones win guts by mia hamm this quote best describes students cholla mi
ddle school approach playing sports especially girls boys soccer teams the teams made 7th 8th grad
e students not opportunity play organized sport due family financial difficulties i teach title on
e middle school urban neighborhood 74 students qualify free reduced lunch many come activity sport
opportunity poor homes my students love participate sports learn new skills apart team atmosphere
my school lacks funding meet students needs i concerned lack exposure not prepare participating sp
orts teams high school by end school year goal provide students opportunity learn variety soccer s
kills positive qualities person actively participates team the students campus come school knowing
face uphill battle comes participating organized sports the players would thrive field confidence
appropriate soccer equipment play soccer best abilities the students experience helpful person par
t team teaches positive supportive encouraging others my students using soccer equipment practice
games daily basis learn practice necessary skills develop strong soccer team this experience
create opportunity students learn part team positive contribution teammates the students get oppor
tunity learn practice variety soccer skills use skills game access type experience nearly
impossible without soccer equipment students players utilize practice games nannan'
 ...
 'welcome my students i pleased take interest education they great group kids deserve best as 3rd
grade teacher i strive provide resources needed achieve not students work hard achieve greatness a
lso encourage support better whole my students kind compassionate smart grateful they role models
younger students leaders among peers my students make proud everyday i grateful teacher students c
lass currently sit bounce wiggle jiggle roll exercise balls day enjoy immensely the problem exerci
se balls keep popping we currently less class set exercise balls number keeps decreasing to solve
problem students asked replace exercise balls wobble stools wobble stools reliable way staying act
ive within classroom exercise balls currently every year students learn importance movement
affects focus health future when students learn purpose movement quickly become enthusiastic these
```

```
supplies allow students experience first hand movement affects daily lives nannan'
 'my school empowers 538 students grades pre k five follow dreams 100 students fall poverty line p
rovided free breakfast lunch many students school strive first college graduates families however
many students english language learners thus struggle reading many students come school talk heari
ng gunshots last night parent incarcerated cousin killed drug related incident despite odds i 23 s
cientists mathematicians artists authors readers classroom year my students incredibly special reg
ardless language speak home life like always look one another try best come school smile every day
a 7 year old girl class desperately yearns write read others she smart imagines creative stories p
uts forth best effort however cannot even write name neatly after much research i found penmanship
improved therapy the items i asking allow strengthen hand grip increase visual perception she able
strengthen practice handwriting school home even city bus she not qualify materials school state n
ot considered special education student nannan'
 'we grit if want meet tenacious respectful seven year olds growth mindsets need come classroom we
give hugs high fives compliments we begin end mind work hard everyday reach goals we not believe m
aking excuses times life need ask help as classroom teacher low income high poverty school
district 2nd grade students face real life struggles classroom even though visitor classroom would
not know daily struggle i ask how learn belly growling how i provide absolute best learning enviro
nment not money buy research based materials education not filling pail lighting fire william butl
er yeats we not asking fill pail things help provide resources light fire young minds receiving bo
oks written author teach students develop writer craft it inspire think different ways established
authors developed successful text appeal various audiences we never forget first love my mother re
ad berenstain bears series i five i fell love berenstain family she took public library every week
i would hunt books written stan jan berenstain next curious monkey man yellow hat curious george t
hank margaret h a rey creating series captured heart attention as teacher hope dream inspire stude
nts classroom find first love reading help help discover writer craft go adventures minds develop
tenacious love reading sake reading nannan']
```

In [26]:

```
import re

essay_lst=list(project_data['essay'].values)
essay_count=[]

for i in essay_lst:
    essay_count.append(len(re.findall(r'\w+', i)))


project_data['number_of_essays']=essay_count
```

Number of words in Titles

In [27]:

```
import re

titles_lst=list(project_data['project_title'].values)
titles_count=[]

for i in titles_lst:
    titles_count.append(len(re.findall(r'\w+', i)))


project_data['number_of_titles']=titles_count
```

1. 4 Preparing data for models

In [28]:

```
project_data.columns
```

Out[28]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'sentiment_pos_essay', 'sentiment_neg_essay', 'sentiment_neu_essay',
       'sentiment_compound_essay', 'number_of_essays', 'number_of_titles'],
```

```
      dtype='object')
```

we are going to consider

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

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

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

In [29]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
grades_list = []
for i in grades:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        j = j.replace(' ','_') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Scien
ce"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('-','_') # we are replacing the & value into
        temp = temp.replace('Grades','grades') # we are replacing the & value into
        temp = temp.replace('PreK','prek') # we are replacing the & value into
    grades_list.append(temp.strip())

project_data['project_grade_category'] = grades_list
```

In [30]:

```
y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
project_data.head(1)
```

Out[30]:

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

1 rows × 25 columns

Assignment 8: DT

1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

- Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
- Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
- Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
- Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

2. **Hyper paramter tuning (best `depth` in range [1, 5, 10, 50, 100, 500, 100], and the best `min_samples_split` in range [5, 10, 100, 500])**

   - Find the best hyper parameter which will give the maximum AUC value
   - Find the best hyper paramter using k-fold cross validation or simple cross validation data
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Graphviz**

   - Visualize your decision tree with Graphviz. It helps you to understand how a decision is being made, given a new vector.
   - Since feature names are not obtained from word2vec related models, visualize only BOW & TFIDF decision trees using Graphviz
   - Make sure to print the words in each node of the decision tree instead of printing its index.
   - Just for visualization purpose, limit max_depth to 2 or 3 and either embed the generated images of graphviz in your notebook, or directly upload them as .png files.

4. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure
   - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
   - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points
   - Once after you plot the confusion matrix with the test data, get all the `false positive data points`
     - Plot the WordCloud WordCloud
     - Plot the box plot with the `price` of these `false positive data points`
     - Plot the pdf with the `teacher_number_of_previously_posted_projects` of these `false positive data points`

5. **[Task-2]**

   - Select 5k best features from features of Set 2 using`feature_importances_`, discard all the other remaining features and then apply any of the model of you choice i.e. (Dession tree, Logistic Regression, Linear SVM), you need to do hyperparameter tuning corresponding to the model you selected and procedure in step 2 and step 3

6. **Conclusion**

   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

---

2. Decision Tree

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [31]:

```
X=project_data
```

In [32]:

```
#train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

In [33]:

```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
```

```
print(X_test.shape, y_test.shape)
print("="*100)
```

```
(22445, 25) (22445,)
(11055, 25) (11055,)
(16500, 25) (16500,)
====================================================================================================
```

2.2 Make Data Model Ready: encoding numerical, categorical features

Normalizing the numerical features: Price

In [34]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(-1,1))
X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalizing the numerical features: Previously posted projects

In [35]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_train_ppp_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].va
lues.reshape(-1,1))
X_cv_ppp_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.r
eshape(-1,1))
X_test_ppp_norm =
normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_ppp_norm.shape, y_train.shape)
print(X_cv_ppp_norm.shape, y_cv.shape)
print(X_test_ppp_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalizing the numerical features : Quantity

In [36]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer.fit(X_train['quantity'].values.reshape(-1,1))
X_train_qty_norm = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_qty_norm = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_qty_norm = normalizer.transform(X_test['quantity'].values.reshape(-1,1))
print("After vectorizations")

print(X_train_qty_norm.shape, y_train.shape)
print(X_cv_qty_norm.shape, y_cv.shape)
print(X_test_qty_norm.shape, y_test.shape)

print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalizing the numerical features : Sentinment scores for Postives

In [37]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['sentiment_pos_essay'].values)
normalizer.fit(X_train['sentiment_pos_essay'].values.reshape(-1,1))
X_train_sentpos_norm = normalizer.transform(X_train['sentiment_pos_essay'].values.reshape(-1,1))
X_cv_sentpos_norm = normalizer.transform(X_cv['sentiment_pos_essay'].values.reshape(-1,1))
X_test_sentpos_norm = normalizer.transform(X_test['sentiment_pos_essay'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_sentpos_norm.shape, y_train.shape)
print(X_cv_sentpos_norm.shape, y_cv.shape)
print(X_test_sentpos_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalizing the numerical features : Sentinment scores for Negatives

In [38]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['sentiment_neg_essay'].values)
normalizer.fit(X_train['sentiment_neg_essay'].values.reshape(-1,1))
X_train_sentneg_norm = normalizer.transform(X_train['sentiment_neg_essay'].values.reshape(-1,1))
X_cv_sentneg_norm = normalizer.transform(X_cv['sentiment_neg_essay'].values.reshape(-1,1))
X_test_sentneg_norm = normalizer.transform(X_test['sentiment_neg_essay'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_sentneg_norm.shape, y_train.shape)
print(X_cv_sentneg_norm.shape, y_cv.shape)
```

```
print(X_test_sentneg_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalizing the numerical features : Sentinment scores for Neutral

In [39]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['sentiment_neu_essay'].values)
normalizer.fit(X_train['sentiment_neu_essay'].values.reshape(-1,1))
X_train_sentneu_norm = normalizer.transform(X_train['sentiment_neu_essay'].values.reshape(-1,1))
X_cv_sentneu_norm = normalizer.transform(X_cv['sentiment_neu_essay'].values.reshape(-1,1))
X_test_sentneu_norm = normalizer.transform(X_test['sentiment_neu_essay'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_sentneu_norm.shape, y_train.shape)
print(X_cv_sentneu_norm.shape, y_cv.shape)
print(X_test_sentneu_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalizing the numerical features : Sentinment scores for Compound

In [40]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['sentiment_compound_essay'].values)
normalizer.fit(X_train['sentiment_compound_essay'].values.reshape(-1,1))
X_train_sentcomp_norm = normalizer.transform(X_train['sentiment_compound_essay'].values.reshape(-1,
1))
X_cv_sentcomp_norm = normalizer.transform(X_cv['sentiment_compound_essay'].values.reshape(-1,1))
X_test_sentcomp_norm = normalizer.transform(X_test['sentiment_compound_essay'].values.reshape(-1,1)
)
print("After vectorizations")
print(X_train_sentcomp_norm.shape, y_train.shape)
print(X_cv_sentcomp_norm.shape, y_cv.shape)
print(X_test_sentcomp_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Number of words in the Essay

In [41]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['number_of_essays'].values)
normalizer.fit(X_train['number_of_essays'].values.reshape(-1,1))
X_train_essaynum_norm = normalizer.transform(X_train['number_of_essays'].values.reshape(-1,1))
```

```
X_cv_essaynum_norm = normalizer.transform(X_cv['number_of_essays'].values.reshape(-1,1))
X_test_essaynum_norm = normalizer.transform(X_test['number_of_essays'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_essaynum_norm.shape, y_train.shape)
print(X_cv_essaynum_norm.shape, y_cv.shape)
print(X_test_essaynum_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Number of words in the Title

In [42]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['number_of_titles'].values)
normalizer.fit(X_train['number_of_titles'].values.reshape(-1,1))
X_train_titlenum_norm = normalizer.transform(X_train['number_of_titles'].values.reshape(-1,1))
X_cv_titlenum_norm = normalizer.transform(X_cv['number_of_titles'].values.reshape(-1,1))
X_test_titlenum_norm = normalizer.transform(X_test['number_of_titles'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_titlenum_norm.shape, y_train.shape)
print(X_cv_titlenum_norm.shape, y_cv.shape)
print(X_test_titlenum_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

In [ ]:

```

```

One hot encoding the catogorical features: State

In [43]:

```
vectorizer_state = CountVectorizer()
vectorizer_state.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer_state.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer_state.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer_state.transform(X_test['school_state'].values)
print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer_state.get_feature_names())
print("="*100)

stateVec=vectorizer_state.get_feature_names()
type(stateVec)
```

```
After vectorizations
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv
```

```
', 'wy']
==================================================================================================
```

Out[43]:

```
list
```

One hot encoding the catogorical features: Project Grade

In [44]:

```
vectorizer_grade = CountVectorizer()
vectorizer_grade.fit(X_train['project_grade_category'].values) # fit has to happen only on train
data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer_grade.transform(X_train['project_grade_category'].values)
X_cv_grade_ohe = vectorizer_grade.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe = vectorizer_grade.transform(X_test['project_grade_category'].values)
print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer_grade.get_feature_names())
print("="*100)

projGradeVec=vectorizer_grade.get_feature_names()
```

```
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
==================================================================================================
```

One hot encoding the catogorical features: Teacher Prefix

In [45]:

```
#replacing nan with empty string
X_train.teacher_prefix=X_train.teacher_prefix.fillna('')
X_cv.teacher_prefix=X_cv.teacher_prefix.fillna('')
X_test.teacher_prefix=X_test.teacher_prefix.fillna('')
uniqueData=X_train['teacher_prefix'].unique()
print(uniqueData)

vectorizer_prefix = CountVectorizer(lowercase=False, binary=True)
vectorizer_prefix.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer_prefix.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer_prefix.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer_prefix.transform(X_test['teacher_prefix'].values)
print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer_prefix.get_feature_names())
print("="*100)

prefixteacherVec=vectorizer_prefix.get_feature_names()
```

```
['Mrs.' 'Ms.' 'Teacher' 'Mr.']
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
['Mr', 'Mrs', 'Ms', 'Teacher']
==================================================================================================
```

One hot encoding the catogorical features: Clean categories

In [46]:

```
vectorizer_clean = CountVectorizer()
vectorizer_clean.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_ccat_ohe = vectorizer_clean.transform(X_train['clean_categories'].values)
X_cv_ccat_ohe = vectorizer_clean.transform(X_cv['clean_categories'].values)
X_test_ccat_ohe = vectorizer_clean.transform(X_test['clean_categories'].values)
print("After vectorizations")
print(X_train_ccat_ohe.shape, y_train.shape)
print(X_cv_ccat_ohe.shape, y_cv.shape)
print(X_test_ccat_ohe.shape, y_test.shape)
print(vectorizer_clean.get_feature_names())
print("="*100)

cleanCatVec=vectorizer_clean.get_feature_names()
```

```
After vectorizations
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language',
'math_science', 'music_arts', 'specialneeds', 'warmth']
====================================================================================================
```

One hot encoding the catogorical features: Cleab subcategories

In [47]:

```
vectorizer_subclean = CountVectorizer()
vectorizer_subclean.fit(X_train['clean_subcategories'].values) # fit has to happen only on train
data

# we use the fitted CountVectorizer to convert the text to vector
X_train_csub_ohe = vectorizer_subclean.transform(X_train['clean_subcategories'].values)
X_cv_csub_ohe = vectorizer_subclean.transform(X_cv['clean_subcategories'].values)
X_test_csub_ohe = vectorizer_subclean.transform(X_test['clean_subcategories'].values)
print("After vectorizations")
print(X_train_csub_ohe.shape, y_train.shape)
print(X_cv_csub_ohe.shape, y_cv.shape)
print(X_test_csub_ohe.shape, y_test.shape)
print(vectorizer_subclean.get_feature_names())
print("="*100)

cleansubCatVec=vectorizer_subclean.get_feature_names()
```

```
After vectorizations
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government',
'college_careerprep', 'communityservice', 'earlydevelopment', 'economics', 'environmentalscience',
'esl', 'extracurricular', 'financialliteracy', 'foreignlanguages', 'gym_fitness',
'health_lifescience', 'health_wellness', 'history_geography', 'literacy', 'literature_writing', 'm
athematics', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'performingarts', 'socia
lsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
====================================================================================================
```

2.3 Make Data Model Ready: encoding essay, and project_title

Bag of Words

In [48]:

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_bow = vectorizer_bow_title.transform(X_train['project_title'].values)
X_cv_title_bow = vectorizer_bow_title.transform(X_cv['project_title'].values)
X_test_title_bow = vectorizer_bow_title.transform(X_test['project_title'].values)
print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
print("="*100)

projTitleBowVec=vectorizer_bow_title.get_feature_names()
```

```
After vectorizations
(22445, 1227) (22445,)
(11055, 1227) (11055,)
(16500, 1227) (16500,)
====================================================================================================
```

In [49]:

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(X_train['essay'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer_bow_essay.transform(X_train['essay'].values)
X_cv_essay_bow = vectorizer_bow_essay.transform(X_cv['essay'].values)
X_test_essay_bow = vectorizer_bow_essay.transform(X_test['essay'].values)
print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)


projEssayBowVec=vectorizer_bow_essay.get_feature_names()
```

```
After vectorizations
(22445, 8865) (22445,)
(11055, 8865) (11055,)
(16500, 8865) (16500,)
====================================================================================================
```

TFIDF vectorizer

In [50]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_tfidf = vectorizer_tfidf_title.transform(X_train['project_title'].values)
X_cv_title_tfidf = vectorizer_tfidf_title.transform(X_cv['project_title'].values)
X_test_title_tfidf = vectorizer_tfidf_title.transform(X_test['project_title'].values)
print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print("="*100)

projTitleTfidfVec=vectorizer_tfidf_title.get_feature_names()
```

```
After vectorizations
(22445, 1227) (22445,)
(11055, 1227) (11055,)
(16500, 1227) (16500,)
==========================================================================================
```

In [51]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(X_train['essay'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer_tfidf_essay.transform(X_train['essay'].values)
X_cv_essay_tfidf = vectorizer_tfidf_essay.transform(X_cv['essay'].values)
X_test_essay_tfidf = vectorizer_tfidf_essay.transform(X_test['essay'].values)
print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)


#print(vectorizer.get_feature_names())

projEssayTfidfVec=vectorizer_tfidf_essay.get_feature_names()
```

```
After vectorizations
(22445, 8865) (22445,)
(11055, 8865) (11055,)
(16500, 8865) (16500,)
==========================================================================================
```

2.4 Appling Decision Tree on different kind of featurization as mentioned in the instructions

Apply Decision Tree on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

2.4.1 Applying SVM on BOW, SET 1

In [52]:

```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_bow = hstack((X_train_qty_norm, X_train_ppp_norm, X_train_price_norm, X_train_state_ohe, X_tra
in_grade_ohe,X_train_teacher_ohe,X_train_ccat_ohe,X_train_csub_ohe,X_train_title_bow,X_train_essay_
bow)).tocsr()

X_cr_bow = hstack((X_cv_qty_norm, X_cv_ppp_norm, X_cv_price_norm, X_cv_state_ohe, X_cv_grade_ohe,X_
cv_teacher_ohe,
X_cv_ccat_ohe,X_cv_csub_ohe,X_cv_title_bow,X_cv_essay_bow)).tocsr()

X_te_bow = hstack((X_test_qty_norm, X_test_ppp_norm, X_test_price_norm, X_test_state_ohe,
X_test_grade_ohe,X_test_teacher_ohe,
X_test_ccat_ohe,X_test_csub_ohe,X_test_title_bow,X_test_essay_bow)).tocsr()
print("Final Data matrix")
print(X_tr_bow.shape, y_train.shape)
print(X_cr_bow.shape, y_cv.shape)
print(X_te_bow.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 10193) (22445,)
(11055, 10193) (11055,)
```

(16500, 10193) (16500,)
==========

GraphViz - Decision Tree

In [53]:

```
bow_features_names = []

for a in vectorizer_state.get_feature_names() :
    bow_features_names.append(a)

print(len(bow_features_names))

for a in vectorizer_grade.get_feature_names() :
    bow_features_names.append(a)

print(len(bow_features_names))

for a in vectorizer_prefix.get_feature_names() :
    bow_features_names.append(a)

print(len(bow_features_names))

for a in vectorizer_clean.get_feature_names() :
    bow_features_names.append(a)

print(len(bow_features_names))

for a in vectorizer_subclean.get_feature_names() :
    bow_features_names.append(a)

print(len(bow_features_names))

for a in vectorizer_bow_essay.get_feature_names() :
    bow_features_names.append(a)

print(len(bow_features_names))

for a in vectorizer_bow_title.get_feature_names() :
    bow_features_names.append(a)

print(len(bow_features_names))
```

```
51
55
59
68
98
8963
10190
```

In [54]:

```
bow_features_names.append("price")

bow_features_names.append("teacher_number_of_previously_posted_projects")

bow_features_names.append("quantity")
```

In [55]:

```
len(bow_features_names)
```

Out[55]:

```
10193
```

In [56]:

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=3)
clf = dtree.fit(X_tr_bow, y_train)
```

In [58]:

```
# Visualize data
import graphviz
from sklearn import tree
from graphviz import Source
dot_data = tree.export_graphviz(dtree, out_file=None, feature_names=bow_features_names)
graph = graphviz.Source(dot_data)
graph.render("Bow tree",view = True)
```

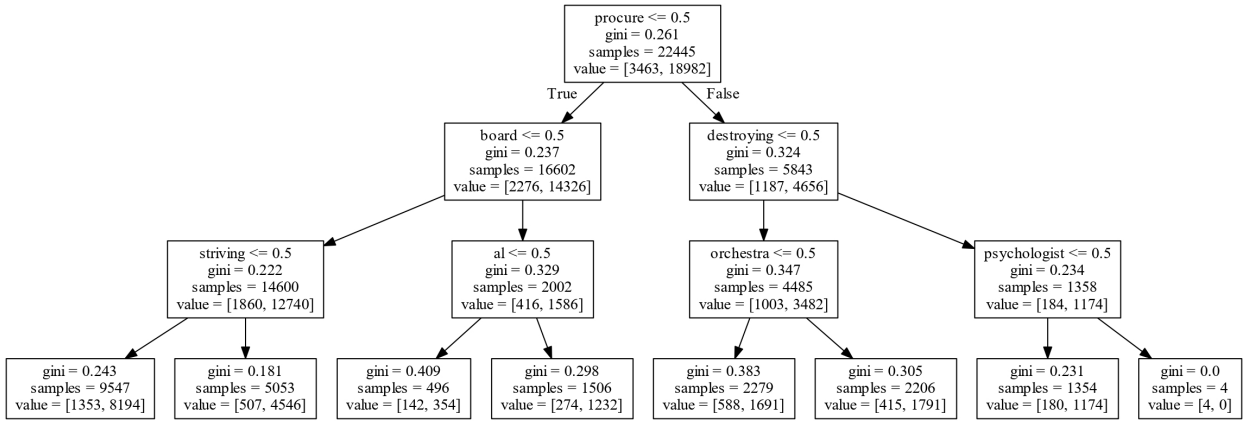
Out[58]:

```
'Bow tree.pdf'
```

In [157]:

```
from IPython.display import Image
#Image("Bow tree_page-0001.jpg")
from IPython.core.display import HTML
Image(url= "https://i.imgur.com/4jjYDpp.jpg")
```

Out[157]:



GridSearchCV (K fold Cross Validation)

In [59]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 100, 500]}

clf = GridSearchCV(dt, parameters, cv= 10, scoring='roc_auc')

clf.fit(X_tr_bow, y_train)
train_auc_bow= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc_bow = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

In [60]:

```
train_auc_bow
```

Out[60]:

```
array([0.54874069, 0.54874069, 0.54874069, 0.54874069, 0.63592269,
       0.63587613, 0.63459441, 0.62850901, 0.74403976, 0.74103169,
       0.71813519, 0.68027919, 0.98124148, 0.97192581, 0.90380867,
       0.7986996 , 0.99574279, 0.99111257, 0.93724947, 0.80984623,
       0.99991143, 0.99834994, 0.95399536, 0.81495549, 0.99991229,
       0.99834351, 0.9544476 , 0.81437822])
```

In [61]:

```
cv_auc_bow
```

Out[61]:

```
array([0.54874006, 0.54874006, 0.54874006, 0.54874006, 0.59816476,
       0.59802256, 0.59835851, 0.59987455, 0.59735608, 0.59441531,
       0.59621818, 0.60246212, 0.55331273, 0.55084899, 0.56831707,
       0.57960223, 0.54044665, 0.54721594, 0.55915897, 0.57695359,
       0.54771044, 0.54299188, 0.55250177, 0.57780519, 0.54603135,
       0.54616789, 0.55458899, 0.57547132])
```

In [62]:

```
print(clf.best_params_)
```

```
{'max_depth': 10, 'min_samples_split': 500}
```

In [63]:

```
depth=[1, 5, 10, 50, 100, 500, 1000]

samples=  [5, 10, 100, 500]
```

In [64]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =train_auc_bow[p]
        p+=1

print(res)
```

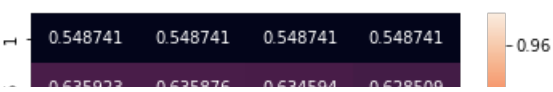
```
[[0.5487406922217154, 0.5487406922217154, 0.5487406922217154, 0.5487406922217154],
[0.6359226925565971, 0.6358761270297626, 0.6345944145720308, 0.6285090063121209],
[0.7440397571314279, 0.7410316901633034, 0.7181351877492221, 0.6802791893103305],
[0.9812414830695548, 0.971925809442558, 0.9038086734916204, 0.7986959573026456],
[0.9957427879442085, 0.9911125706923528, 0.9372494703574707, 0.809846225352915],
[0.9999114265971452, 0.9983499420463298, 0.9539953555588034, 0.8149554889543744],
[0.9999122862009923, 0.9983435132497853, 0.9544476002403112, 0.8143782188652683]]
```

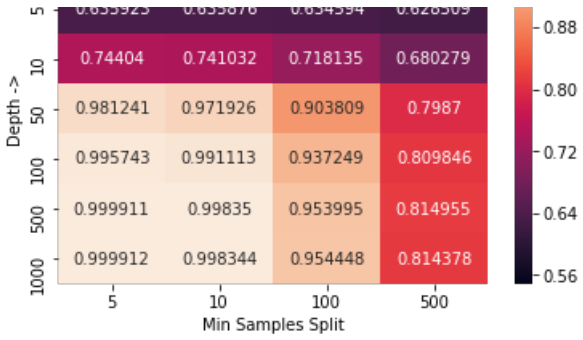
In [65]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[65]:

```
Text(0.5, 15.0, 'Min Samples Split')
```





In [66]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =cv_auc_bow[p]
        p+=1

print(res)
```

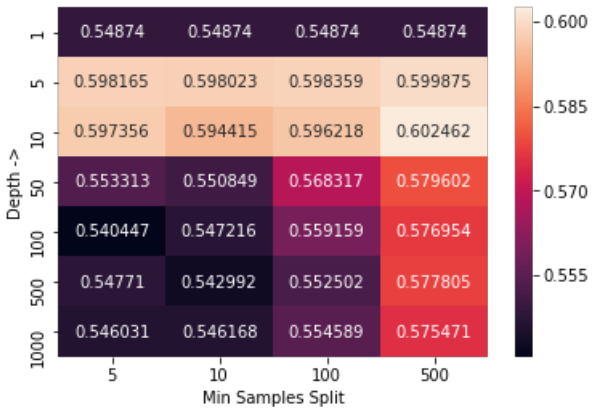
```
[[0.5487400577635687, 0.5487400577635687, 0.5487400577635687, 0.5487400577635687],
[0.598164756019225, 0.5980225631517949, 0.598358510163597, 0.5998745454879655],
[0.5973560808671695, 0.5944153092599588, 0.596218175176886, 0.6024621212062958],
[0.5533127265050518, 0.5508489929683957, 0.5683170735579378, 0.5796022293127928],
[0.5404466518476125, 0.547215941895201, 0.5591589692735554, 0.5769535855214543],
[0.5477104357351315, 0.5429918830366329, 0.5525017708447573, 0.5778051850684923],
[0.5460313539409667, 0.5461678918804023, 0.5545889872384362, 0.5754713188816746]]
```

In [67]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[67]:

```
Text(0.5, 15.0, 'Min Samples Split')
```



Observations :

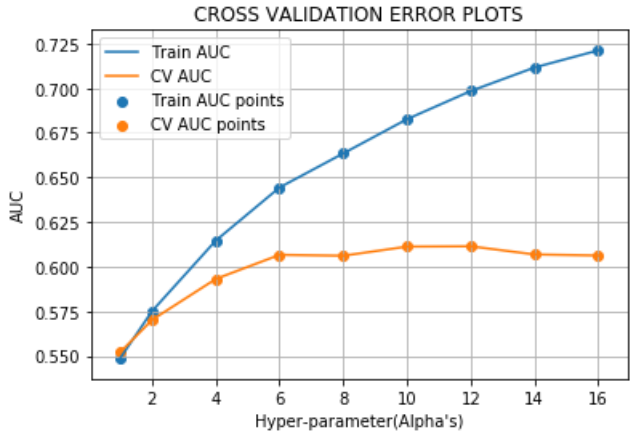
1) Decision Tree with maximum depth between 1 ane 10, performs decently on both Train as well as Cross Validation Data.

2) 500 as the value for Minimum samples per split can be considered from above analysis

2) 500 as the value for minimum samples per split can be considered from above analysis.

In [68]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
import numpy as np
from sklearn import tree
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.
"""
train_auc = []
cv_auc = []
depth = [1, 2, 4, 6, 8, 10, 12,14,16]
for i in depth:
    clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=500)
    clf.fit(X_tr_bow, y_train)
    y_train_pred = clf.predict_proba( X_tr_bow)[:,1]
    y_cv_pred = clf.predict_proba( X_cr_bow)[:,1]
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
plt.plot(depth, train_auc, label='Train AUC')
plt.plot(depth, cv_auc, label='CV AUC')
plt.scatter(depth, train_auc, label='Train AUC points')
plt.scatter(depth, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("Hyper-parameter(Alpha's)")
plt.ylabel("AUC")
plt.title("CROSS VALIDATION ERROR PLOTS")
plt.grid()
plt.show()
```



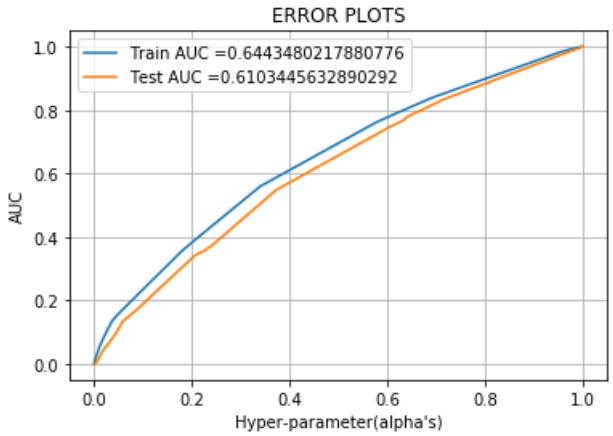
In [73]:

```
i=6
```

In [74]:

```
#https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.r
rve
from sklearn.metrics import roc_curve, auc
from sklearn import tree

clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=500)
clf.fit(X_tr_bow, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
```

```
y_train_pred = clf.predict_proba(X_tr_bow)[:, 1]
y_test_pred = clf.predict_proba(X_te_bow)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Hyper-parameter(alpha's)")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



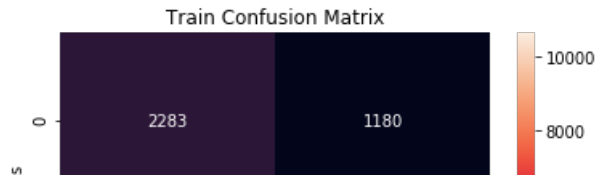
In [75]:

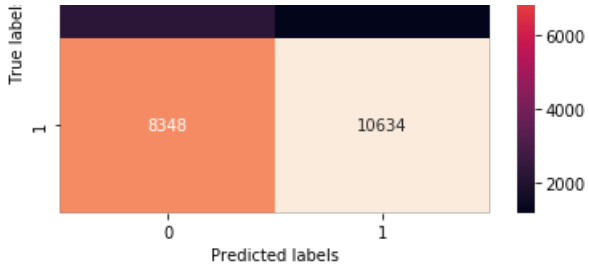
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [76]:

```
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix


print("Train confusion matrix")
a=confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
ax= plt.subplot()
sns.heatmap(a, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.36932449006434437 for threshold 0.558
```

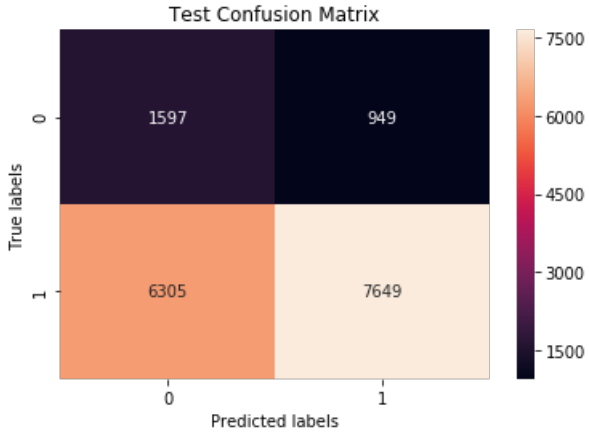




In [77]:

```
import seaborn as sn
import matplotlib.pyplot as plt

print("Test confusion matrix")
b=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))
ax1= plt.subplot()
sns.heatmap(b, annot=True, ax = ax1,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax1.set_xlabel('Predicted labels');
ax1.set_ylabel('True labels');
ax1.set_title('Test Confusion Matrix');
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3438368813881904 for threshold 0.558
```



LET US UNDERSTAND WHY OUR MODEL PERFORMS BAD BY ANALYSING THE FALSE POSITIVE POINTS

Obtaining the False Positive words from BOW encoded Essays

In [78]:

```
bow_test_essay=X_test_essay_bow.todense()
```

In [79]:

```
vectorizer_bow_essay = CountVectorizer(min_df=10)
vec_bow = vectorizer_bow_essay.fit(X_train["essay"])
```

In [80]:

```
bow_features = vec_bow.get_feature_names()
len(bow_features)
```

Out[80]:

```
8865
```

In [81]:

```
y_test_converted = list(y_test[::])
```

In [82]:

```
false_positives_index_a = []
fp_count = 0
for i in tqdm(range(len(y_test_pred))):
    if y_test_converted[i] == 0 and y_test_pred[i] <= 0.558:
        false_positives_index_a.append(i)
        fp_count = fp_count + 1
    else :
        continue
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 485213.60it/s]
```

In [83]:

```
fp_count
```

Out[83]:

```
1931
```

In [84]:

```
false_positives_index_a[0:5]
```

Out[84]:

```
[2, 6, 28, 32, 36]
```

In [85]:

```
df1 = pd.DataFrame(bow_test_essay)
```

In [86]:

```
df1_final = df1.iloc[false_positives_index_a,:]
```

In [87]:

```
df1_final.shape
```

Out[87]:

```
(1931, 8865)
```

In [88]:

```
best_indices = []
for j in range(5000):
    s = df1_final[j].sum()
    if s >= 100 :
        best_indices.append(j)
    else :
        continue
```

In [89]:

```
bow_features[0:10]
```

Out[89]:

```
['00', '000', '10', '100', '1000', '10th', '11', '110', '11th', '12']
```

In [90]:

```
fp_words = []
for a in best_indices :
    fp_words.append(str(bow_features[a]))
```

In [91]:

```
fp_words
```

Out[91]:

```
['100',
 '21st',
 'abilities',
 'ability',
 'able',
 'academic',
 'academically',
 'access',
 'achieve',
 'achievement',
 'active',
 'activities',
 'activity',
 'addition',
 'age',
 'all',
 'allow',
 'allowing',
 'allows',
 'already',
 'also',
 'always',
 'amazing',
 'another',
 'area',
 'areas',
 'around',
 'art',
 'arts',
 'as',
 'ask',
 'asking',
 'at',
 'attend',
 'attention',
 'autism',
 'available',
 'back',
 'backgrounds',
 'based',
 'basic',
 'basis',
 'become',
 'begin',
 'behavior',
 'believe',
 'benefit',
 'best',
 'better',
 'beyond',
 'big',
 'book',
 'books',
 'breakfast',
 'bring',
 'build',
 'building',
 'by'
```

```
'by',
'cannot',
'care',
'career',
'center',
'centers',
'century',
'chairs',
'challenge',
'challenges',
'challenging',
'chance',
'change',
'child',
'children',
'choice',
'choices',
'choose',
'city',
'class',
'classes',
'classroom',
'classrooms',
'college',
'come',
'comes',
'comfortable',
'coming',
'communication',
'community',
'complete',
'computer',
'computers',
'concepts',
'confidence',
'content',
'continue',
'control',
'core',
'could',
'create',
'creating',
'creative',
'creativity',
'critical',
'curious',
'currently',
'curriculum',
'daily',
'day',
'deserve',
'design',
'desire',
'desks',
'despite',
'develop',
'difference',
'different',
'difficult',
'disabilities',
'district',
'diverse',
'donation',
'donations',
'due',
'each',
'eager',
'early',
'economic',
'education',
'educational',
'elementary',
'emotional',
'encourage',
'end',
'energetic',
'energy',
'engage',
```

```
'engage',
'engaged',
'engagement',
'engaging',
'engineering',
'english',
'enhance',
'enjoy',
'enough',
'environment',
'equipment',
'especially',
'essential',
'even',
'ever',
'every',
'everyday',
'everyone',
'everything',
'excited',
'exciting',
'experience',
'experiences',
'explore',
'express',
'extra',
'extremely',
'face',
'faced',
'families',
'family',
'feel',
'find',
'first',
'flexible',
'focus',
'focused',
'food',
'for',
'forward',
'foster',
'free',
'full',
'fun',
'funding',
'future',
'games',
'get',
'getting',
'give',
'given',
'giving',
'go',
'goal',
'goals',
'going',
'good',
'grade',
'graders',
'grades',
'great',
'greatly',
'group',
'groups',
'grow',
'growing',
'growth',
'hand',
'hands',
'hard',
'having',
'health',
'healthy',
'help',
'helping',
'helps',
'high',
'home',
```

```
 'nome',
 'homes',
 'hope',
 'however',
 'ideas',
 'if',
 'impact',
 'important',
 'improve',
 'in',
 'income',
 'increase',
 'independent',
 'individual',
 'information',
 'inspire',
 'instruction',
 'interactive',
 'interest',
 'ipad',
 'it',
 'items',
 'job',
 'keep',
 'kids',
 'kind',
 'kindergarten',
 'know',
 'knowledge',
 'lack',
 'language',
 'large',
 'last',
 'learn',
 'learners',
 'learning',
 'lesson',
 'lessons',
 'let',
 'level',
 'levels',
 'library',
 'life',
 'like',
 'limited',
 'literacy',
 'little',
 'live',
 'lives',
 'living',
 'located',
 'long',
 'look',
 'looking',
 'lot',
 'love',
 'low',
 'lunch',
 'made',
 'majority',
 'make',
 'makes',
 'making',
 'manipulatives',
 'many',
 'materials',
 'math',
 'may',
 'meaningful',
 'means']
```

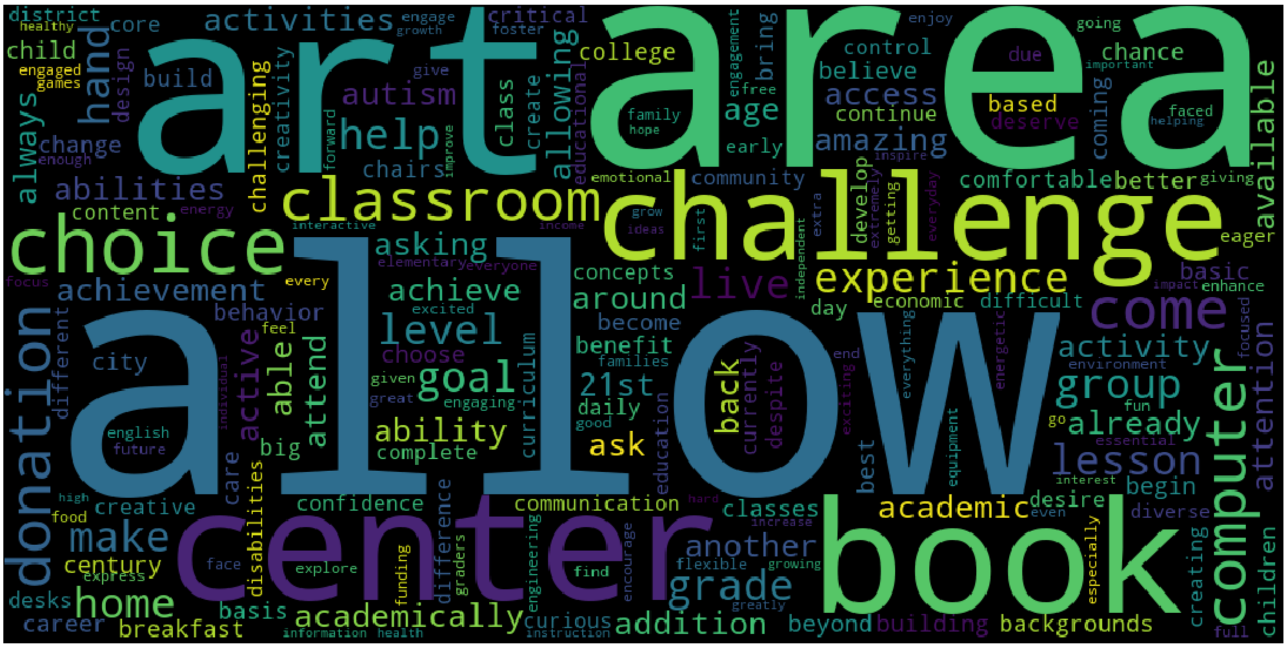
Word Cloud for False Positives words

In [92]:

```python
from wordcloud import WordCloud
```

In [93]:

```python
#convert list to string and generate
unique_string=(" ").join(fp_words)
wordcloud = WordCloud(width = 1000, height = 500).generate(unique_string)
plt.figure(figsize=(25,10))
plt.imshow(wordcloud)
plt.axis("off")
plt.savefig("your_file_name"+".png", bbox_inches='tight')
plt.show()
plt.close()
```



Box - Plot with the price of these False positive data points

In [94]:

```python
len(false_positives_index_a)
```

Out[94]:

1931

In [95]:

```python
df2 = pd.DataFrame(X_test['price'])
```

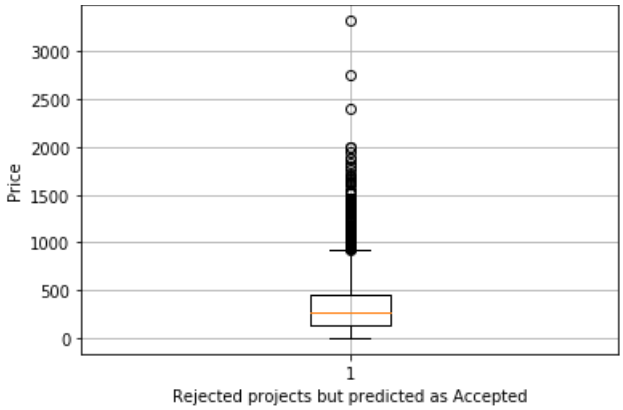
In [96]:

```python
df2_final = df2.iloc[false_positives_index_a,:]
```

In [97]:

```python
plt.boxplot(df2_final.values)
plt.title('Box Plots of Cost per Rejected Project that got predicted as Accepted')
plt.xlabel('Rejected projects but predicted as Accepted')
plt.ylabel('Price')
plt.grid()
plt.show()
```

Box Plots of Cost per Rejected Project that got predicted as Accepted



Inference

1) Majority of the projects that were rejected but predicted as accepted Costs almost less than 500 Dollars.

2) A Few of them are Extremely costs costing more than 3000 Dollars.

2.4.2 Applying SVM on TFIDF, SET 2

In [98]:

```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_tfidf = hstack((X_train_qty_norm, X_train_ppp_norm, X_train_price_norm, X_train_state_ohe, X_t
rain_grade_ohe,X_train_teacher_ohe,
X_train_ccat_ohe,X_train_csub_ohe,X_train_title_tfidf,X_train_essay_tfidf)).tocsr()

X_cr_tfidf = hstack((X_cv_qty_norm, X_cv_ppp_norm, X_cv_price_norm, X_cv_state_ohe, X_cv_grade_ohe,
X_cv_teacher_ohe,
X_cv_ccat_ohe,X_cv_csub_ohe,X_cv_title_tfidf,X_cv_essay_tfidf)).tocsr()

X_te_tfidf = hstack((X_test_qty_norm, X_test_ppp_norm, X_test_price_norm, X_test_state_ohe,
X_test_grade_ohe,X_test_teacher_ohe,
X_test_ccat_ohe,X_test_csub_ohe,X_test_title_tfidf,X_test_essay_tfidf)).tocsr()
print("Final Data matrix")
print(X_tr_tfidf.shape, y_train.shape)
print(X_cr_tfidf.shape, y_cv.shape)
print(X_te_tfidf.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 10193) (22445,)
(11055, 10193) (11055,)
(16500, 10193) (16500,)
====================================================================================================
```

GraphViz - Decision Tree

In [99]:

```
tfidf_features_names = []

for a in vectorizer_state.get_feature_names() :
    tfidf_features_names.append(a)

print(len(tfidf_features_names))

for a in vectorizer_grade.get_feature_names() :
    tfidf_features_names.append(a)

print(len(tfidf_features_names))
```

```
for a in vectorizer_prefix.get_feature_names() :
    tfidf_features_names.append(a)

print(len(tfidf_features_names))

for a in vectorizer_clean.get_feature_names() :
    tfidf_features_names.append(a)

print(len(tfidf_features_names))

for a in vectorizer_subclean.get_feature_names() :
    tfidf_features_names.append(a)

print(len(tfidf_features_names))

for a in vectorizer_tfidf_essay.get_feature_names() :
    tfidf_features_names.append(a)

print(len(tfidf_features_names))

for a in vectorizer_tfidf_title.get_feature_names() :
    tfidf_features_names.append(a)

print(len(tfidf_features_names))
```

```
51
55
59
68
98
8963
10190
```

In [100]:

```
tfidf_features_names.append("price")

tfidf_features_names.append("teacher_number_of_previously_posted_projects")

tfidf_features_names.append("quantity")
```

In [101]:

```
len(tfidf_features_names)
```

Out[101]:

```
10193
```

In [102]:

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=3)
clf = dtree.fit(X_tr_tfidf, y_train)
```

In [105]:

```
# Visualize data
import graphviz
from sklearn import tree
from graphviz import Source
dot_data = tree.export_graphviz(dtree, out_file=None, feature_names=tfidf_features_names)
graph = graphviz.Source(dot_data)
graph.render("TFIDF tree",view = True)
```

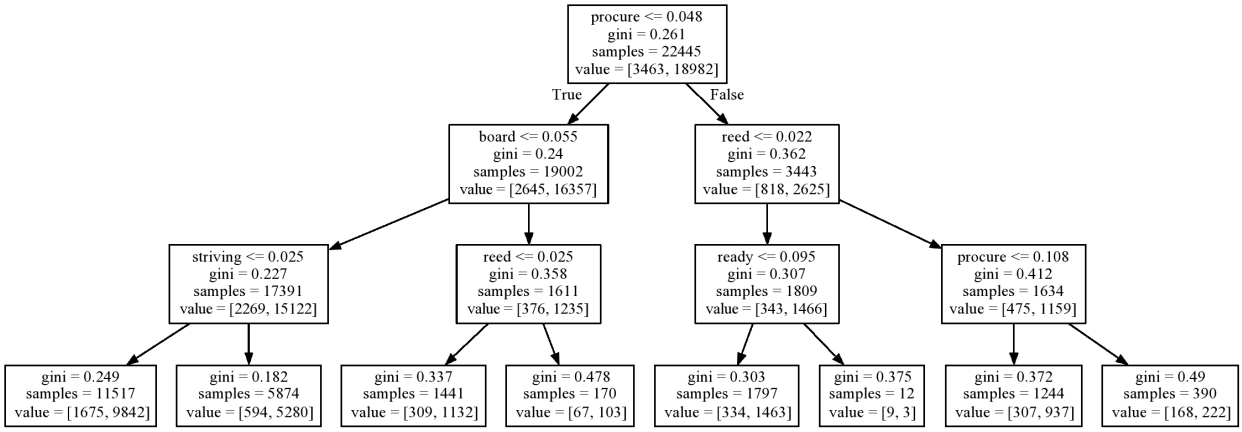
Out[105]:

```
'TFIDF tree.pdf'
```

In [158]:

```
from IPython.display import Image
#Image("Bow tree_page-0001.jpg")
from IPython.core.display import HTML
Image(url= "https://i.imgur.com/c7jQIRg.jpg")
```

Out[158]:



GridSearchCV (K fold Cross Validation)

In [106]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 100, 500]}

clf = GridSearchCV(dt, parameters, cv= 10, scoring='roc_auc')

clf.fit(X_tr_tfidf, y_train)
train_auc_tfidf= clf.cv_results_['mean_train_score']

cv_auc_tfidf = clf.cv_results_['mean_test_score']
```

In [112]:

```
clf.best_params_
```

Out[112]:

```
{'max_depth': 10, 'min_samples_split': 500}
```

In [107]:

```
depth=[1, 5, 10, 50, 100, 500, 1000]

samples=  [5, 10, 100, 500]
```

In [108]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =train_auc_tfidf[p]
        p+=1

print(res)
```

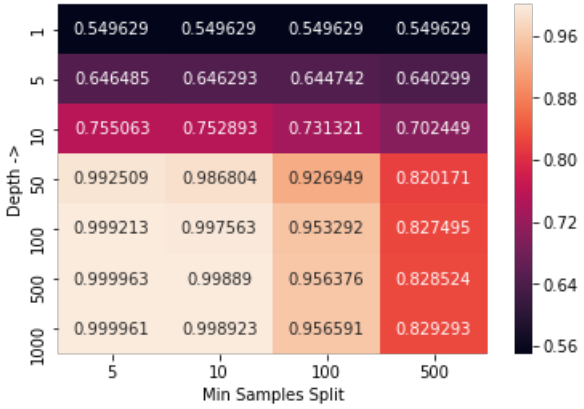
```
[[0.5496291956441629, 0.5496291956441629, 0.5496291956441629, 0.5496291956441629],
[0.6464852038826885, 0.6462928002045617, 0.6447417270456832, 0.6402989078630995],
[0.7550629855287002, 0.7528934932411139, 0.7313214821087928, 0.7024488879597505],
[0.9925085027121963, 0.9868037236441678, 0.9269486797039489, 0.8201709884002473],
[0.9992125368109883, 0.9975633667311525, 0.9532918863690265, 0.827495174351063],
[0.9999633283301945, 0.9988904077273881, 0.9563756402131703, 0.8285240118265985],
[0.9999607376172364, 0.9989228241669027, 0.9565908846704273, 0.8292932130454979]]
```

In [109]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[109]:

```
Text(0.5, 15.0, 'Min Samples Split')
```



In [110]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =cv_auc_tfidf[p]
        p+=1

print(res)
```

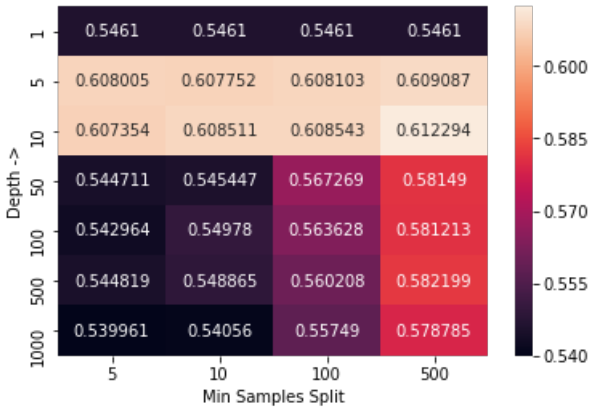
```
[[0.546099862212307, 0.546099862212307, 0.546099862212307, 0.546099862212307],
[0.6080052299261302, 0.6077516823391079, 0.6081034174914978, 0.609087243975394],
[0.6073536731177599, 0.6085105181762509, 0.6085433275743266, 0.6122943411058667],
[0.5447112786977637, 0.5454468674462515, 0.5672688732011799, 0.5814903426386664],
[0.5429635084270313, 0.5497795177997035, 0.5636278332569563, 0.5812134783479503],
[0.5448192035192608, 0.5488645502677327, 0.5602082162566865, 0.5821992805860293],
[0.5399610809582793, 0.5405596591147176, 0.5574901700204726, 0.5787849451076227]]
```

In [111]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[111]:

```
Text(0.5, 15.0, 'Min Samples Split')
```



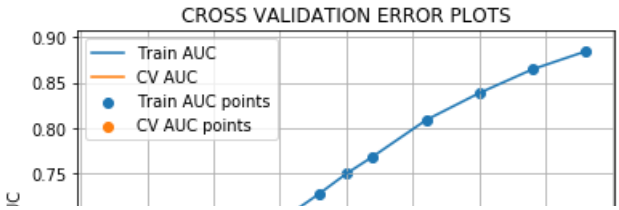
Observations :

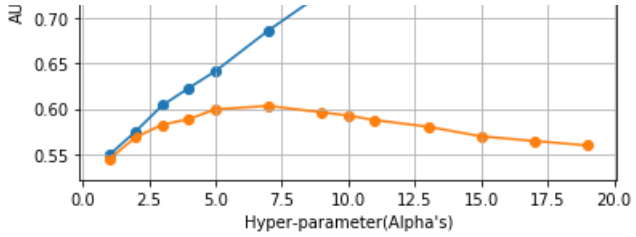
1) Decision Tree with maximum depth between 1 ane 10, performs decently on both Train as well as Cross Validation Data.

2) 500 as the value for Minimum samples per split can be considered from above analysis.

In [113]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
import numpy as np
from sklearn import tree
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.
"""
train_auc = []
cv_auc = []
depth = [1, 2, 3, 4, 5, 7, 9,10,11,13,15,17,19]
for i in depth:
    clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=10)
    clf.fit(X_tr_tfidf, y_train)
    y_train_pred = clf.predict_proba( X_tr_tfidf)[:,1]
    y_cv_pred = clf.predict_proba( X_cr_tfidf)[:,1]
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
plt.plot(depth, train_auc, label='Train AUC')
plt.plot(depth, cv_auc, label='CV AUC')
plt.scatter(depth, train_auc, label='Train AUC points')
plt.scatter(depth, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("Hyper-parameter(Alpha's)")
plt.ylabel("AUC")
plt.title("CROSS VALIDATION ERROR PLOTS")
plt.grid()
plt.show()
```



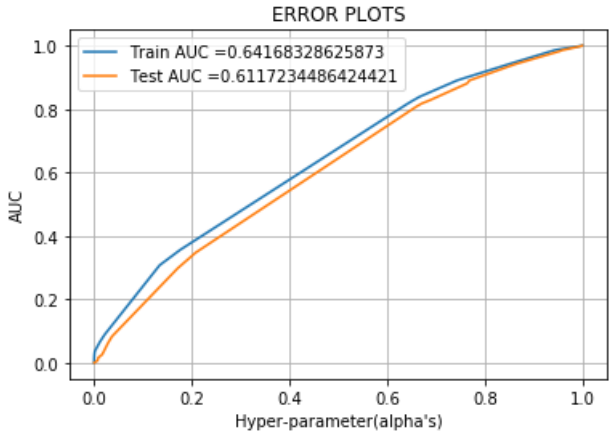


In [120]:

```
i=5
```

In [121]:

```
#https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.r
rve
from sklearn.metrics import roc_curve, auc
from sklearn import tree

clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=10)
clf.fit(X_tr_tfidf, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X_tr_tfidf)[:, 1]
y_test_pred = clf.predict_proba(X_te_tfidf)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Hyper-parameter(alpha's)")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



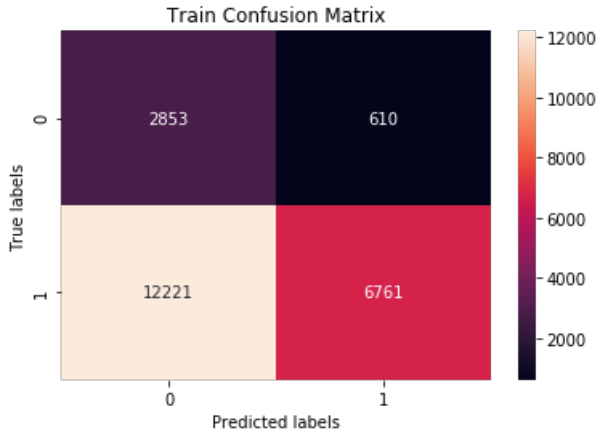
In [122]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [123]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
c=confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))

ax= plt.subplot()
sns.heatmap(c, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

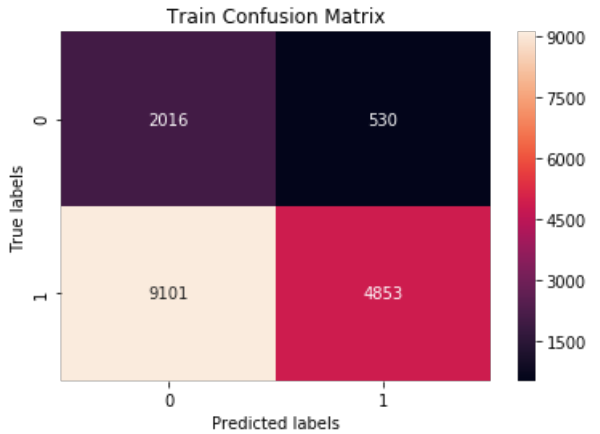
```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2934392790555899 for threshold 0.535
```



In [124]:

```
print("Test confusion matrix")
d=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))


ax= plt.subplot()
sns.heatmap(d, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.27538716877055697 for threshold 0.535
```



LET US UNDERSTAND WHY OUR MODEL PERFORMS BAD BY ANALYSING THE FALSE POSITIVE POINTS

In [133]:

```
tfidf_test = X_test_essay_tfidf.todense()
```

In [134]:

```
tfidf_test.shape
```

Out[134]:

```
(16500, 8865)
```

In [135]:

```
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
bv = vectorizer_tfidf_essay.fit(X_train["essay"])
```

In [136]:

```
tfidf_features = bv.get_feature_names()
len(tfidf_features)
```

Out[136]:

```
8865
```

In [137]:

```
y_test_converted = list(y_test[::])
```

In [138]:

```
false_positives_index_b = []
fp_count = 0

for i in tqdm(range(len(y_test_pred))):
    if y_test_converted[i] == 0 and y_test_pred[i] <= 0.535:
        false_positives_index_b.append(i)
        fp_count = fp_count + 1
    else :
        continue
```

```
100%|██████████████████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 335539.44it/s]
```

In [139]:

```
fp_count
```

Out[139]:

```
2016
```

In [140]:

```
false_positives_index_b[0:5]
```

Out[140]:

```
[2, 6, 28, 32, 36]
```

In [141]:

```
df2 = pd.DataFrame(tfidf_test)
```

In [142]:

```
df2_final = df2.iloc[false_positives_index_b,:]
```

In [143]:

```
df2_final.shape
```

Out[143]:

```
(2016, 8865)
```

In [144]:

```
best_indices_b = []

for j in range(5000):

    s = df2_final[j].sum()

    if s >= 10 :
        best_indices_b.append(j)
    else :
        continue
```

In [145]:

```
len(best_indices_b)
```

Out[145]:

```
199
```

In [146]:

```
fp_words_b = []

for a in best_indices_b :
    fp_words_b.append(str(tfidf_features[a]))
```

In [147]:

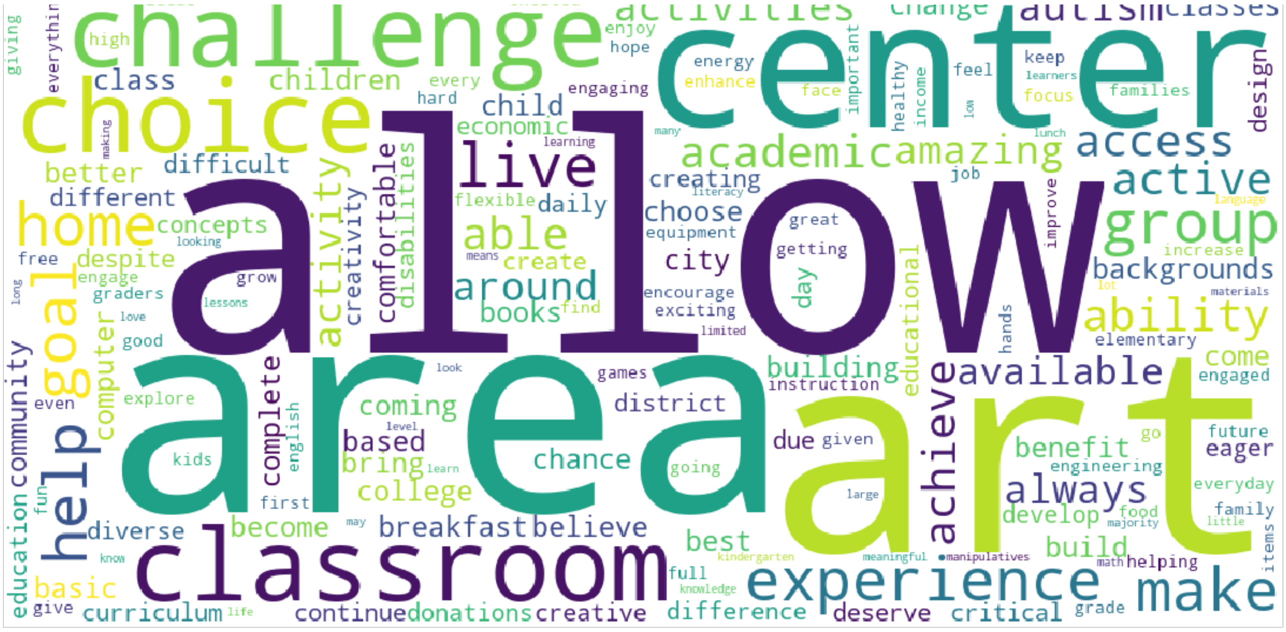
```
fp_words_b = []

for a in best_indices_b :
    fp_words_b.append(str(tfidf_features[a]))
```

In [148]:

```
#  Word Cloud for False Positives words
```

In [149]:

```
#convert list to string and generate
unique_string=(" ").join(fp_words_b)
wordcloud = WordCloud(width = 1000, height = 500, background_color ='white').generate(unique_string
)
plt.figure(figsize=(25,10))
plt.imshow(wordcloud)
plt.axis("off")
plt.savefig("Word_Cloud_tfidf"+".png", bbox_inches='tight')
plt.show()
plt.close()
```



Box - Plot with the price of these False positive data points

In [150]:

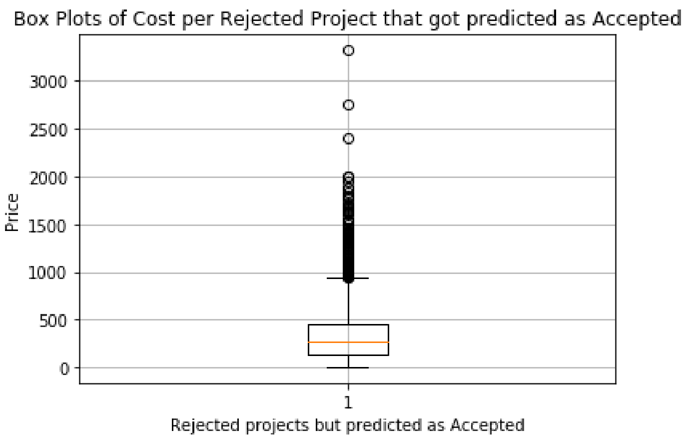
```
df2_b = pd.DataFrame(X_test['price'])
```

In [151]:

```
df2_b_final = df2_b.iloc[false_positives_index_b,:]
```

In [152]:

```
plt.boxplot(df2_b_final.values)
plt.title('Box Plots of Cost per Rejected Project that got predicted as Accepted')
plt.xlabel('Rejected projects but predicted as Accepted')
plt.ylabel('Price')
plt.grid()
plt.show()
```



Observations

1) Majority of the projects that were rejected but predicted as accepted Costs almost less than 500 Dollars.

2) A Few of them are Extremely costs costing more than 2000 Dollars.

2.4.3 Applying DT on AVG W2V, SET 3

In [160]:

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

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

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

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

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

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

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


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

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

```
Loading Glove Model
```

```
1917495it [12:30, 2555.36it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 97525
the unique words in the coupus 8022
The number of words that are present in both glove vectors and our coupus 7782 ( 97.008 %)
word 2 vec length 7782
```

In [161]:

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

```
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [162]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_train_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_train_title.append(vector)

print(len(avg_w2v_train_title))
print(len(avg_w2v_train_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445
[00:01<00:00, 12878.11it/s]
```

```
22445
300
```

In [163]:

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

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

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

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

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

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

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

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

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

```
Loading Glove Model
```

```
1917495it [10:02, 3180.81it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 48155
the unique words in the coupus 5687
The number of words that are present in both glove vectors and our coupus 5559 ( 97.749 %)
word 2 vec length 5559
```

In [164]:

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

In [165]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_cv_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_cv_title.append(vector)

print(len(avg_w2v_cv_title))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 11055/11055 [00:01<00:00, 10040.36it/s]
```

```
11055
```

In [166]:

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

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

Loading Glove Model
1917495it [06:32, 4870.60it/s]
```

```
1917495it [06:52, 4879.69it/s]
Done. 1917495  words loaded!
'''
# ============================

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

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

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

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


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

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

```
Loading Glove Model
```

```
1917495it [12:57, 2467.59it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 71409
the unique words in the coupus 6977
The number of words that are present in both glove vectors and our coupus 6772 ( 97.062 %)
word 2 vec length 6772
```

In [167]:

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

In [168]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_test_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_test_title.append(vector)

print(len(avg_w2v_test_title))
print(len(avg_w2v_test_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 16500/16500
```

```
[00:01<00:00, 13334.00it/s]
```

```
16500
300
```

In [169]:

```
# Similarly you can vectorize for title also

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

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

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

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

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

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

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


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

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

```
Loading Glove Model
```

```
1917495it [12:10, 2625.92it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 3398118
the unique words in the coupus 30470
The number of words that are present in both glove vectors and our coupus 28812 ( 94.559 %)
word 2 vec length 28812
```

In [170]:

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

In [171]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_train_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_train_essay.append(vector)

print(len(avg_w2v_train_essay))
print(len(avg_w2v_train_essay[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445 [00:21<00:00, 1022.85it/s]
```

```
22445
300
```

In [172]:

```
# Similarly you can vectorize for title also

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

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

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

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

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

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

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


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

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

```
Loading Glove Model
```

```
1917495it [11:42, 2730.44it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 1670077
the unique words in the coupus 23434
The number of words that are present in both glove vectors and our coupus 22504 ( 96.031 %)
word 2 vec length 22504
```

In [173]:

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

In [174]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_cv_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_cv_essay.append(vector)

print(len(avg_w2v_cv_essay))
print(len(avg_w2v_cv_essay[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 11055/11055 [00:11<00:00, 971.93it/s]
```

```
11055
300
```

In [175]:

```
# Similarly you can vectorize for title also

# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
```

```
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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

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

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


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

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

```
Loading Glove Model
```

```
1917495it [13:08, 2431.40it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 2493267
the unique words in the coupus 27229
The number of words that are present in both glove vectors and our coupus 25900 ( 95.119 %)
word 2 vec length 25900
```

In [176]:

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

In [177]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_test_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
```

```
            if word in glove_words:
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_test_essay.append(vector)

print(len(avg_w2v_test_essay))
print(len(avg_w2v_test_essay[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 16500/16500
[00:16<00:00, 1010.06it/s]
```

```
16500
300
```

In [178]:

```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_avg_w2v = hstack((X_train_qty_norm.T, X_train_ppp_norm.T, X_train_price_norm.T, X_train_state_
ohe, X_train_grade_ohe,X_train_teacher_ohe,
X_train_ccat_ohe,X_train_csub_ohe,avg_w2v_train_title,avg_w2v_train_essay)).tocsr()

X_cv_avg_w2v = hstack((X_cv_qty_norm.T, X_cv_ppp_norm.T, X_cv_price_norm.T, X_cv_state_ohe, X_cv_gr
ade_ohe,X_cv_teacher_ohe,
X_cv_ccat_ohe,X_cv_csub_ohe,avg_w2v_cv_title,avg_w2v_cv_essay)).tocsr()

X_te_avg_w2v = hstack((X_test_qty_norm.T, X_test_ppp_norm.T, X_test_price_norm.T, X_test_state_ohe,
X_test_grade_ohe,X_test_teacher_ohe,
X_test_ccat_ohe,X_test_csub_ohe,avg_w2v_test_title,avg_w2v_test_essay)).tocsr()

print("Final Data matrix")
print(X_tr_avg_w2v.shape, y_train.shape)
print(X_cv_avg_w2v.shape, y_cv.shape)
print(X_te_avg_w2v.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 701) (22445,)
(11055, 701) (11055,)
(16500, 701) (16500,)
====================================================================================================
```

In [179]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 100, 500]}

clf = GridSearchCV(dt, parameters, cv= 10, scoring='roc_auc')

clf.fit(X_tr_avg_w2v, y_train)
train_auc_avg_w2v= clf.cv_results_['mean_train_score']

cv_auc_avg_w2v = clf.cv_results_['mean_test_score']
```

In [180]:

```
train_auc_avg_w2v
```

Out[180]:

```
array([0.57118602, 0.57118602, 0.57118602, 0.57118602, 0.68947289,
       0.68941844, 0.68862288, 0.68741172, 0.79405228, 0.79137469,
       0.76288106, 0.73590585, 0.99703455, 0.99363646, 0.94237542,
       0.85117218, 0.99954409, 0.99656461, 0.94590435, 0.85574936,
       0.9995437 , 0.99656488, 0.94693908, 0.85390979, 0.9995359 ,
       0.99662028, 0.94654834, 0.85444327])
```

In [181]:

```
cv_auc_avg_w2v
```

Out[181]:

```
array([0.56767377, 0.56767377, 0.56767377, 0.56767377, 0.64580562,
       0.6455958 , 0.6447989 , 0.64450313, 0.59856508, 0.60072908,
       0.62211589, 0.64193214, 0.50398724, 0.5092294 , 0.57048435,
       0.61444959, 0.54297757, 0.54549579, 0.58425604, 0.62160264,
       0.54574315, 0.54582398, 0.57840926, 0.61875934, 0.54354527,
       0.54636903, 0.58356341, 0.61835296])
```

In [183]:

```
clf.best_params_
```

Out[183]:

```
{'max_depth': 5, 'min_samples_split': 5}
```

In [184]:

```
depth=[1, 5, 10, 50, 100, 500, 1000]

samples=  [5, 10, 100, 500]
```

In [185]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =train_auc_avg_w2v[p]
        p+=1

print(res)
```

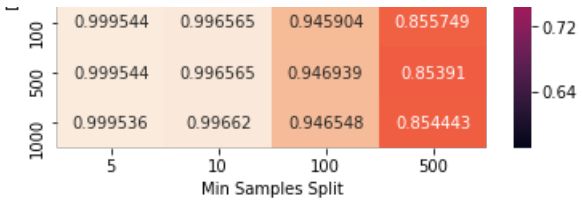
```
[[0.5711860159478361, 0.5711860159478361, 0.5711860159478361, 0.5711860159478361],
[0.6894728885098924, 0.6894184389515917, 0.6886228842009219, 0.6874117150569259],
[0.794052283598353, 0.7913746921960636, 0.7628810554251607, 0.7359058472489979],
[0.9970345535844084, 0.9936364562351576, 0.9423754196661337, 0.8511721836980304],
[0.9995440898653328, 0.996564605088673, 0.945904345129434, 0.8557493625342125],
[0.9995436977148969, 0.9965648751686353, 0.9469390818052548, 0.8539097854574436],
[0.9995358966472552, 0.9966202786165705, 0.9465483435635736, 0.8544432730444317]]
```

In [186]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[186]:

```
Text(0.5, 15.0, 'Min Samples Split')
```



In [187]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =cv_auc_avg_w2v[p]
        p+=1

print(res)
```

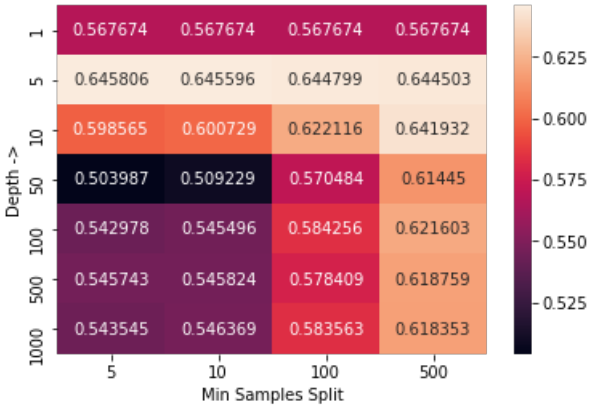
```
[[0.5676737718268385, 0.5676737718268385, 0.5676737718268385, 0.5676737718268385],
[0.645805616984862, 0.6455958048600782, 0.6447989003814679, 0.6445031308941184],
[0.598565076613451, 0.6007290764397939, 0.6221158905148322, 0.6419321414996204],
[0.5039872435746646, 0.5092293983409422, 0.5704843547528884, 0.6144495936869323],
[0.5429775678485022, 0.5454957877669285, 0.5842560358933279, 0.6216026395557471],
[0.5457431526032458, 0.5458239820335745, 0.5784092642331677, 0.6187593361352076],
[0.5435452707512594, 0.5463690285161655, 0.5835634054327219, 0.6183529583142704]]
```

In [188]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[188]:

```
Text(0.5, 15.0, 'Min Samples Split')
```



Observations :

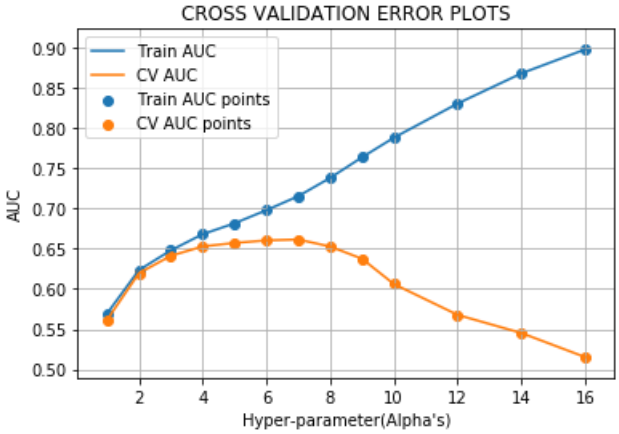
1) Decision Tree with maximum depth between 1 ane 10, performs decently on both Train as well as Cross Validation Data.

2) 5 as the value for Minimum samples per split can be considered from above analysis.

In [189]:

```
import matplotlib.pyplot as plt
```

```
from sklearn.metrics import roc_auc_score
import numpy as np
from sklearn import tree
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.
"""
train_auc = []
cv_auc = []
depth = [1, 2, 3, 4, 5, 6, 7,8,9,10,12,14,16]
for i in depth:
    clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=5)
    clf.fit(X_tr_avg_w2v, y_train)
    y_train_pred = clf.predict_proba( X_tr_avg_w2v)[:,1]
    y_cv_pred = clf.predict_proba( X_cv_avg_w2v)[:,1]
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
plt.plot(depth, train_auc, label='Train AUC')
plt.plot(depth, cv_auc, label='CV AUC')
plt.scatter(depth, train_auc, label='Train AUC points')
plt.scatter(depth, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("Hyper-parameter(Alpha's)")
plt.ylabel("AUC")
plt.title("CROSS VALIDATION ERROR PLOTS")
plt.grid()
plt.show()
```



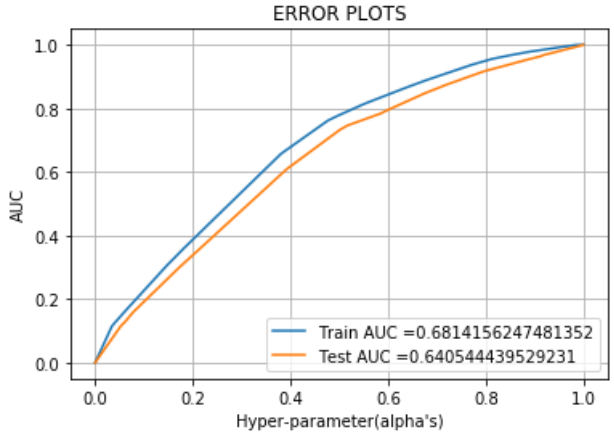
In [194]:

```
i=5
```

In [195]:

```
#https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.r
rve
from sklearn.metrics import roc_curve, auc
from sklearn import tree

clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=5)
clf.fit(X_tr_avg_w2v, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X_tr_avg_w2v)[:, 1]
y_test_pred = clf.predict_proba(X_te_avg_w2v)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

```
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Hyper-parameter(alpha's)")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



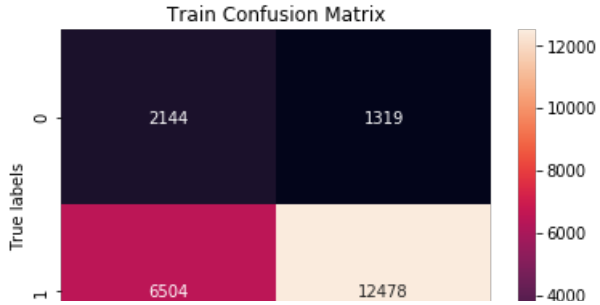
In [196]:

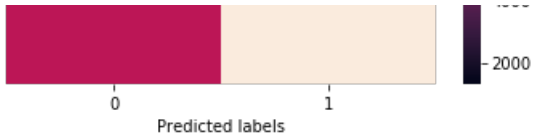
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [197]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
c=confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))

ax= plt.subplot()
sns.heatmap(c, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4069820937403105 for threshold 0.891
```

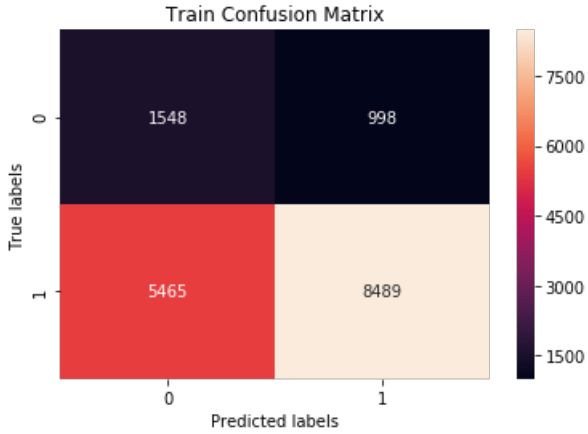




In [198]:

```
print("Test confusion matrix")
d=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))


ax= plt.subplot()
sns.heatmap(d, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3698881106488258 for threshold 0.891
```



In [ ]:

```

```

2.4.4 Applying DT on TFIDF W2V, SET 4

In [205]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [206]:

```
# average Word2Vec
# compute average word2vec for each review.
X_train_tfidf_w2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
```

```
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_tfidf_w2v_title.append(vector)

print(len(X_train_tfidf_w2v_title))
```

```
100%|████████████████████████████████████████████████████████████████| 22445/22445
[00:05<00:00, 3805.03it/s]
```

```
22445
```

In [207]:

```
# average Word2Vec
# compute average word2vec for each review.
X_cv_tfidf_w2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_cv_tfidf_w2v_title.append(vector)

print(len(X_cv_tfidf_w2v_title))
```

```
100%|████████████████████████████████████████████████████████████████| 11055/11055
[00:02<00:00, 5347.21it/s]
```

```
11055
```

In [208]:

```
# average Word2Vec
# compute average word2vec for each review.
X_test_tfidf_w2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_tfidf_w2v_title.append(vector)

print(len(X_test_tfidf_w2v_title))
```

```
100%|████████████████████████████████████████████████████████████████| 16500/16500
[00:02<00:00, 6018.83it/s]
```

```
16500
```

In [209]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [210]:

```
# average Word2Vec
# compute average word2vec for each review.
X_train_tfidf_w2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_tfidf_w2v_essay.append(vector)

print(len(X_train_tfidf_w2v_essay))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445 [02:
25<00:00, 153.82it/s]
```

```
22445
```

In [211]:

```
# average Word2Vec
# compute average word2vec for each review.
X_cv_tfidf_w2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_cv_tfidf_w2v_essay.append(vector)

print(len(X_cv_tfidf_w2v_essay))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11055/11055 [01:
12<00:00, 151.99it/s]
```

```
11055
```

In [212]:

```
# average Word2Vec
# compute average word2vec for each review.
X_test_tfidf_w2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
```

```
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_tfidf_w2v_essay.append(vector)

print(len(X_test_tfidf_w2v_essay))
```

```
100%|████████████████████████████████████████████████████████████████| 16500/16500 [01:
42<00:00, 161.42it/s]
```

```
16500
```

In [213]:

```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_tfidf_w2v = hstack((X_train_qty_norm.T, X_train_ppp_norm.T, X_train_price_norm.T, X_train_stat
e_ohe, X_train_grade_ohe,X_train_teacher_ohe,
X_train_ccat_ohe,X_train_csub_ohe,X_train_tfidf_w2v_essay,X_train_tfidf_w2v_title)).tocsr()

X_cv_tfidf_w2v = hstack((X_cv_qty_norm.T, X_cv_ppp_norm.T, X_cv_price_norm.T, X_cv_state_ohe, X_cv_
grade_ohe,X_cv_teacher_ohe,
X_cv_ccat_ohe,X_cv_csub_ohe,X_cv_tfidf_w2v_title,X_cv_tfidf_w2v_essay)).tocsr()

X_te_tfidf_w2v = hstack((X_test_qty_norm.T, X_test_ppp_norm.T, X_test_price_norm.T, X_test_state_oh
e, X_test_grade_ohe,X_test_teacher_ohe,
X_test_ccat_ohe,X_test_csub_ohe,X_test_tfidf_w2v_essay,X_test_tfidf_w2v_title)).tocsr()
print("Final Data matrix")
print(X_tr_tfidf_w2v.shape, y_train.shape)
print(X_cv_tfidf_w2v.shape, y_cv.shape)
print(X_te_tfidf_w2v.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 701) (22445,)
(11055, 701) (11055,)
(16500, 701) (16500,)
====================================================================================================
```

In [214]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 100, 500]}

clf = GridSearchCV(dt, parameters, cv= 10, scoring='roc_auc')

clf.fit(X_tr_tfidf_w2v, y_train)
train_auc_tfidf_w2v= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc_tfidf_w2v = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

In [215]:

```
train_auc_tfidf_w2v
```

Out[215]:

```
array([0.57118602, 0.57118602, 0.57118602, 0.57118602, 0.69224437,
       0.69215126, 0.69101701, 0.69049586, 0.80613876, 0.80351014,
```

```
       0.7768032 , 0.74501147, 0.99944787, 0.996425  , 0.93980641,
       0.82604121, 0.99951328, 0.99674887, 0.93970675, 0.82658929,
       0.99953587, 0.99668922, 0.93961553, 0.8262793 , 0.99953717,
       0.99665559, 0.93879603, 0.82643881])
```

In [216]:

```
cv_auc_tfidf_w2v
```

Out[216]:

```
array([0.56767377, 0.56767377, 0.56767377, 0.56767377, 0.64307799,
       0.64316131, 0.64188981, 0.64206951, 0.60673733, 0.6082717 ,
       0.62024594, 0.63389353, 0.53559784, 0.53755505, 0.57167492,
       0.61674339, 0.54071745, 0.54355044, 0.5728732 , 0.61783062,
       0.53627056, 0.54465096, 0.5738918 , 0.61436699, 0.54192105,
       0.54768161, 0.57398795, 0.6153287 ])
```

In [223]:

```
clf.best_params_
```

Out[223]:

```
{'max_depth': 5, 'min_samples_split': 10}
```

In [217]:

```
depth=[1, 5, 10, 50, 100, 500, 1000]

samples=  [5, 10, 100, 500]
```

In [218]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =train_auc_tfidf_w2v[p]
        p+=1

print(res)
```

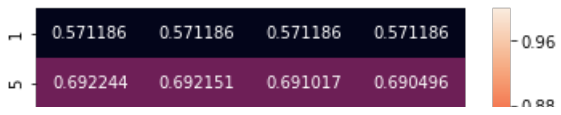
```
[[0.5711860159478361, 0.5711860159478361, 0.5711860159478361, 0.5711860159478361],
[0.6922443720616237, 0.6921512557077896, 0.6910170111785687, 0.6904958577660912],
[0.8061387625739969, 0.8035101397542551, 0.7768032030491143, 0.7450114654278256],
[0.9994478659236327, 0.996424998291843, 0.9398064087737596, 0.8260412142828208],
[0.999513278463878, 0.9967488692243652, 0.9397067460463641, 0.8265892925864478],
[0.9995358697807205, 0.996689223255478, 0.9396155261230337, 0.8262793048446314],
[0.9995371682334643, 0.9966555949084391, 0.9387960278119329, 0.8264388076618575]]
```

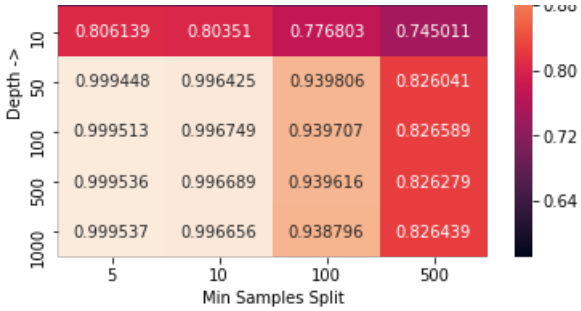
In [219]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[219]:

```
Text(0.5, 15.0, 'Min Samples Split')
```





In [220]:

```
p=0
res = [[0 for x in range(len(samples))] for y in range(len(depth))]

for i, value1 in enumerate(depth):
    for j, value2 in enumerate(samples):
        res[i][j] =cv_auc_tfidf_w2v[p]
        p+=1

print(res)
```

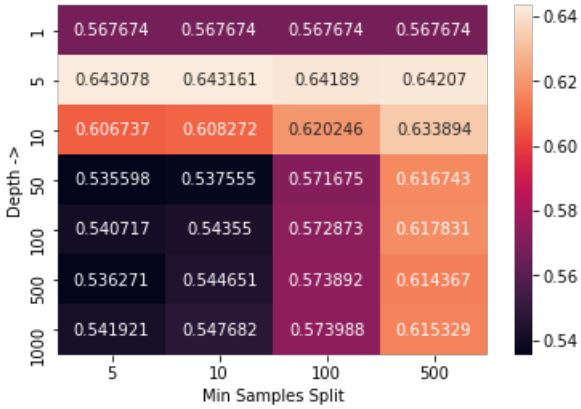
```
[[0.5676737718268385, 0.5676737718268385, 0.5676737718268385, 0.5676737718268385],
[0.6430779879956529, 0.6431613145676998, 0.6418898132671146, 0.6420695120996052],
[0.6067373302300797, 0.608271702437401, 0.6202459357381045, 0.6338935328584684],
[0.5355978442031306, 0.5375550536220466, 0.57167491731111, 0.6167433866269295],
[0.5407174543396159, 0.5435504370499181, 0.5728732006619889, 0.6178306238612611],
[0.5362705630043444, 0.544650962466531, 0.5738917950356841, 0.614366992586569],
[0.5419210466442772, 0.5476816138846999, 0.5739879483794785, 0.6153287009737284]]
```

In [221]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(res, xticklabels=samples,yticklabels=depth,annot=True, ax = ax,fmt='g');
plt.ylabel('Depth ->')
plt.xlabel('Min Samples Split')
```

Out[221]:

```
Text(0.5, 15.0, 'Min Samples Split')
```



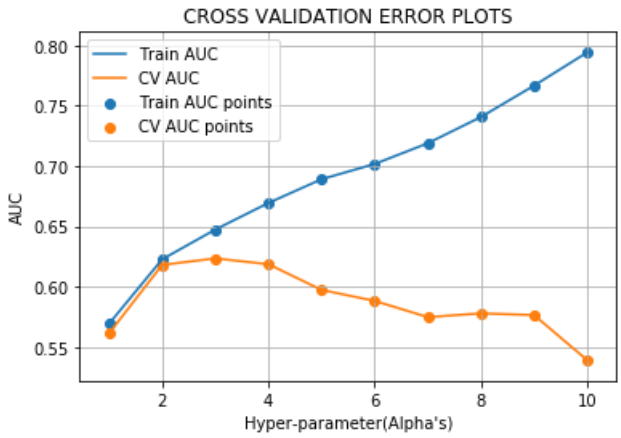
Observations :

1) Decision Tree with maximum depth between 1 ane 10, performs decently on both Train as well as Cross Validation Data.

2) 10 as the value for Minimum samples per split can be considered from above analysis.

In [224]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
import numpy as np
from sklearn import tree
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.
"""
train_auc = []
cv_auc = []
depth = [1, 2, 3, 4, 5, 6, 7,8,9,10]
for i in depth:
    clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=5)
    clf.fit(X_tr_tfidf_w2v, y_train)
    y_train_pred = clf.predict_proba( X_tr_tfidf_w2v)[:,1]
    y_cv_pred = clf.predict_proba( X_cv_tfidf_w2v)[:,1]
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
plt.plot(depth, train_auc, label='Train AUC')
plt.plot(depth, cv_auc, label='CV AUC')
plt.scatter(depth, train_auc, label='Train AUC points')
plt.scatter(depth, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("Hyper-parameter(Alpha's)")
plt.ylabel("AUC")
plt.title("CROSS VALIDATION ERROR PLOTS")
plt.grid()
plt.show()
```



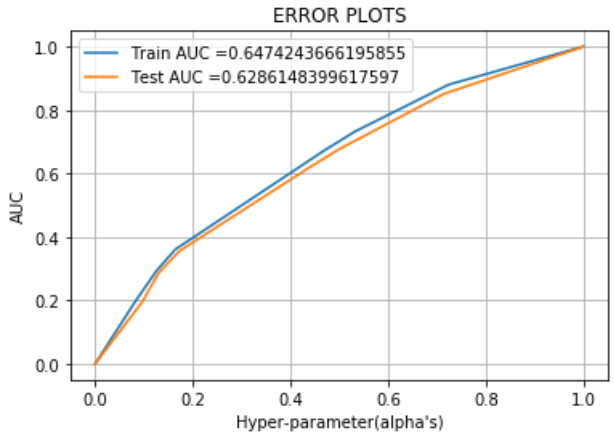
In [225]:

```
i=3
```

In [226]:

```
#https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.r
rve
from sklearn.metrics import roc_curve, auc
from sklearn import tree

clf = tree.DecisionTreeClassifier(class_weight='balanced',max_depth = i, min_samples_split=10)
clf.fit(X_tr_tfidf_w2v, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X_tr_tfidf_w2v)[:, 1]
```

```
y_test_pred = clf.predict_proba(X_te_tfidf_w2v)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Hyper-parameter(alpha's)")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



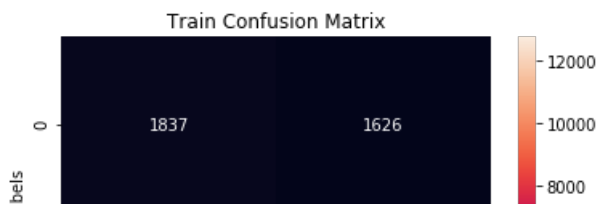
In [227]:

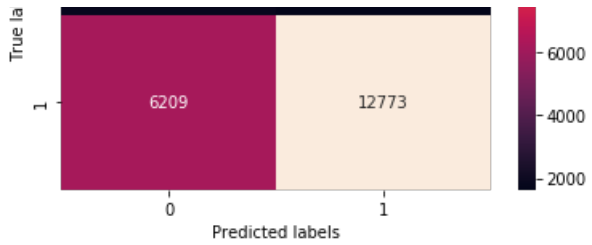
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [228]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
c=confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))

ax= plt.subplot()
sns.heatmap(c, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.35695018211547613 for threshold 0.849
```

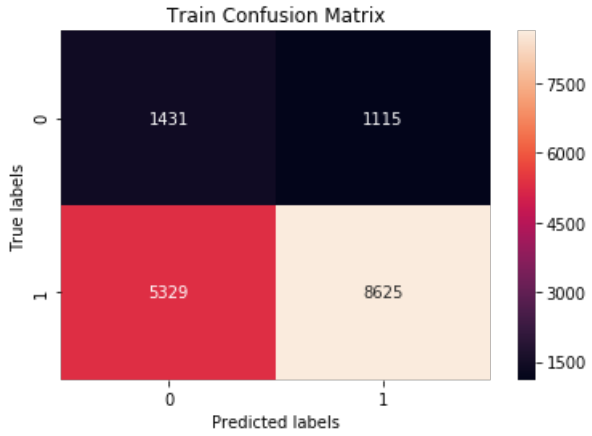




In [229]:

```
print("Test confusion matrix")
d=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))


ax= plt.subplot()
sns.heatmap(d, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.34740944350762654 for threshold 0.849
```



[Task 2] Select best 5k features from Set 2

In [56]:

```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_tfidf = hstack((X_train_qty_norm.T, X_train_ppp_norm.T, X_train_price_norm.T,
X_train_state_ohe, X_train_grade_ohe,X_train_teacher_ohe,
X_train_ccat_ohe,X_train_csub_ohe,X_train_title_tfidf,X_train_essay_tfidf)).tocsr()

X_cr_tfidf = hstack((X_cv_qty_norm.T, X_cv_ppp_norm.T, X_cv_price_norm.T, X_cv_state_ohe, X_cv_grad
e_ohe,X_cv_teacher_ohe,
X_cv_ccat_ohe,X_cv_csub_ohe,X_cv_title_tfidf,X_cv_essay_tfidf)).tocsr()

X_te_tfidf = hstack((X_test_qty_norm.T, X_test_ppp_norm.T, X_test_price_norm.T, X_test_state_ohe, X
_test_grade_ohe,X_test_teacher_ohe,
X_test_ccat_ohe,X_test_csub_ohe,X_test_title_tfidf,X_test_essay_tfidf)).tocsr()
print("Final Data matrix")
print(X_tr_tfidf.shape, y_train.shape)
print(X_cr_tfidf.shape, y_cv.shape)
print(X_te_tfidf.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 6324) (22445,)
(11055, 6324) (11055,)
(16500, 6324) (16500,)
====================================================================================================
```

In [108]:

```
## Fit the Model to obtain the best 5k features

from sklearn.tree import DecisionTreeClassifier


model = DecisionTreeClassifier()
model.fit(X_tr_tfidf, y_train)
```

Out[108]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [109]:

```
## Compute the Feature importances for our Train Features
a=model.tree_.compute_feature_importances(normalize=False)
```

In [110]:

```
df9 = pd.DataFrame(a)
df9 = np.transpose(df9)
```

In [111]:

```
## Store the indexes of the features with atleast some importance. Lets ignore the features with 0
## as the feature importance value and instead consider all the values other than these
best_ind = []
for j in range(6324):
    s = df9[j].sum()
    if s > 0 :
        best_ind.append(j)
    else :
        continue
```

In [112]:

```
## Identify number of Features after feature importance step
len(best_ind)
```

Out[112]:

```
1563
```

While calculating the feature importances of every Column using the DecisionTrees classifier, we hardly get 1563 columns with some importance. The remaining columns of the the total 6234 columns contribute 0 importance.

In [113]:

```
a = X_tr_tfidf.todense()
```

In [114]:

```
set5_train = pd.DataFrame(a)
```

In [115]:

```
final_df_train = set5_train.iloc[:, best_ind]
```

In [116]:

```
final_df_train.shape
```

Out[116]:

```
(22445, 1563)
```

In [117]:

```
b = X_te_tfidf.todense()
```

In [118]:

```
b.shape
```

Out[118]:

```
(16500, 6324)
```

In [119]:

```
set5_test = pd.DataFrame(b)
```

In [120]:

```
final_df_test = set5_test.iloc[:, best_ind]
```

In [121]:

```
c=X_te_tfidf.todense()
```

In [122]:

```
c.shape
```

Out[122]:

```
(16500, 6324)
```

In [123]:

```
set5_cv = pd.DataFrame(c)
```

In [124]:

```
final_df_cv = set5_cv.iloc[:, best_ind]
```

In [125]:

```
final_df_cv.shape
```

Out[125]:

```
(16500, 1563)
```

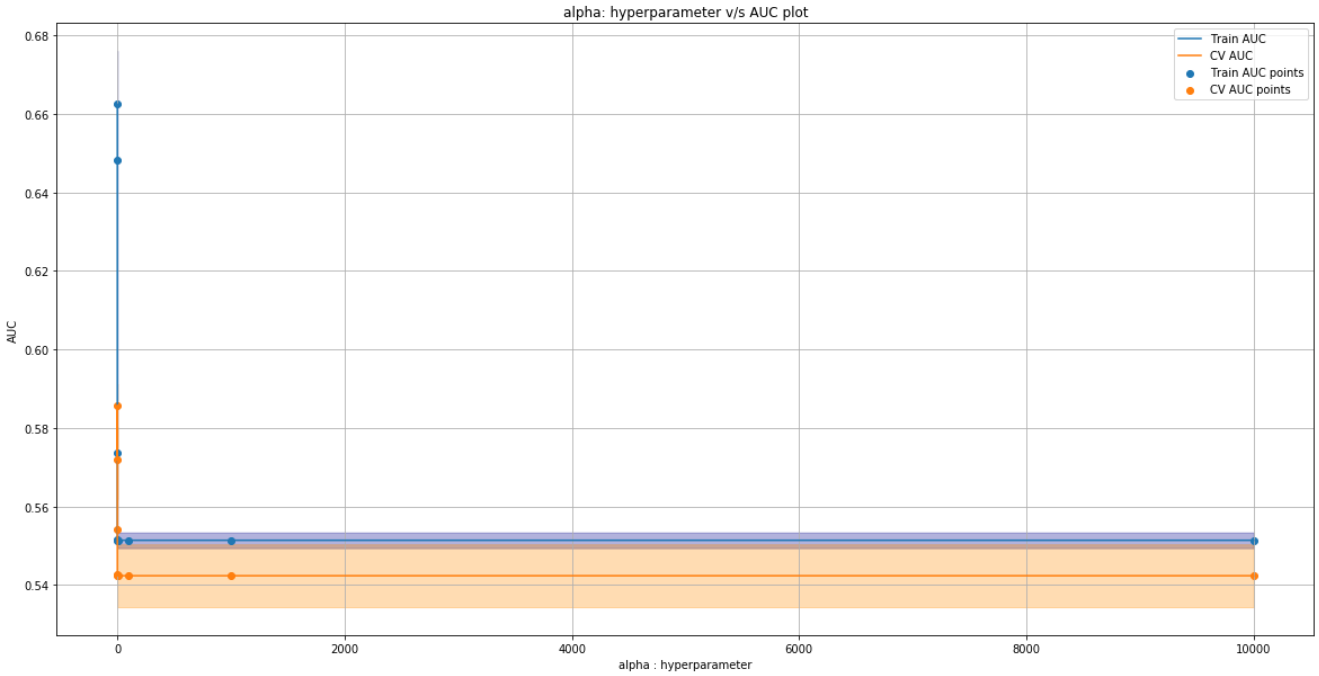
In [140]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
```

```
sv = SGDClassifier(class_weight = 'balanced',loss='hinge', penalty='l2')

parameters = {'alpha':[10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**1, 10**2, 10**3, 10**4]}
clf = GridSearchCV(sv, parameters, cv= 3, scoring='roc_auc')
clf.fit(final_df_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

In [141]:

```
plt.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()
```



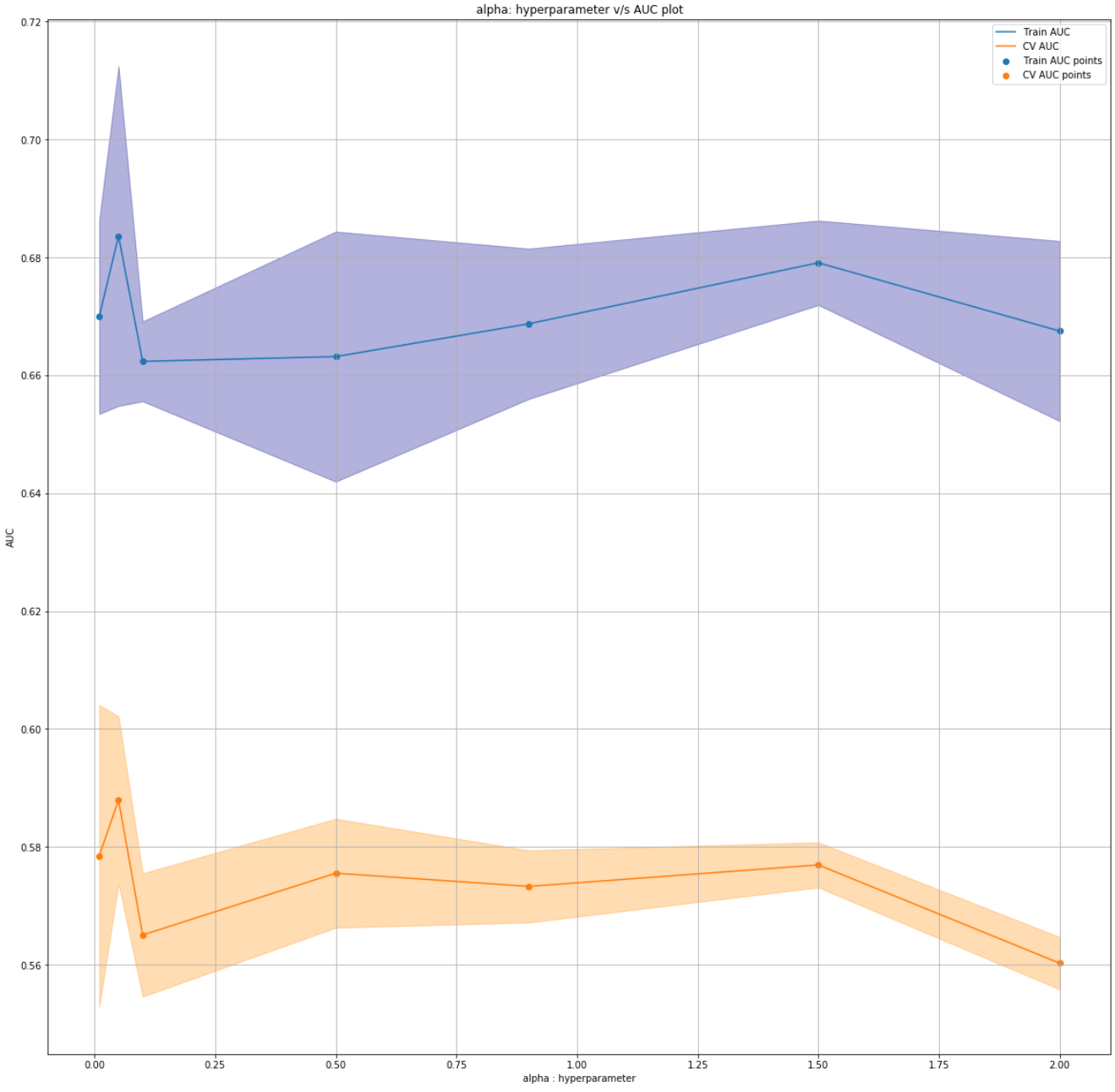
In [142]:

```
from sklearn.linear_model import SGDClassifier
sv = SGDClassifier(class_weight = 'balanced',loss='hinge', penalty='l2')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.5, 0.9, 1.5, 2.0]}
clf = GridSearchCV(sv, parameters, cv= 3, scoring='roc_auc')
clf.fit(final_df, y_train)
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

In [143]:

```
plt.figure(figsize=(20,20))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
```

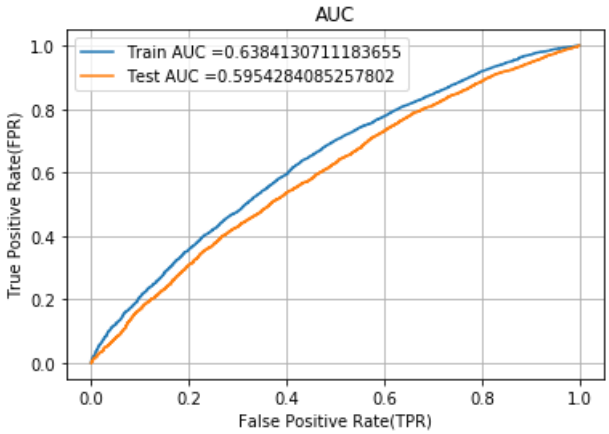
```
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()
```



In [149]:

```
#
https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc
ve
from sklearn.metrics import roc_curve, auc
model = SGDClassifier(class_weight = 'balanced',loss='hinge', penalty='l2', alpha=0.125)
model.fit(final_df_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
```

```
# plot the predicted outputs
y_train_pred = model.decision_function(final_df_train)
y_test_pred = model.decision_function(final_df_test)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix

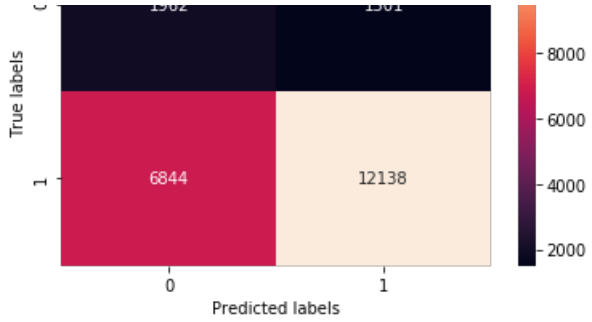
In [151]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
   # print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [152]:

```
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix


print("Train confusion matrix")
a=confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
ax= plt.subplot()
sns.heatmap(a, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Train Confusion Matrix');
```

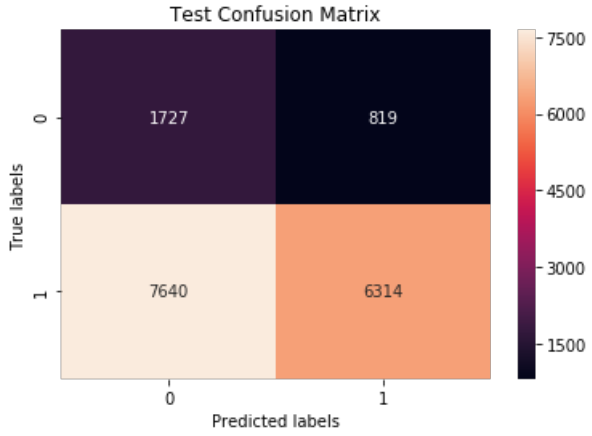
```
Train confusion matrix
```



In [153]:

```
import seaborn as sn
import matplotlib.pyplot as plt

print("Test confusion matrix")
b=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr))
ax1= plt.subplot()
sns.heatmap(b, annot=True, ax = ax1,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
ax1.set_xlabel('Predicted labels');
ax1.set_ylabel('True labels');
ax1.set_title('Test Confusion Matrix');
```

```
Test confusion matrix
```



3. Conclusion

In [161]:

```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperparameters(max depth,min samples split)", "AUC"]

x.add_row(["BOW", "Decision Trees","(5, 500)", 0.611])
x.add_row(["TFIDF", "Decision Trees", "(6, 500)", 0.612])
x.add_row(["AVG W2V", "Decision Trees", "(5, 5)", 0.643])
x.add_row(["TFIDF W2V", "Decision Trees", "(3, 10)", 0.620])
x.add_row(["TFIDF-5k Features", "Linear SVM", "alpha = 0.125", 0.596])


print(x)
```

Vectorizer	Model	Hyperparameters(max depth,min samples split)	AUC
BOW	Decision Trees	(5, 500)	0.611
TFIDF	Decision Trees	(6, 500)	0.612
AVG W2V	Decision Trees	(5, 5)	0.643
TFIDF W2V	Decision Trees	(3, 10)	0.62
TFIDF-5k Features	Linear SVM	alpha = 0.125	0.596

In [ ]: