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`

Feature	Description
	• `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

your neighborhood, and your school are all helpful.

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

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

In [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
from sklearn.model_selection import train_test_split
```

1.1 Reading Data

In [2]:

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

In [3]:

```
project_data
```

Out[3]:

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

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetim
1	140945	p258320	897464ce9ddc600bced1151f324dd63a	Ms.	FL	2016-10-25 09:22:10
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09
5	141660	p154343	a50a390e8327a95b77b9e495b58b9a6e	Mrs.	FL	2017-04-08 22:40:43
6	21147	p099819	9b40170bfa65e399981717ee8731efc3	Mrs.	CT	2017-02-17 19:58:56
7	94142	p092424	5bfd3d12fae3d2fe88684bbac570c9d2	Ms.	GA	2016-09-01 00:02:15
8	112489	p045029	487448f5226005d08d36bdd75f095b31	Mrs.	SC	2016-09-25 17:00:26
9	158561	p001713	140eeac1885c820ad5592a409a3a8994	Ms.	NC	2016-11-17 18:18:56
10	43184	p040307	363788b51d40d978fe276bcb1f8a2b35	Mrs.	CA	2017-01-04 16:40:30
11	127083	p251806	4ba7c721133ef651ca54a03551746708	Ms.	CA	2016-11-14 22:57:28
12	19090	p051126	5e52c92b7e3c472aad247a239d345543	Mrs.	NY	2016-05-23 15:46:02
13	15126	p003874	178f6ae765cd4e0fb143a77c47fd65e2	Mrs.	OK	2016-10-17 09:49:27
14	62232	p233127	424819801de22a60bba7d0f4354d0258	Ms.	MA	2017-02-14 16:29:10
15	67303	p132832	bb6d6d054824fa01576ab38dfa2be160	Ms.	TX	2016-10-05 21:05:38
16	127215	p174627	4ad7e280fddff889e1355cc9f29c3b89	Mrs.	FL	2017-01-18 10:59:05
17	157771	p152491	e39abda057354c979c5b075cffbe5f88	Ms.	NV	2016-11-23 17:14:17

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetim
18	122186	p196421	fcd9b003fc1891383f340a89da02a1a6	Mrs.	GA	2016-08-28 15:04:42
19	146331	p058343	8e07a98deb1bc74c75b97521e05b1691	Ms.	OH	2016-08-06 13:05:20
20	75560	p052326	e0c1aad1f71badeff703fadc15f57680	Mrs.	PA	2016-10-07 18:27:02
21	132078	p187097	2d4a4d2d774e5c2fdd25b2ba0e7341f8	Mrs.	NC	2016-05-17 19:45:13
22	84810	p165540	30f08fbe02eba5453c4ce2e857e88eb4	Ms.	CA	2016-09-01 10:09:15
23	8636	p219330	258ef2e6ab5ce007ac6764ce15d261ba	Mr.	AL	2017-01-10 11:41:06
24	21478	p126524	74f8690562c44fc88f65f845b9fe61d0	Mrs.	FL	2017-03-31 12:34:44
25	20142	p009037	b8bf3507cee960d5fedcb27719df2d59	Mrs.	AL	2017-03-09 15:36:20
26	33903	p040091	7a0a5de5ed94e7036946b1ac3eaa99d0	Ms.	TX	2016-09-18 22:10:40
27	1156	p161033	efdc3cf14d136473c9f62becc00d4cec	Teacher	LA	2016-11-06 16:02:31
28	35430	p085706	22c8184c4660f1c589bea061d14b7f35	Mrs.	GA	2017-01-27 12:34:59
29	22088	p032018	45f16a103f1e00b7439861d4e0728a59	Mrs.	VA	2016-07-15 12:58:40
...	...	...	...	...	...	...
109218	127181	p077978	91f5c69bf72c82edb9bc1f55596d8d95	Mrs.	IL	2017-01-10 14:08:28
109219	65838	p042022	9a6784108c76576565f46446594f99c4	Teacher	FL	2016-07-26 22:43:52
109220	21062	p064087	19c622a38a0cd76c2e9dbcc40541fabd	Mrs.	WI	2016-09-18 13:15:13
109221	81490	p117254	031e299278ac511616b2950fc1312a55	Teacher	NY	2016-07-03 23:09:29

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetim
109222	69138	p152194	6f6e951e435aa9dc966091945414bcc4	Ms.	NC	2016-12-01 20:29:04
109223	5110	p041136	6db62616b4ef6efc2310088f7ea0ae14	Ms.	GA	2017-02-15 14:07:07
109224	109630	p257774	651866d8215616f65934aafcbee21bf5	Ms.	NY	2016-05-23 20:36:51
109225	177841	p079425	c628dff071aa8028b08a5d4972bef2a1	Mrs.	NC	2016-11-14 21:04:43
109226	65359	p085810	1d286ff10ee3982b2b47813f1e415ef2	Ms.	CA	2016-08-12 09:19:22
109227	55643	p146149	e15cd063caa1ce11a45f2179535105f2	Mrs.	NY	2016-10-19 10:10:04
109228	103666	p191845	d0603199630760d8d0eb003108208998	Mrs.	LA	2016-10-14 18:05:17
109229	121219	p055363	523f95270c6aec82bee90e3931ceeeca	Mrs.	CO	2016-09-06 23:19:17
109230	117282	p235512	ee59900af64d9244487e7ed87d0bc423	Ms.	NY	2016-08-09 21:06:33
109231	170085	p248898	9d7a4dae637d1a170778e2db1515e574	Mrs.	AZ	2016-09-17 09:58:59
109232	36083	p204774	c116af7435274872bea9ff123a69cf6a	Mrs.	MD	2017-03-14 19:59:52
109233	155847	p120664	b90258ab009b84e0dc11a7186d597141	Ms.	AZ	2016-12-21 16:36:26
109234	52918	p057638	dd68d9fbae85933c0173c13f66291cbe	Ms.	NY	2017-03-29 20:06:10
109235	69971	p105083	9636fcacbf65eb393133a94c83c4a0d4	Mrs.	TX	2017-01-07 14:50:08
109236	120581	p254202	2950019dd34581dbcddcae683e74207a	Mrs.	OH	2016-08-14 08:27:24

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetim
109237	115336	p056813	07fd2c09f8dfcc74dbb161e1ec3df1fe	Mrs.	IN	2016-05-05 13:03:58
109238	32628	p143363	5b42211690ca8418c7c839436d0b7e49	Mrs.	WI	2016-08-01 21:17:33
109239	156548	p103958	8b9a9dc5bd4aa0301b0ff416e2ed29f6	Mrs.	MN	2016-08-15 17:01:00
109240	93971	p257729	58c112dcb2f1634a4d4236bf0dcdcb31	Mrs.	MD	2016-08-25 13:09:19
109241	36517	p180358	3e5c98480f4f39d465837b2955df6ae0	Mrs.	MD	2016-06-24 11:48:12
109242	34811	p080323	fe10e79b7aeb570dfac87eeea7e9a8f1	Mrs.	SC	2017-03-09 20:00:33
109243	38267	p048540	fadf72d6cd83ce6074f9be78a6fcd374	Mr.	MO	2016-06-17 12:02:31
109244	169142	p166281	1984d915cc8b91aa16b4d1e6e39296c6	Ms.	NJ	2017-01-11 12:49:39
109245	143653	p155633	cdbfd04aa041dc6739e9e576b1fb1478	Mrs.	NJ	2016-08-25 17:11:32
109246	164599	p206114	6d5675dbfafa1371f0e2f6f1b716fe2d	Mrs.	NY	2016-07-29 17:53:15
109247	128381	p191189	ca25d5573f2bd2660f7850a886395927	Ms.	VA	2016-06-29 09:17:01

109248 rows × 17 columns

In [4]:

```
resource_data
```

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
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95
5	p069063	Last to Finish: A Story About the Smartest Boy...	1	16.99

id	description	quantity	price
6	p069063 Mrs. Gorski, I Think I Have the Wiggle Fidgets...	1	9.95
7	p069063 See-N-Read 1503905CQ Reading Tool - Book Size,...	2	10.11
8	p096795 Brewster WPD90218 Wall Pops Flirt Dot, Set of ...	2	9.95
9	p096795 Brewster Wall Pops WPE99065 Peel & Stick Calyp...	2	9.02
10	p096795 TIME For Kids - 3-4 PRINT Bundle - 24 issues /...	40	5.01
11	p149007 Ahora, Spanish, Grades 6 - 12, Level 2 (min. 1...	60	7.99
12	p149007 Scholastic News, Grades 5/6 (min. 10 subscript...	96	5.25
13	p149007 Science Spin Grades 3–6 - 8 Issues / Min. 10 S...	96	0.99
14	p236235 PP440X - Fairy Tales Problem Solving STEM Kits	2	149.00
15	p052460 DD165AT - Calming Colors® Easy-Clean Room...	1	129.00
16	p052460 DD165SB - Calming Colors® Easy-Clean Room...	1	129.00
17	p052460 DD165SE - Calming Colors® Easy-Clean Room...	1	129.00
18	p052460 DD165SG - Calming Colors® Easy-Clean Room...	1	129.00
19	p233680 AA758BU - Connect & Store Book Bin - Blue	4	4.99
20	p233680 AA758GR - Connect & Store Book Bin - Green	4	4.99
21	p233680 AA758RD - Connect & Store Book Bin - Red	4	4.99
22	p233680 AA758RG - Connect & Store Book Bin - Orange	4	4.99
23	p233680 AA758VT - Connect & Store Book Bin	5	4.99
24	p233680 AA758YE - Connect & Store Book Bin - Yellow	5	4.99
25	p233680 JJ302 - Books On Wheels Mobile Library - 6 Bins	1	149.00
26	p233680 LX468BU - Extra Storage Bin - Blue	2	8.99
27	p233680 LX468GR - Extra Storage Bin - Green	2	8.99
28	p233680 LX468RD - Extra Storage Bin - Red	2	8.99
29	p233680 LX468YE - Extra Storage Bin - Yellow	2	8.99
...	... ...	...	...
1541242	p187432 Samsung Chromebook, 11.6" Screen, 2 GB RAM, 16...	3	202.99
1541243	p187432 Sentry Folding Headphones, Black	10	7.99
1541244	p187432 Sentry Folding Headphones, White	4	7.99
1541245	p149426 Piper Computer Kit | Educational Computer that...	1	299.00
1541246	p238803 CARPET MY FAVORITE COLORS-7FT6INX12FT	1	314.97
1541247	p087783 BALL STAY N PLACE SAND FILL	2	34.07
1541248	p087783 BR302BU - Comfy Floor Seat - Blue	1	49.99
1541249	p087783 BR302RD - Comfy Floor Seat - Red	1	49.99
1541250	p087783 CARDINAL (PP) - CLASSROOM SELECT	3	0.00
1541251	p087783 CF521GR - Giant Comfy Pillow - Green	1	69.99
1541252	p087783 OPTION CLASS - CS NEOCLASS/NEOMOVE SHELL COLOR...	3	0.00
1541253	p087783 STOOL - CS NEOROK - STOOL HEIGHT 12 - RUBBER B...	3	59.47
1541254	p086116 Apple iPad 2 2nd generation Tablet, 1 GHz proc...	1	124.99
1541255	p086116 Apple iPad with Retina Display MD513LL/A (16GB...	11	367.95
1541256	p086116 ProCase iPad Case 9.7" 2017 - Vintage Folio St...	3	11.99
1541257	p086116 iPad 2 Case, iPad 3 Case, iPad 4 Case, AiSMei ...	7	10.99
1541258	p086116 iPad 9.7 2017 Case (New 2017 Model), EasyAcc U...	2	9.90
1541259	p086116 iPad Mini Case, Apple iPad Mini 2 Case, iPad M...	1	14.99
1541260	p228679 AA162 - First 100 Sight-Words Talking Boards	1	59.99
1541261	p228679 EE809 - Magnetic Fishing Poles - Set of 2	2	12.99

	id	description	quantity	price
1541262	p228679	FF468 - Magnetic Sight-Word Sentence Board	1	29.99
1541263	p228679	TT507 - Fishing for Sight-Words - Level 1	1	21.99
1541264	p183340	42 PC GRADESTUFF MID SCHOOL - PACK OF 42	1	219.10
1541265	p183340	Rubbermaid Commercial FG9S3100GRAY Brute Tote ...	1	27.49
1541266	p031981	5pcs DC3V/0.1A 1.5V/0.05A 10x2.7mm Coin Mobile...	2	6.46
1541267	p031981	AmazonBasics 9 Volt Everyday Alkaline Batterie...	1	9.99
1541268	p031981	AmazonBasics AAA Performance Alkaline Batterie...	1	6.99
1541269	p031981	Black Electrical Tape (GIANT 3 PACK) Each Roll...	6	8.99
1541270	p031981	Flormoon DC Motor Mini Electric Motor 0.5-3V 1...	2	8.14
1541271	p031981	WAYLLSHINE 6PCS 2 x 1.5V AAA Battery Spring Cl...	2	7.39

1541272 rows × 4 columns

In [5]:

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

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

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[6]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5

In [7]:

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

```
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[7]:

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

1.2 preprocessing of `project_subject_categories`

In [8]:

```
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 [9]:

```
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 & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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]))
```

In [ ]:

```
```

1.3 Text preprocessing

In [10]:

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

In [11]:

```
X = project_data
y = project_data['project_is_approved'].values
X_train, X_test, y_train, y_test=train_test_split(project_data, 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 [12]:

```
X_train.head(2)
```

Out[12]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate
80707	139034	p031899	e8039815b384d9deda0af6ee2818cee5	Mrs.	SC	2016-07-09 23:34:13	Grades 9-12
72721	168951	p178835	06b70d4ae880ceb16c4c59fe0880ce25	Mrs.	VA	2016-09-29 20:54:16	Grades 3-5

In [13]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [14]:

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

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom as well as the STEM j
ournals, which my students really enjoyed.  I would love to implement more of the Lakeshore STEM k
its in my classroom for the next school year as they provide excellent and engaging STEM
lessons.My students come from a variety of backgrounds, including language and socioeconomic statu
s.  Many of them don't have a lot of experience in science and engineering and these kits give me
the materials to provide these exciting opportunities for my students.Each month I try to do
several science or STEM/STEAM projects.  I would use the kits and robot to help guide my science i
nstruction in engaging and meaningful ways.  I can adapt the kits to my current language arts paci
ng guide where we already teach some of the material in the kits like tall tales (Paul Bunyan) or
Johnny Appleseed.  The following units will be taught in the next school year where I will
implement these kits: magnets, motion, sink vs. float, robots.  I often get to these units and don
't know If I am teaching the right way or using the right materials.    The kits will give me
additional ideas, strategies, and lessons to prepare my students in science.It is challenging to d
evelop high quality science activities.  These kits give me the materials I need to provide my
students with science activities that will go along with the curriculum in my classroom.  Although
I have some things (like magnets) in my classroom, I don't know how to use them effectively.  The
kits will provide me with the right amount of materials and show me how to use them in an
appropriate way.
==================================================
I teach high school English to students with learning and behavioral disabilities. My students all
vary in their ability level. However, the ultimate goal is to increase all students literacy level
s. This includes their reading, writing, and communication levels.I teach a really dynamic group o
f students. However, my students face a lot of challenges. My students all live in poverty and in
a dangerous neighborhood. Despite these challenges, I have students who have the the desire to def
eat these challenges. My students all have learning disabilities and currently all are performing
below grade level. My students are visual learners and will benefit from a classroom that fulfills
their preferred learning style.The materials I am requesting will allow my students to be prepared
for the classroom with the necessary supplies.  Too often I am challenged with students who come t
o school unprepared for class due to economic challenges.  I want my students to be able to focus
on learning and not how they will be able to get school supplies.  The supplies will last all year
.  Students will be able to complete written assignments and maintain a classroom journal.  The ch
art paper will be used to make learning more visual in class and to create posters to aid students
in their learning.  The students have access to a classroom printer.  The toner will be used to pr
int student work that is completed on the classroom Chromebooks.I want to try and remove all barri
ers for the students learning and create opportunities for learning. One of the biggest barriers i
s the students not having the resources to get pens, paper, and folders. My students will be able
to increase their literacy skills because of this project.
==================================================
\"Life moves pretty fast. If you don't stop and look around once in awhile, you could miss it.\"
from the movie, Ferris Bueller's Day Off.  Think back...what do you remember about your
grandparents?  How amazing would it be to be able to flip through a book to see a day in their
lives?My second graders are voracious readers! They love to read both fiction and nonfiction books
.  Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson.
They also love to read about insects, space and plants. My students are hungry bookworms! My stude
nts are eager to learn and read about the world around them. My kids love to be at school and are
like little sponges absorbing everything around them. Their parents work long hours and usually do
not see their children. My students are usually cared for by their grandparents or a family
friend. Most of my students do not have someone who speaks English at home. Thus it is difficult f
or my students to acquire language.Now think forward... wouldn't it mean a lot to your kids,
nieces or nephews or grandchildren, to be able to see a day in your life today 30 years from now?
Memories are so precious to us and being able to share these memories with future generations will
be a rewarding experience.  As part of our social studies curriculum, students will be learning ab
out changes over time.  Students will be studying photos to learn about how their community has ch
anged over time.  In particular, we will look at photos to study how the land, buildings,
clothing, and schools have changed over time.  As a culminating activity, my students will capture
a slice of their history and preserve it through scrap booking. Key important events in their
young lives will be documented with the date, location, and names.   Students will be using photos
from home and from school to create their second grade memories.   Their scrap books will preserve
their unique stories for future generations to enjoy.Your donation to this project will provide my
second graders with an opportunity to learn about social studies in a fun and creative manner.  Th
rough their scrapbooks, children will share their story with others and have a historical document
```

```
for the rest of their lives.
==================================================
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the bi
ggest enthusiasm for learning. My students learn in many different ways using all of our senses an
d multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nSt
udents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it's healthy for their bodies. This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroo
m garden in the spring. We will also create our own cookbooks to be printed and shared with famili
es. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
==================================================
My classroom consists of twenty-two amazing sixth graders from different cultures and backgrounds.
They are a social bunch who enjoy working in partners and working with groups. They are hard-worki
ng and eager to head to middle school next year. My job is to get them ready to make this
transition and make it as smooth as possible. In order to do this, my students need to come to
school every day and feel safe and ready to learn. Because they are getting ready to head to
middle school, I give them lots of choice- choice on where to sit and work, the order to complete
assignments, choice of projects, etc. Part of the students feeling safe is the ability for them to
come into a welcoming, encouraging environment. My room is colorful and the atmosphere is casual.
I want them to take ownership of the classroom because we ALL share it together. Because my time w
ith them is limited, I want to ensure they get the most of this time and enjoy it to the best of t
heir abilities.Currently, we have twenty-two desks of differing sizes, yet the desks are similar t
o the ones the students will use in middle school. We also have a kidney table with crates for sea
ting. I allow my students to choose their own spots while they are working independently or in
groups. More often than not, most of them move out of their desks and onto the crates. Believe it
or not, this has proven to be more successful than making them stay at their desks! It is because
of this that I am looking toward the “Flexible Seating” option for my classroom.\r\n The students
look forward to their work time so they can move around the room. I would like to get rid of the c
onstricting desks and move toward more “fun” seating options. I am requesting various seating so m
y students have more options to sit. Currently, I have a stool and a papasan chair I inherited fro
m the previous sixth-grade teacher as well as five milk crate seats I made, but I would like to gi
ve them more options and reduce the competition for the “good seats”. I am also requesting two rug
s as not only more seating options but to make the classroom more welcoming and appealing. In orde
r for my students to be able to write and complete work without desks, I am requesting a class set
of clipboards. Finally, due to curriculum that requires groups to work together, I am requesting t
ables that we can fold up when we are not using them to leave more room for our flexible seating o
ptions.\r\nI know that with more seating options, they will be that much more excited about coming
to school! Thank you for your support in making my classroom one students will remember
forever!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[20000])
print(sent)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the b
iggest enthusiasm for learning. My students learn in many different ways using all of our senses a
nd multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nS
tudents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies. \r\nStudents will gain math and literature skills as well as a life long enjoyment for health
y cooking.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)
```

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smallest students with the big
gest enthusiasm for learning. My students learn in many different ways using all of our senses and
multiple intelligences. I use a wide range of techniques to help all my students succeed.
Students in my class come from a variety of different backgrounds which makes for wonderful
sharing of experiences and cultures, including Native Americans.  Our school is a caring community
of successful learners which can be seen through collaborative student project based learning in a
nd out of the classroom. Kindergarteners in my class love to work with hands-on materials and have
many different opportunities to practice a skill before it is mastered. Having the social skills t
o work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is
the perfect place to learn about agriculture and nutrition. My students love to role play in our p
retend kitchen in the early childhood classroom. I have had several kids ask me,  Can we try cooki
ng with REAL food?  I will take their idea and create  Common Core Cooking Lessons  where we learn
important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies.   Students will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
```

In [18]:

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

```
 A person is a person no matter how small Dr Seuss I teach the smallest students with the biggest
enthusiasm for learning My students learn in many different ways using all of our senses and multi
ple intelligences I use a wide range of techniques to help all my students succeed Students in my
class come from a variety of different backgrounds which makes for wonderful sharing of
experiences and cultures including Native Americans Our school is a caring community of successful
learners which can be seen through collaborative student project based learning in and out of the
classroom Kindergarteners in my class love to work with hands on materials and have many different
opportunities to practice a skill before it is mastered Having the social skills to work
cooperatively with friends is a crucial aspect of the kindergarten curriculum Montana is the
perfect place to learn about agriculture and nutrition My students love to role play in our
pretend kitchen in the early childhood classroom I have had several kids ask me Can we try cooking
with REAL food I will take their idea and create Common Core Cooking Lessons where we learn
```

with REAL food I will take their idea and create common core cooking Lessons where we learn important math and writing concepts while cooking delicious healthy food for snack time My students will have a grounded appreciation for the work that went into making the food and knowledge of where the ingredients came from as well as how it is healthy for their bodies This project would expand our learning of nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce make our own bread and mix up healthy plants from our classroom garden in the spring We will also create our own cookbooks to be printed and shared with families Students will gain math and literature skills as well as a life long enjoyment for healthy cooking 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', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [20]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_tr = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['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.lower() not in stopwords)
    preprocessed_essays_tr.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 73196/73196 [03:20<00:00, 365.14it/s]
```

In [21]:

```
# after preprocesing
#preprocessed_essays[20000]
X_train.columns.values
```

Out[21]:

```
array(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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'], dtype=object)
```

In [22]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_te = []
# tqdm is for printing the status bar
for sentance in X_test['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.lower() not in stopwords)
    preprocessed_essays_te.append(sent.lower().strip())
```

1.4 Preprocessing of `project_title`

In [23]:

```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_project_title_tr = []
# tqdm is for printing the status bar
for sentance in X_train['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_project_title_tr.append(sent.lower().strip())
```

In [24]:

```
from tqdm import tqdm
preprocessed_project_title_te = []
# tqdm is for printing the status bar
for sentance in X_test['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_project_title_te.append(sent.lower().strip())
```

1.5 Preparing data for models

In [25]:

```
project_data.columns
```

Out[25]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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'],
      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 (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

1.5.1 Vectorizing Categorical data

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

In [26]:

```
my_counter = Counter()
for word in X_train['clean_subcategories'].values:
    my_counter.update(word.split())

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

In [27]:

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

In [28]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict_tr.keys()), lowercase=False, binary=T
rue)
categories_one_hot = vectorizer.fit((X_train['clean_categories'].values))
categories_one_hot_tr=categories_one_hot.transform((X_train['clean_categories'].values))
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot_tr.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (73196, 9)
```

In [29]:

```
categories_one_hot_te = categories_one_hot.transform((X_test['clean_categories'].values))
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot_te.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (36052, 9)
```

In [30]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict_tr.keys()), lowercase=False, bina
ry=True)
sub_categories_one_hot = vectorizer.fit((X_train['clean_subcategories'].values))
sub_categories_one_hot_tr=sub_categories_one_hot.transform((X_train['clean_subcategories'].values)
)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_tr.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'EarlyDevelopment', 'Health_LifeScience', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (73196, 30)
```

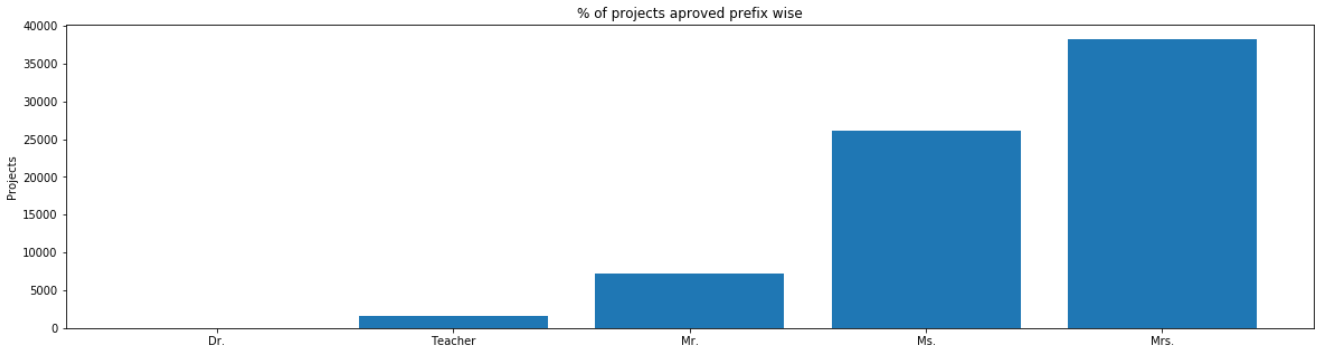
In [31]:

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'EarlyDevelopment', 'Health_LifeScience', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (36052, 30)
```

In [35]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
#teacher_prefix
#how to remove nan from string:https://stackoverflow.com/questions/26837998/pandas-replace-nan-wit
h-blank-empty-string
#cleanedList_teacher_prefix = [x for x in project_data['teacher_prefix'].values if x != np.nan]
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
#from math import isnan
import numpy as np
X_train = X_train.replace(np.nan, 'NA', regex=True)
#X_cv = X_cv.replace(np.nan, 'NA', regex=True)
X_test = X_test.replace(np.nan, 'NA', regex=True)

my_counter = Counter()
for word in X_train['teacher_prefix'].values:
    if word == "NA":
        continue
    my_counter[word] += 1

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict_teacher_prefix = dict(my_counter)
'''cat_dict_teacher_prefix = {k:cat_dict_teacher_prefix [k] for k in cat_dict_teacher_prefix  if n
ot isnan(k)}
cat_dict_teacher_prefix = {k: cat_dict_teacher_prefix[k] for k in cat_dict_teacher_prefix if not i
snan(my_dict[k])}
'''
sorted_cat_dict_teacher_prefix = dict(sorted(cat_dict_teacher_prefix.items(), key=lambda kv: kv[1])
)


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

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



In [36]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict_teacher_prefix.keys()), lowercase=Fal
se, binary=True)
vectorizer.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())


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

```
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encodig  (73196, 5)
```

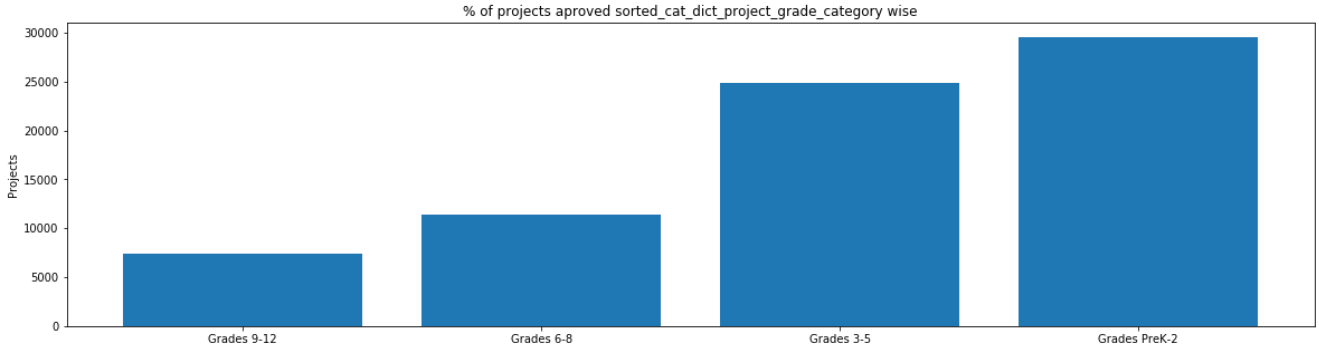
In [37]:

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

```
Shape of matrix after one hot encodig  (36052, 5)
```

In [38]:

```
#project_grade_category
#how to remove nan from string:https://stackoverflow.com/questions/26837998/pandas-replace-nan-wit
h-blank-empty-string
#cleanedList_teacher_prefix = [x for x in project_data['teacher_prefix'].values if x != np.nan]
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
#from math import isnan
import numpy as np
my_counter = Counter()
for word in X_train['project_grade_category'].values:
    my_counter[word] += 1

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict_project_grade_category= dict(my_counter)
'''cat_dict_teacher_prefix = {k:cat_dict_teacher_prefix [k] for k in cat_dict_teacher_prefix  if n
ot isnan(k)}
cat_dict_teacher_prefix = {k: cat_dict_teacher_prefix[k] for k in cat_dict_teacher_prefix if not i
snan(my_dict[k])}
'''
sorted_cat_dict_project_grade_category = dict(sorted(cat_dict_project_grade_category.items(), key=
lambda kv: kv[1]))


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

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

#type(project_data['teacher_prefix'].values[0])
```



In [39]:

```
print(sorted_cat_dict_project_grade_category.keys())
print(sorted_cat_dict_project_grade_category.values())
```

```
dict_keys(['Grades 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2'])
dict_values([7369, 11433, 24851, 29543])
```

In [40]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict_project_grade_category.keys()), lower
case=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values.astype('U'))
print(vectorizer.get_feature_names())


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

```
['Grades 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2']
Shape of matrix after one hot encodig  (73196, 4)
```

In [41]:

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

```
Shape of matrix after one hot encodig  (36052, 4)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [42]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4))
vectorizer.fit(preprocessed_essays_tr)
```

Out[42]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [43]:

```
X_train_essay_bow = vectorizer.transform(preprocessed_essays_tr)
#X_cv_essay_bow = vectorizer.transform(preprocessed_essays_cv)
```

```
X_test_essay_bow = vectorizer.transform(preprocessed_essays_te)
```

In [44]:

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

vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4))
vectorizer.fit(preprocessed_project_title_tr)
```

Out[44]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [45]:

```
X_train_title_bow = vectorizer.transform(preprocessed_project_title_tr)
#X_cv_title_bow =    vectorizer.transform(preprocessed_project_title_cv)
X_test_title_bow =  vectorizer.transform(preprocessed_project_title_te)
```

1.5.2.2 TFIDF vectorizer

In [46]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4))
vectorizer.fit(preprocessed_essays_tr)
```

Out[46]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), norm='l2', preprocessor=None, smooth_idf=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=True,
        vocabulary=None)
```

In [47]:

```
X_train_essay_tfidf = vectorizer.transform(preprocessed_essays_tr)
#X_cv_essay_tfidf = vectorizer.transform(preprocessed_essays_cv)
X_test_essay_tfidf = vectorizer.transform(preprocessed_essays_te)
```

In [48]:

```
vectorizer.fit(preprocessed_project_title_tr)
```

Out[48]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), norm='l2', preprocessor=None, smooth_idf=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=True,
        vocabulary=None)
```

In [49]:

```
X_train_title_tfidf = vectorizer.transform(preprocessed_project_title_tr)
#X_cv_title_tfidf =    vectorizer.transform(preprocessed_project_title_cv)
X_test_title_tfidf =  vectorizer.transform(preprocessed_project_title_te)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [50]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[50]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef
loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\',
encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n
word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\'
\'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the
coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus",
len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words tha
t are present in both glove vectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
```

```
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\r
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [51]:

```
# 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 [52]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_tr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in preprocessed_essays_tr: # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_tr.append(vector)

print(len(avg_w2v_vectors_tr))
print(len(avg_w2v_vectors_tr[0]))
```

```
73196
300
```

In [53]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_te = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in preprocessed_essays_te: # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_te.append(vector)

print(len(avg_w2v_vectors_te))
print(len(avg_w2v_vectors_te[0]))
```

```
36052
300
```

In [54]:

```
avg_w2v_vectors_preprocessed_project_title_tr = []; # the avg-w2v for each sentence/review is stor
ed in this list
for sentence in preprocessed_project_title_tr: # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_preprocessed_project_title_tr.append(vector)
```

```
print(len(avg_w2v_vectors_preprocessed_project_title_tr))
print(len(avg_w2v_vectors_preprocessed_project_title_tr[0]))
```

```
73196
300
```

In [55]:

```
avg_w2v_vectors_preprocessed_project_title_te = []; # the avg-w2v for each sentence/review is stor
ed in this list
for sentence in tqdm(preprocessed_project_title_te): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_preprocessed_project_title_te.append(vector)

print(len(avg_w2v_vectors_preprocessed_project_title_te))
print(len(avg_w2v_vectors_preprocessed_project_title_te[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 36052/36052
[00:03<00:00, 10231.77it/s]

36052
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [56]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays_tr)
# 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 [57]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_tr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in preprocessed_essays_tr: # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_tr.append(vector)

print(len(tfidf_w2v_vectors_tr))
print(len(tfidf_w2v_vectors_tr[0]))
```

```
73196
300
```

In [58]:

```
tfidf_w2v_vectors_te = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in preprocessed_essays_te: # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_te.append(vector)

print(len(tfidf_w2v_vectors_te))
print(len(tfidf_w2v_vectors_te[0]))
```

```
36052
300
```

In [59]:

```
# Similarly you can vectorize for title also
# Similarly you can vectorize for title also
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_project_title_tr)
# 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 [60]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_preprocessed_project_title_tr = []; # the avg-w2v for each sentence/review is st
ored in this list
for sentence in tqdm(preprocessed_project_title_tr): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_preprocessed_project_title_tr.append(vector)

print(len(tfidf_w2v_vectors_preprocessed_project_title_tr))
print(len(tfidf_w2v_vectors_preprocessed_project_title_tr[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 73196/73196
[00:16<00:00, 4528.65it/s]
```

```
73196
300
```

In [61]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_preprocessed_project_title_te = []; # the avg-w2v for each sentence/review is st
ored in this list
```

```
for sentence in preprocessed_project_title_te: # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_preprocessed_project_title_te.append(vector)

print(len(tfidf_w2v_vectors_preprocessed_project_title_te))
print(len(tfidf_w2v_vectors_preprocessed_project_title_te[0]))
```

```
36052
300
```

1.5.3 Vectorizing Numerical features

In [62]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
X_train = pd.merge(X_train, price_data, on='id', how='left')
```

In [63]:

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

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

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

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

```
Mean : 299.0205873271763, Standard deviation : 374.01030845757055
```

In [64]:

```
price_standardized_tr.shape
```

Out[64]:

```
(73196, 1)
```

In [65]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
X_test = pd.merge(X_test, price_data, on='id', how='left')
price_standardized_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
```

In [66]:

```
price_standardized_test.shape
```

Out[66]:

```
(36052, 1)
```

1.5.4 Merging all the above features

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

In [67]:

```
print(categories_one_hot_tr.shape)
print(sub_categories_one_hot_tr.shape)

print(price_standardized_tr.shape)
```

```
(73196, 9)
(73196, 30)
(73196, 1)
```

In [68]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X1_tr = hstack((categories_one_hot_tr, sub_categories_one_hot_tr,
price_standardized_tr,project_grade_category_one_hot_tr,teacher_prefix_one_hot_tr))
X1_tr.shape
```

Out[68]:

```
(73196, 49)
```

In [69]:

```
X1_te = hstack((categories_one_hot_te, sub_categories_one_hot_te,
price_standardized_test,project_grade_category_one_hot_te,teacher_prefix_one_hot_te))
X1_te.shape
```

Out[69]:

```
(36052, 49)
```

Assignment 4: Naive Bayes Apply Multinomial NaiveBayes on these feature sets Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW) Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

The hyper paramter tuning(find best Alpha) Find the best hyper parameter which will give the maximum AUC value Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001 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

Feature importance Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of feature_log*prob* parameter of MultinomialNB and print their corresponding feature names

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. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values. 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. Please visualize your confusion matrices using seaborn heatmaps.

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

2.4.1 Applying Naive Bayes on BOW, SET 1

In [70]:

```
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_bow,X_train_title_bow ,X1_tr)).tocsr()
#X_cr = hstack((X_cv_essay_bow,X_cv_title_bow ,X1_cv)).tocsr()
X_te = hstack((X_test_essay_bow,X_test_title_bow ,X1_te)).tocsr()
```

In [71]:

```
'''X_tr = X_tr[0:30000,:]
y_train = y_train[0:30000]
X_te = X_te[0:20000,:]
y_test = y_test[0:20000]'''
```

Out[71]:

```
'X_tr = X_tr[0:30000,:]\ny_train = y_train[0:30000]\nX_te = X_te[0:20000,:]\ny_test =
y_test[0:20000]'
```

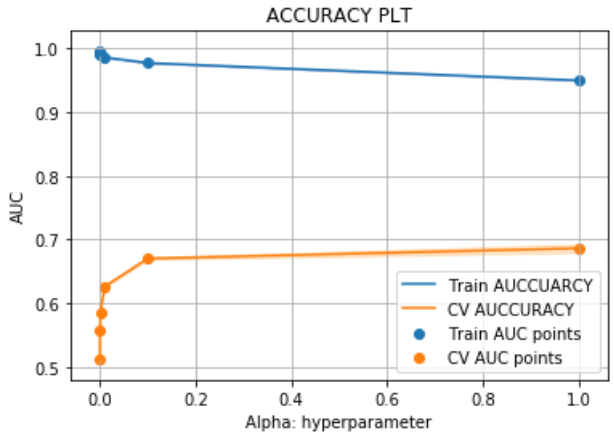
In [72]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
#print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(73196, 240322) (73196,)
(36052, 240322) (36052,)
====================================================================================================
```

In [73]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
parameters = {'alpha':[1,0.1,0.01,0.001,0.0001,0]}
clf = GridSearchCV(model, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(abs(X_tr), 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']

plt.plot(parameters['alpha'], train_auc, label='Train AUCCUARCY')
# 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=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUCCURACY')
# 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=0.2,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("ACCURACY PLT")
plt.grid()
plt.show()
print(clf.best_estimator_)
print(clf.score(X_tr, y_train))
print(clf.best_params_)
```

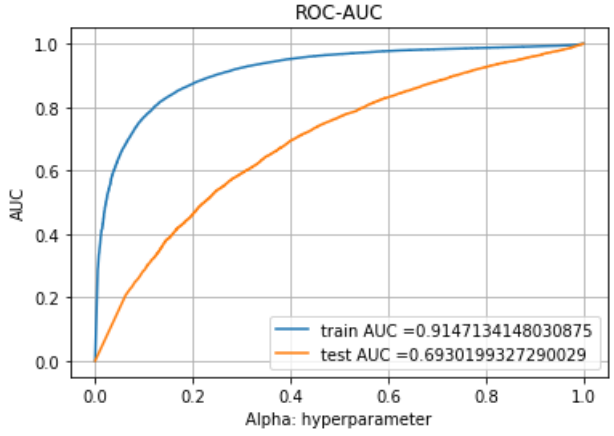



```
MultinomialNB(alpha=1, class_prior=None, fit_prior=True)
0.9147134148030875
{'alpha': 1}
```

From above AUC plot best value of hyper parameter = 1.

In [74]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = MultinomialNB(alpha=1, class_prior=None, fit_prior=True)
neigh.fit(abs(X_tr), 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 = neigh.predict_log_proba(X_tr)[:,1]
y_test_pred =  neigh.predict_log_proba(X_te)[:,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("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ROC-AUC")
plt.grid()
plt.show()
```



1. From above plot its clear that train AUC = .914 which is very high.
2. And test AUC = .693 which is much better than .5 i.e. a random model.

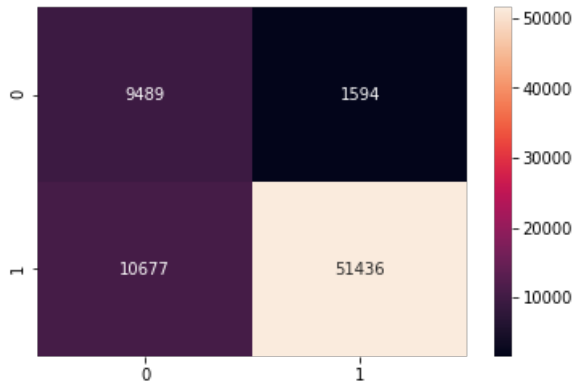
In [75]:

```
import seaborn as sns
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)),np.argmax(tpr*(1-fpr)), "for thresho
ld",threshould[np.argmax(tpr*(1-fpr))], np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [76]:

```
ax = sns.heatmap(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tp
r)), annot=True, fmt="d")
```

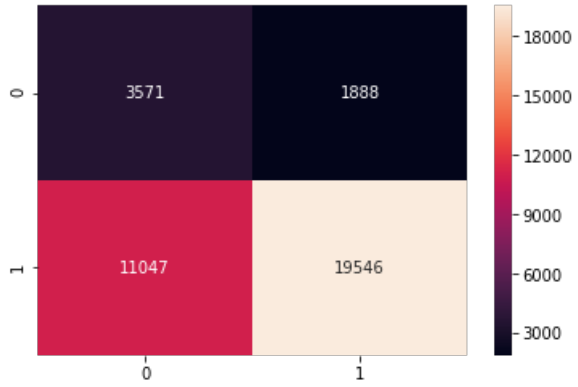
```
the maximum value of tpr*(1-fpr) 0.7090025469104134 4570 for threshold -0.19065872191185917 -0.191
```



In [77]:

```
ax = sns.heatmap(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)),
annot=True, fmt="d")
```

```
the maximum value of tpr*(1-fpr) 0.41894383263877144 3017 for threshold -0.0001916680180329422 -0.
0
```



TN = 3571 < FN = 11047, majority of the negative points are classified as +ve points.

FP = 1888 < Tp = 19546 +ve points are classified more accurately.

2.4.1.1 Top 10 important features of positive class from SET 1

2.4.1.2 Top 10 important features of negative class from SET 1

In [78]:

```
C1 = CountVectorizer(min_df=10,ngram_range=(1,4))

C2=CountVectorizer(X_train_title_bow)
```

In [79]:

```
C1.fit(preprocessed_essays_tr)
```

Out[79]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [80]:

```
f1=C1.get_feature_names()
```

In [81]:

```
len(f1)
```

Out[81]:

```
234380
```

In [82]:

```
C2 = CountVectorizer(min_df=10,ngram_range=(1,4))
C2.fit(preprocessed_project_title_tr)
```

Out[82]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [83]:

```
f2=C2.get_feature_names()
len(f2)
```

Out[83]:

```
5893
```

In [86]:

```
F=f1+f2+['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds'
, 'Health_Sports', 'Math_Science', 'Literacy_Language']+['Economics', 'CommunityService', 'Financia
lLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutri
tionEducation', 'Warmth', 'Care_Hunger', 'PerformingArts', 'SocialSciences', 'CharacterEducation',
'TeamSports', 'Other', 'College_CareerPrep', 'History_Geography', 'Music', 'Health_LifeScience',
'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness',
'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']+['Price']+['Gra
des 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2']+[ 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
```

In [87]:

```
len(F)
```

Out[87]:

240322

In [89]:

```
neg_class_prob_sorted = neigh.feature_log_prob_[0, :].argsort()
pos_class_prob_sorted = neigh.feature_log_prob_[1, :].argsort()
print(np.take(F, neg_class_prob_sorted[240322-50 :240322]))
print(np.take(F, pos_class_prob_sorted[240322-50 :240322]))
```

```
['create' 'science' 'allow' 'high' 'teach' 'supplies' 'best' 'education'
 'different' 'world' 'every' 'Math_Science' 'project' 'provide'
 'Literacy_Language' 'grade' 'get' 'children' 'math' 'one' 'time'
 'technology' 'student' 'also' 'new' 'would' 'make' 'want' 'year' 'class'
 'Price' 'use' 'day' 'skills' 'materials' 'able' 'reading' 'love' 'come'
 'work' 'need' 'many' 'nannan' 'help' 'learn' 'not' 'classroom' 'learning'
 'school' 'students']
['group' 'learners' 'like' 'project' 'different' 'Math_Science' 'high'
 'teach' 'world' 'materials' 'children' 'provide' 'every' 'math' 'get'
 'read' 'allow' 'Literacy_Language' 'grade' 'one' 'student' 'want' 'time'
 'make' 'Price' 'year' 'new' 'skills' 'also' 'books' 'technology' 'would'
 'class' 'come' 'able' 'day' 'love' 'use' 'work' 'need' 'reading' 'nannan'
 'many' 'help' 'learn' 'not' 'classroom' 'learning' 'school' 'students']
```

1. Top 10 features from both +ve and -ve class are similar
2. Top 50 features from both +ve and -ve class are considered.

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [90]:

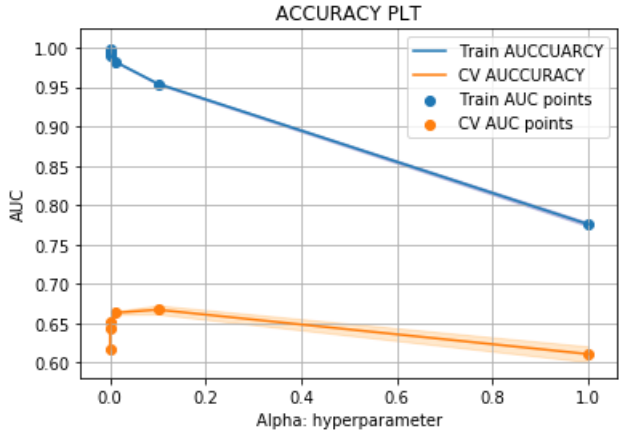
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_tfidf,X_train_title_tfidf ,X1_tr)).tocsr()

X_te = hstack((X_test_essay_tfidf ,X_test_title_tfidf  ,X1_te)).tocsr()

print("Final Data matrix")
print(X_tr.shape, y_train.shape)

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

```
Final Data matrix
(73196, 240322) (73196,)
(36052, 240322) (36052,)
====================================================================================================
```

In [91]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
parameters = {'alpha':[1,0.1,0.01,0.001,0.0001,0]}
clf = GridSearchCV(model, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(abs(X_tr), 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']
```

```
plt.plot(parameters['alpha'], train_auc, label='Train AUCCUARCY')
# 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=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUCCURACY')
# 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=0.2,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("ACCURACY PLT")
plt.grid()
plt.show()
print(clf.best_estimator_)
print(clf.score(X_tr, y_train))
print(clf.best_params_)
```



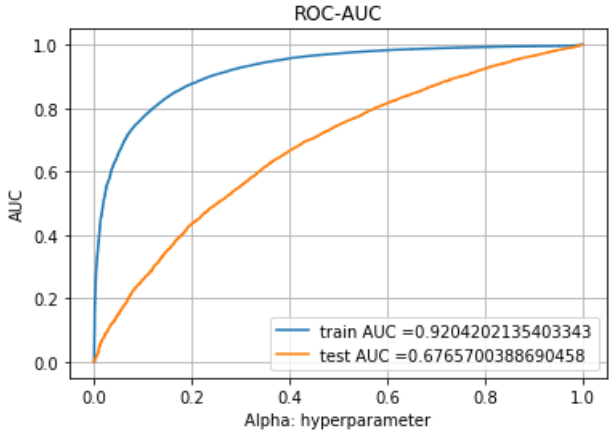

```
MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
0.9204202135403343
{'alpha': 0.1}
```

From the above plot the best value of alpha = 0.1.

In [92]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
neigh.fit(abs(X_tr), 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 = neigh.predict_log_proba(X_tr)[:,1]
y_test_pred =  neigh.predict_log_proba(X_te)[:,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("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ROC-AUC")
plt.grid()
```

```
plt.show()
```

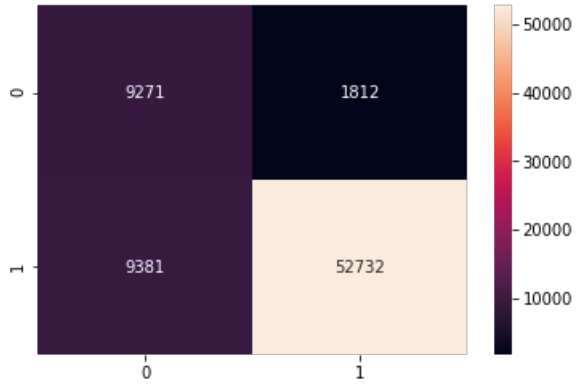


1. From above plot its clear that train AUC = .92 which is very high.
2. And test AUC = .676 which is much better than .5 i.e. a random model.

In [93]:

```
ax = sns.heatmap(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)), annot=True, fmt="d")
```

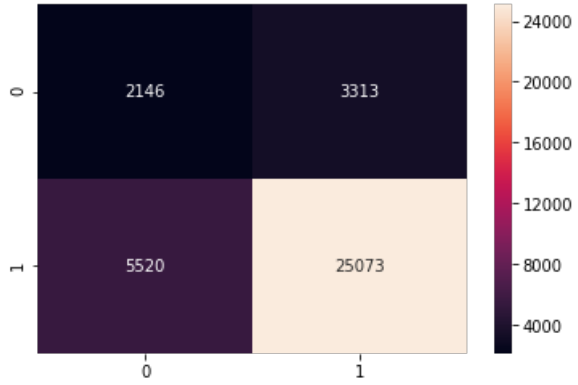
```
the maximum value of tpr*(1-fpr) 0.710167814035483 3394 for threshold -0.1930614896228633 -0.193
```



In [94]:

```
ax = sns.heatmap(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)), annot=True, fmt="d")
```

```
the maximum value of tpr*(1-fpr) 0.4019518752806728 3629 for threshold -0.20494176067154513 -0.205
```



TN = 2146 < FN = 5520, majority of the negative points are classified as +ve points.

FP = 3313 < Tp = 25073, +ve points are classified more accurately.

In [95]:

```
C1 = TfidfVectorizer(min_df=10,ngram_range=(1,4))
C1.fit(preprocessed_essays_tr)
```

Out[95]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), norm='l2', preprocessor=None, smooth_idf=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=True,
        vocabulary=None)
```

In [96]:

```
C2 = TfidfVectorizer(min_df=10,ngram_range=(1,4))
C2.fit(preprocessed_project_title_tr)
```

Out[96]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=10,
        ngram_range=(1, 4), norm='l2', preprocessor=None, smooth_idf=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=True,
        vocabulary=None)
```

In [97]:

```
f1 = C1.get_feature_names()
f2 = C2.get_feature_names()
F=f1+f2+['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds'
, 'Health_Sports', 'Math_Science', 'Literacy_Language']+['Economics', 'CommunityService', 'Financia
lLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutri
tionEducation', 'Warmth', 'Care_Hunger', 'PerformingArts', 'SocialSciences', 'CharacterEducation',
'TeamSports', 'Other', 'College_CareerPrep', 'History_Geography', 'Music', 'Health_LifeScience',
'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness',
'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']+['Price']+['Gra
des 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2']+[ 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
```

In [100]:

```
neg_class_prob_sorted = neigh.feature_log_prob_[0, :].argsort()
pos_class_prob_sorted = neigh.feature_log_prob_[1, :].argsort()
print(np.take(F, neg_class_prob_sorted[240322-50:240322]))
print(np.take(F, pos_class_prob_sorted[240322-50:240322]))
```

```
['our' 'skills' 'seating' 'technology' 'reading' 'supplies' 'math' 'work'
 'need' 'many' 'NutritionEducation' 'materials' 'PerformingArts'
 'classroom' 'SocialSciences' 'learn' 'not' 'help' 'classroom' 'learning'
 'learning' 'CharacterEducation' 'History_Geography' 'Other' 'Music'
 'College_CareerPrep' 'school' 'TeamSports' 'Teacher' 'ESL' 'Gym_Fitness'
 'Health_LifeScience' 'EarlyDevelopment' 'History_Civics'
 'EnvironmentalScience' 'VisualArts' 'students' 'Health_Wellness'
 'Music_Arts' 'AppliedSciences' 'AppliedLearning' 'Health_Sports'
 'SpecialNeeds' 'SpecialNeeds' 'Literature_Writing' 'Mathematics'
 'Literacy' 'Math_Science' 'Literacy_Language' 'Price']
['use' 'need' 'technology' 'Warmth' 'Care_Hunger' 'Care_Hunger' 'Warmth'
 'work' 'classroom' 'many' 'books' 'learn' 'help' 'technology' 'not'
 'reading' 'CharacterEducation' 'PerformingArts' 'SocialSciences'
 'TeamSports' 'learning' 'Teacher' 'classroom' 'learning' 'Other'
 'College_CareerPrep' 'school' 'Music' 'History_Geography'
 'Health_LifeScience' 'EarlyDevelopment' 'ESL' 'Gym_Fitness'
 'EnvironmentalScience' 'VisualArts' 'History_Civics' 'students'
 'Music_Arts' 'Health_Wellness' 'AppliedSciences' 'AppliedLearning'
 'SpecialNeeds' 'SpecialNeeds' 'Health_Sports' 'Literature_Writing'
```

```
'Mathematics' 'Literacy' 'Math_Science' 'Literacy_Language' 'Price']
```

1. Top 10 features from both +ve and -ve class are similar
2. Top 50 features from both +ve and -ve class are considered.

3. Conclusions

In [99]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["vectorizer", "Model", "Hyper parameter ", "AUC"]

x.add_row(["BOW", "Multinomial Naive Bayes", 1, 0.693])
x.add_row(["TFIDF", "Multinomial Naive Bayes", 0.1, 0.676])


print(x)
```

```
+------------+--------------------------+------------------+-------+
| vectorizer |          Model           | Hyper parameter  |  AUC  |
+------------+--------------------------+------------------+-------+
|    BOW     | Multinomial Naive Bayes |         1          | 0.693 |
|   TFIDF    | Multinomial Naive Bayes |        0.1         | 0.676 |
+------------+--------------------------+------------------+-------+
```