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`

Feature	Description
`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`

Feature	Description
school_state	State where school is located (Two-letter U.S. postal code). **Example:** WY
project_subject_subcategories	One or more (comma-separated) subject subcategories for the project. **Examples:** • Literacy • Literature & Writing, Social Sciences
project_resource_summary	An explanation of the resources needed for the project. **Example:** • My students need hands on literacy materials to manage sensory needs!
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2016-04-28 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project. **Example:** bdf8baa8fedef6bfeec7ae4ff1c15c56

Feature	Description
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications 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

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

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
```

```
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:
```
project_data = pd.read_csv('C:/Users/pramod reddy chandi/Desktop/pram/applied ai course/DonorsChoose_2018/train_data.csv')
```

```
resource_data = pd.read_csv('C:/Users/pramod reddy chandi/Desktop/pram/
applied ai course/DonorsChoose_2018/resources.csv')
```

In [3]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

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

In [4]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

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

In [5]:
```
# how to replace elements in list python: https://stackoverflow.com/a/2
582163/4084039
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[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

In [6]:

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

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

Out[6]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00

	id	description	quantity	price
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [7]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stacko
verflow.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 & H
unger"
    for j in i.split(','): # it will split it in three parts ["Math & S
cience", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory b
ased on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are g
oing 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 int
o
    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 [8]:
```
sub_catogories = list(project_data['project_subject_subcategories'].val
ues)
# remove special characters from list of strings python: https://stacko
verflow.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 & H
unger"
    for j in i.split(','): # it will split it in three parts ["Math & S
cience", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory b
ased on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are g
oing 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=Tr
ue)

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

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

1.3 Text preprocessing

In [9]:
```
# 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 [10]:
```
project_data.head(2)
```

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

In [11]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [12]:
```
# 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 clas
sroom as well as the STEM journals, which my students really enjoyed.
I would love to implement more of the Lakeshore STEM kits in my classro
om for the next school year as they provide excellent and engaging STEM
```

lessons.My students come from a variety of backgrounds, including langu
age and socioeconomic status. Many of them don't have a lot of experie
nce in science and engineering and these kits give me the materials to
provide these exciting opportunities for my students.Each month I try t
o do several science or STEM/STEAM projects. I would use the kits and
robot to help guide my science instruction in engaging and meaningful w
ays. I can adapt the kits to my current language arts pacing guide whe
re we already teach some of the material in the kits like tall tales (P
aul Bunyan) or Johnny Appleseed. The following units will be taught in
the next school year where I will implement these kits: magnets, motio
n, sink vs. float, robots. I often get to these units and don't know I
f I am teaching the right way or using the right materials.   The kits
will give me additional ideas, strategies, and lessons to prepare my st
udents in science.It is challenging to develop high quality science act
ivities. These kits give me the materials I need to provide my student
s 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 wit
h the right amount of materials and show me how to use them in an appro
priate way.
==========
I teach high school English to students with learning and behavioral di
sabilities. My students all vary in their ability level. However, the u
ltimate goal is to increase all students literacy levels. This includes
their reading, writing, and communication levels.I teach a really dynam
ic group of students. However, my students face a lot of challenges. My
students all live in poverty and in a dangerous neighborhood. Despite t
hese challenges, I have students who have the the desire to defeat thes
e challenges. My students all have learning disabilities and currently
all are performing below grade level. My students are visual learners a
nd will benefit from a classroom that fulfills their preferred learning
style.The materials I am requesting will allow my students to be prepar
ed for the classroom with the necessary supplies. Too often I am chall
enged with students who come to school unprepared for class due to econ
omic challenges. I want my students to be able to focus on learning an
d not how they will be able to get school supplies. The supplies will
last all year. Students will be able to complete written assignments a
nd maintain a classroom journal. The chart 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 print student work that is completed on the class room Chromebooks.I want to try and remove all barriers for the students learning and create opportunities for learning. One of the biggest barr iers is the students not having the resources to get pens, paper, and f olders. My students will be able to increase their literacy skills beca use of this project.
==========
\"Life moves pretty fast. If you don't stop and look around once in awh ile, you could miss it.\" from the movie, Ferris Bueller's Day Off. T hink 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 liv es?My second graders are voracious readers! They love to read both fict ion and nonfiction books. Their favorite characters include Pete the C at, Fly Guy, Piggie and Elephant, and Mercy Watson. They also love to r ead about insects, space and plants. My students are hungry bookworms! My students are eager to learn and read about the world around them. My kids love to be at school and are like little sponges absorbing everyth ing around them. Their parents work long hours and usually do not see t heir 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 for my students to acquire langua ge.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 abo ut changes over time. Students will be studying photos to learn about how their community has changed 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 imp ortant events in their young lives will be documented with the date, lo cation, and names. Students will be using photos from home and from s chool to create their second grade memories. Their scrap books will p reserve their unique stories for future generations to enjoy.Your donat ion to this project will provide my second graders with an opportunity to learn about social studies in a fun and creative manner. Through th eir 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 sm
allest students with the biggest enthusiasm for learning. My students l
earn in many different ways using all of our senses and multiple intell
igences. I use a wide range of techniques to help all my students succe
ed. \r\nStudents in my class come from a variety of different backgroun
ds which makes for wonderful sharing of experiences and cultures, inclu
ding Native Americans.\r\nOur school is a caring community of successfu
l learners which can be seen through collaborative student project base
d learning in and out of the classroom. Kindergarteners in my class lov
e 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 Co
re Cooking Lessons\" where we learn important math and writing concepts
while cooking delicious healthy food for snack time. My students will h
ave 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's he
althy for their bodies. This project would expand our learning of nutri
tion and agricultural cooking recipes by having us peel our own apples
to make homemade applesauce, make our own bread, and mix up healthy pla
nts from our classroom garden in the spring. We will also create our ow
n cookbooks to be printed and shared with families. \r\nStudents will g
ain math and literature skills as well as a life long enjoyment for hea
lthy cooking.nannan
==========
My classroom consists of twenty-two amazing sixth graders from differen
t cultures and backgrounds. They are a social bunch who enjoy working i
n partners and working with groups. They are hard-working and eager to
head to middle school next year. My job is to get them ready to make th
is transition and make it as smooth as possible. In order to do this, m
y 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 comp
lete assignments, choice of projects, etc. Part of the students feeling

```
safe is the ability for them to come into a welcoming, encouraging envi
ronment. My room is colorful and the atmosphere is casual. I want them
to take ownership of the classroom because we ALL share it together. Be
cause my time with them is limited, I want to ensure they get the most
of this time and enjoy it to the best of their abilities.Currently, we
have twenty-two desks of differing sizes, yet the desks are similar to
the ones the students will use in middle school. We also have a kidney
table with crates for seating. I allow my students to choose their own
spots while they are working independently or in groups. More often tha
n not, most of them move out of their desks and onto the crates. Believ
e it or not, this has proven to be more successful than making them sta
y at their desks! It is because of this that I am looking toward the “F
lexible 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 ge
t rid of the constricting desks and move toward more “fun” seating opti
ons. I am requesting various seating so my students have more options t
o sit. Currently, I have a stool and a papasan chair I inherited from t
he previous sixth-grade teacher as well as five milk crate seats I mad
e, but I would like to give them more options and reduce the competitio
n for the “good seats”. I am also requesting two rugs as not only more
seating options but to make the classroom more welcoming and appealing.
In order for my students to be able to write and complete work without
desks, I am requesting a class set of clipboards. Finally, due to curri
culum that requires groups to work together, I am requesting tables tha
t we can fold up when we are not using them to leave more room for our
flexible seating options.\r\nI know that with more seating options, the
y 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 [13]:

```
# 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 [14]:

```
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 s
mallest students with the biggest enthusiasm for learning. My students
learn in many different ways using all of our senses and multiple intel
ligences. I use a wide range of techniques to help all my students succ
eed. \r\nStudents in my class come from a variety of different backgrou
nds which makes for wonderful sharing of experiences and cultures, incl
uding Native Americans.\r\nOur school is a caring community of successf
ul learners which can be seen through collaborative student project bas
ed learning in and out of the classroom. Kindergarteners in my class lo
ve to work with hands-on materials and have many different opportunitie
s to practice a skill before it is mastered. Having the social skills t
o work cooperatively with friends is a crucial aspect of the kindergart
en curriculum.Montana is the perfect place to learn about agriculture a
nd nutrition. My students love to role play in our pretend kitchen in t
he early childhood classroom. I have had several kids ask me, \"Can we
try cooking with REAL food?\" I will take their idea and create \"Commo
n Core Cooking Lessons\" where we learn important math and writing conc
epts while cooking delicious healthy food for snack time. My students w
ill 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 ap
ples to make homemade applesauce, make our own bread, and mix up health
y plants from our classroom garden in the spring. We will also create o
```

```
ur own cookbooks to be printed and shared with families. \r\nStudents w
ill gain math and literature skills as well as a life long enjoyment fo
r healthy cooking.nannan
==================================================
```

In [15]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remov
e-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 sma
llest students with the biggest enthusiasm for learning. My students le
arn in many different ways using all of our senses and multiple intelli
gences. I use a wide range of techniques to help all my students succee
d.   Students in my class come from a variety of different backgrounds
which makes for wonderful sharing of experiences and cultures, includin
g Native Americans.  Our school is a caring community of successful lea
rners which can be seen through collaborative student project based lea
rning in and out of the classroom. Kindergarteners in my class love to
work with hands-on materials and have many different opportunities to p
ractice a skill before it is mastered. Having the social skills to work
cooperatively with friends is a crucial aspect of the kindergarten curr
iculum.Montana is the perfect place to learn about agriculture and nutr
ition. My students love to role play in our pretend kitchen in the earl
y childhood classroom. I have had several kids ask me,  Can we try cook
ing with REAL food?  I will take their idea and create  Common Core Coo
king 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 k
nowledge of where the ingredients came from as well as how it is health
y for their bodies. This project would expand our learning of nutrition
and agricultural cooking recipes by having us peel our own apples to ma
ke homemade applesauce, make our own bread, and mix up healthy plants f
rom our classroom garden in the spring. We will also create our own coo
kbooks to be printed and shared with families.   Students will gain mat
h and literature skills as well as a life long enjoyment for healthy co
oking.nannan
```

In [16]:
```
#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 multiple intelligences
I use a wide range of techniques to help all my students succeed Studen
ts in my class come from a variety of different backgrounds which makes
for wonderful sharing of experiences and cultures including Native Amer
icans 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 bef
ore it is mastered Having the social skills to work cooperatively with
friends is a crucial aspect of the kindergarten curriculum Montana is t
he perfect place to learn about agriculture and nutrition My students l
ove to role play in our pretend kitchen in the early childhood classroo
m I have had several kids ask me Can we try cooking with REAL food I wi
ll take their idea and create Common Core Cooking Lessons where we lear
n important math and writing concepts while cooking delicious healthy f
ood for snack time My students will have a grounded appreciation for th
e work that went into making the food and knowledge of where the ingred
ients came from as well as how it is healthy for their bodies This proj
ect would expand our learning of nutrition and agricultural cooking rec
ipes 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 th
e 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 [17]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
t'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves'
, 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselve
s', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
```

```
s', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'th
is', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'h
ave', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between',
'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'h
ow', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 's
o', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should',
"should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",
'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is
n't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
 "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:

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

```
100%|██████████| 109248/109248 [00:54<00:00, 2022.76it/s]
```

In [19]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[19]:
```
'a person person no matter small dr seuss i teach smallest students big
gest enthusiasm learning my students learn many different ways using se
nses multiple intelligences i use wide range techniques help students s
ucceed students class come variety different backgrounds makes wonderfu
l sharing experiences cultures including native americans our school ca
ring community successful learners seen collaborative student project b
ased learning classroom kindergarteners class love work hands materials
many different opportunities practice skill mastered having social skil
ls work cooperatively friends crucial aspect kindergarten curriculum mo
ntana perfect place learn agriculture nutrition my students love role p
lay pretend kitchen early childhood classroom i several kids ask can tr
y cooking real food i take idea create common core cooking lessons lear
n important math writing concepts cooking delicious healthy food snack
time my students grounded appreciation work went making food knowledge
ingredients came well healthy bodies this project would expand learning
nutrition agricultural cooking recipes us peel apples make homemade app
lesauce make bread mix healthy plants classroom garden spring we also c
reate cookbooks printed shared families students gain math literature s
kills well life long enjoyment healthy cooking nannan'
```

1.4 Preprocessing of `project_title`

In [20]:
```
# similarly you can preprocess the titles also

# similarly you can preprocess the titles also

project_data.columns
#sent1= decontracted(project_data['project_title'].values[20000])
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
```

```
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e not in stopwords)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248 [00:02<00:00, 48218.53it/s]
```

1.5 Preparing data for models

In [21]:
```
project_data.columns
```

Out[21]:
```
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

In [22]:
```
Y=project_data['project_is_approved']
```

In [23]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity'
:'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [24]:
```
project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_title'] = preprocessed_title

column_values=['clean_categories', 'clean_subcategories', 'school_stat
e', 'project_grade_category', 'teacher_prefix','preprocessed_essays','p
reprocessed_title' ,'price']

def select_columns(dataframe, column_names):
    new_frame = dataframe.loc[:, column_names]
    return new_frame

process_columns=select_columns(project_data,column_values)
```

In [25]:
```
process_columns.head()
```

Out[25]:

	clean_categories	clean_subcategories	school_state	project_grade_category	teacher
0	Math_Science	AppliedSciences Health_LifeScience	CA	Grades PreK-2	Mrs.

	clean_categories	clean_subcategories	school_state	project_grade_category	teacher
1	SpecialNeeds	SpecialNeeds	UT	Grades 3-5	Ms.
2	Literacy_Language	Literacy	CA	Grades PreK-2	Mrs.
3	AppliedLearning	EarlyDevelopment	GA	Grades PreK-2	Mrs.
4	Literacy_Language	Literacy	WA	Grades 3-5	Mrs.

In [26]:

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

# X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, shuffle=Flase)# this is for time series split
X_train, X_test, y_train, y_test = train_test_split(process_columns, Y,
 test_size=0.33) # this is random splitting
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33) # this is random splitting


print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(49041, 8) (49041,)
```

```
(24155, 8) (24155,)
(36052, 8) (36052,)
==========================================================================
============================
```

In [27]:
```
print("train columns",X_train.columns)

print("cv columns",X_cv.columns)

print("test columns",X_test.columns)
```

```
train columns Index(['clean_categories', 'clean_subcategories', 'school
_state',
       'project_grade_category', 'teacher_prefix', 'preprocessed_essay
s',
       'preprocessed_title', 'price'],
      dtype='object')
cv columns Index(['clean_categories', 'clean_subcategories', 'school_st
ate',
       'project_grade_category', 'teacher_prefix', 'preprocessed_essay
s',
       'preprocessed_title', 'price'],
      dtype='object')
test columns Index(['clean_categories', 'clean_subcategories', 'school_
state',
       'project_grade_category', 'teacher_prefix', 'preprocessed_essay
s',
       'preprocessed_title', 'price'],
      dtype='object')
```

1.5.1 Vectorizing Categorical data

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

In [28]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
```

```
vectorizer_categories= CountVectorizer(vocabulary=list(sorted_cat_dict.
keys()), lowercase=False, binary=True)

vectorizer_categories.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer_categories.transform(X_train['cle
an_categories'].values)
categories_one_hot_test = vectorizer_categories.transform(X_test['clean
_categories'].values)
categories_one_hot_cv = vectorizer_categories.transform(X_cv['clean_cat
egories'].values)

print(vectorizer_categories.get_feature_names())

print("Shape of train matrix after one hot encodig ",categories_one_hot
_train.shape)
print("Shape of test matrix after one hot encodig ",categories_one_hot_
test.shape)
print("Shape of cv matrix after one hot encodig ",categories_one_hot_cv
.shape)
```

```
['Care_Hunger', 'Literacy_Language', 'Music_Arts', 'Health_Sports', 'Sp
ecialNeeds', 'AppliedLearning', 'Math_Science', 'History_Civics', 'Warm
th']
Shape of train matrix after one hot encodig  (49041, 9)
Shape of test matrix after one hot encodig  (36052, 9)
Shape of cv matrix after one hot encodig  (24155, 9)
```

In [29]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_subcategories = CountVectorizer(vocabulary=list(sorted_sub_c
at_dict.keys()), lowercase=False, binary=True)
vectorizer_subcategories.fit(X_train['clean_subcategories'].values)

print(vectorizer_subcategories.get_feature_names())

sub_categories_one_hot_train = vectorizer_subcategories.transform(X_tra
in['clean_subcategories'].values)
```

```
sub_categories_one_hot_test = vectorizer_subcategories.transform(X_test
['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer_subcategories.transform(X_cv['cl
ean_subcategories'].values)

print("Shape of train matrix after one hot encodig ",sub_categories_one
_hot_train.shape)
print("Shape of test matrix after one hot encodig ",sub_categories_one_
hot_test.shape)
print("Shape of cv matrix after one hot encodig ",sub_categories_one_ho
t_cv.shape)
```

```
['College_CareerPrep', 'History_Geography', 'EarlyDevelopment', 'TeamSp
orts', 'Literature_Writing', 'Extracurricular', 'VisualArts', 'Music',
'Civics_Government', 'Health_Wellness', 'Mathematics', 'Health_LifeScie
nce', 'Gym_Fitness', 'AppliedSciences', 'ESL', 'PerformingArts', 'Othe
r', 'Economics', 'Care_Hunger', 'CommunityService', 'FinancialLiterac
y', 'Warmth', 'Literacy', 'SpecialNeeds', 'ForeignLanguages', 'Environm
entalScience', 'SocialSciences', 'NutritionEducation', 'CharacterEducat
ion', 'ParentInvolvement']
Shape of train matrix after one hot encodig  (49041, 30)
Shape of test matrix after one hot encodig  (36052, 30)
Shape of cv matrix after one hot encodig  (24155, 30)
```

In [30]:

```
# we use count vectorizer to convert the values of categorical data :sc
hool_state
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_schoolstate= CountVectorizer()
vectorizer_schoolstate.fit(X_train['school_state'])

print(vectorizer_schoolstate.get_feature_names())

school_state_one_hot_train = vectorizer_schoolstate.transform(X_train[
'school_state'].values)
school_state_one_hot_test = vectorizer_schoolstate.transform(X_test['sc
hool_state'].values)
school_state_one_hot_cv = vectorizer_schoolstate.transform(X_cv['school
_state'].values)
```

```
print("Shape of train matrix after one hot encodig ",school_state_one_h
ot_train.shape)
print("Shape of test matrix after one hot encodig ",school_state_one_ho
t_test.shape)
print("Shape of cv matrix after one hot encodig ",school_state_one_hot_
cv.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'h
i', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi',
'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'ny',
'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt',
'wa', 'wi', 'wv', 'wy']
Shape of train matrix after one hot encodig  (49041, 51)
Shape of test matrix after one hot encodig  (36052, 51)
Shape of cv matrix after one hot encodig  (24155, 51)
```

In [31]:
```
#we use count vectorizer to convert the values of categorical data :pro
ject_grade_category
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_project_grade_category = CountVectorizer(stop_words=None)

k=X_train['project_grade_category']
l=X_test['project_grade_category']
m=X_test['project_grade_category']

k.replace(['Grades PreK-2', 'Grades 6-8', 'Grades 3-5','Grades 9-12'],
['A1', 'B2' ,'C3', 'D4'],inplace=True)
l.replace(['Grades PreK-2', 'Grades 6-8', 'Grades 3-5','Grades 9-12'],
['A1', 'B2' ,'C3', 'D4'],inplace=True)
m.replace(['Grades PreK-2', 'Grades 6-8', 'Grades 3-5','Grades 9-12'],
['A1', 'B2' ,'C3', 'D4'],inplace=True)

vectorizer_project_grade_category.fit(k)

project_grade_category_one_hot_train=vectorizer_project_grade_category.
transform(X_train['project_grade_category'].values)
project_grade_category_one_hot_test=vectorizer_project_grade_category.t
ransform(X_test['project_grade_category'].values)
```

```
project_grade_category_one_hot_cv=vectorizer_project_grade_category.tra
nsform(X_cv['project_grade_category'].values)

print("Shape of train matrix after one hot encodig ",project_grade_cate
gory_one_hot_train.shape)
print("Shape of test matrix after one hot encodig ",project_grade_categ
ory_one_hot_test.shape)
print("Shape of cv matrix after one hot encodig ",project_grade_categor
y_one_hot_cv.shape)
```

```
Shape of train matrix after one hot encodig  (49041, 4)
Shape of test matrix after one hot encodig  (36052, 4)
Shape of cv matrix after one hot encodig  (24155, 4)
```

In [32]:

```
#we use count vectorizer to convert the values of categorical data : te
acher_prefix
# getting error as we have null balues replacing them with 0
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_teacher_prefix = CountVectorizer()
project_data['teacher_prefix'].unique()

X_train['teacher_prefix'].fillna("", inplace = True)
X_test['teacher_prefix'].fillna("", inplace = True)
X_cv['teacher_prefix'].fillna("", inplace = True)

vectorizer_teacher_prefix.fit(X_train['teacher_prefix'].values)
print(vectorizer_teacher_prefix.get_feature_names())

teacher_prefix_one_hot_train = vectorizer_teacher_prefix.transform(X_tr
ain['teacher_prefix'].values)
teacher_prefix_one_hot_test = vectorizer_teacher_prefix.transform(X_tes
t['teacher_prefix'].values)
teacher_prefix_one_hot_cv = vectorizer_teacher_prefix.transform(X_cv['t
eacher_prefix'].values)

print("Shape of train matrix after one hot encodig ",teacher_prefix_one
_hot_train.shape)
print("Shape of test matrix after one hot encodig ",teacher_prefix_one_
```

```
hot_test.shape)
print("Shape of cv matrix after one hot encodig ",teacher_prefix_one_ho
t_cv.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of train matrix after one hot encodig  (49041, 5)
Shape of test matrix after one hot encodig  (36052, 5)
Shape of cv matrix after one hot encodig  (24155, 5)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [33]:
```
# We are considering only the words which appeared in at least 10 docum
ents(rows or projects).
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(X_train['preprocessed_essays'])

text_bow_train= vectorizer_bow_essay.transform(X_train['preprocessed_es
says'])
text_bow_test= vectorizer_bow_essay.transform(X_test['preprocessed_essa
ys'])
text_bow_cv= vectorizer_bow_essay.transform(X_cv['preprocessed_essays'
])

print("Shape of train matrix after one hot encodig ",text_bow_train.sha
pe)
print("Shape of test matrix after one hot encodig ",text_bow_test.shape
)
print("Shape of cv matrix after one hot encodig ",text_bow_cv.shape)
```

```
Shape of train matrix after one hot encodig  (49041, 12129)
Shape of test matrix after one hot encodig  (36052, 12129)
Shape of cv matrix after one hot encodig  (24155, 12129)
```

In [34]:
```
# before you vectorize the title make sure you preprocess it
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit(X_train['preprocessed_title'])

title_bow_train = vectorizer_bow_title.transform(X_train['preprocessed_
title'])
title_bow_test = vectorizer_bow_title.transform(X_test['preprocessed_ti
tle'])
title_bow_cv= vectorizer_bow_title.transform(X_cv['preprocessed_title'
])

print("Shape of train matrix after one hot encodig title_bow",title_bow
_train.shape)
print("Shape of test matrix after one hot encodig title_bow",title_bow_
test.shape)
print("Shape of cv matrix after one hot encodig title_bow",title_bow_cv
.shape)
```

```
Shape of train matrix after one hot encodig title_bow (49041, 91)
Shape of test matrix after one hot encodig title_bow (36052, 91)
Shape of cv matrix after one hot encodig title_bow (24155, 91)
```

In [35]:
```
print(vectorizer_bow_title.get_feature_names())
```

```
['absolutely', 'additional', 'afford', 'allow', 'also', 'always', 'arou
nd', 'backgrounds', 'based', 'best', 'books', 'children', 'class', 'com
e', 'correspond', 'cultural', 'curiosity', 'day', 'discussions', 'diver
se', 'eager', 'economic', 'engaging', 'enthusiasm', 'environment', 'exp
ose', 'find', 'first', 'full', 'games', 'genuinely', 'graders', 'high',
'homes', 'importance', 'important', 'includes', 'information', 'inspir
e', 'interest', 'interesting', 'issues', 'kids', 'know', 'lead', 'lear
n', 'learners', 'learning', 'lifelong', 'literature', 'love', 'magazine
s', 'many', 'materials', 'my', 'nannan', 'nonfiction', 'not', 'online',
'our', 'parents', 'past', 'printable', 'provide', 'reading', 'real', 'r
esources', 'rich', 'rigorous', 'school', 'simply', 'skill', 'spark', 's
tandards', 'students', 'subscription', 'teach', 'text', 'the', 'these',
'they', 'topics', 'used', 'using', 'variety', 'various', 'videos', 'wan
t', 'way', 'worksheets', 'world']
```

1.5.2.2 TFIDF vectorizer

In [36]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay= TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(X_train['preprocessed_essays'])

text_tfidf_train= vectorizer_tfidf_essay.transform(X_train['preprocesse
d_essays'])
text_tfidf_test= vectorizer_tfidf_essay.transform(X_test['preprocessed_
essays'])
text_tfidf_cv = vectorizer_tfidf_essay.transform(X_cv['preprocessed_ess
ays'])

print("Shape of train matrix after one hot encodig ",text_tfidf_train.s
hape)
print("Shape of test matrix after one hot encodig ",text_tfidf_test.sha
pe)
print("Shape of cv matrix after one hot encodig ",text_tfidf_cv.shape)
```

```
Shape of train matrix after one hot encodig  (49041, 12129)
Shape of test matrix after one hot encodig  (36052, 12129)
Shape of cv matrix after one hot encodig  (24155, 12129)
```

In [37]:
```
# Similarly you can vectorize for title also

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(X_train['preprocessed_title'])

title_tfidf_train = vectorizer_tfidf_title.transform(X_train['preproces
sed_title'])
title_tfidf_test = vectorizer_tfidf_title.transform(X_test['preprocesse
d_title'])
title_tfidf_cv = vectorizer_tfidf_title.transform(X_cv['preprocessed_ti
tle'])
```

```
print("Shape of train matrix after one hot encodig ",title_tfidf_train.
shape)
print("Shape of test matrix after one hot encodig ",title_tfidf_test.sh
ape)
print("Shape of cv matrix after one hot encodig ",title_tfidf_cv.shape)
```

```
Shape of train matrix after one hot encodig  (49041, 91)
Shape of test matrix after one hot encodig  (36052, 91)
Shape of cv matrix after one hot encodig  (24155, 91)
```

1.5.3 Vectorizing Numerical features

In [38]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 21
3.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = Normalizer()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the m
ean and standard deviation of this data


# Now standardize the data with above maen and variance.
price_standardized_train= price_scalar.transform(X_train['price'].value
s.reshape(-1, 1))
price_standardized_test= price_scalar.transform(X_test['price'].values.
reshape(-1, 1))
price_standardized_cv= price_scalar.transform(X_cv['price'].values.resh
ape(-1, 1))

print("After vectorizations")
```

```
print(price_standardized_train.shape, y_train.shape)
print(price_standardized_test.shape, y_test.shape)
print(price_standardized_cv.shape, y_cv.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(36052, 1) (36052,)
(24155, 1) (24155,)
```

1.5.4 Merging all the above features

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

In [39]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)

X_1 = hstack((school_state_one_hot_train,categories_one_hot_train,sub_categories_one_hot_train))
X_cat_train=hstack((X_1,teacher_prefix_one_hot_train,project_grade_category_one_hot_train))


X_2 = hstack((school_state_one_hot_test,categories_one_hot_test,sub_categories_one_hot_test))
X_cat_test=hstack((X_2,teacher_prefix_one_hot_test,project_grade_category_one_hot_test))

X_3 = hstack((school_state_one_hot_cv,categories_one_hot_cv,sub_categories_one_hot_cv))
X_cat_cv=hstack((X_3,teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv))

#dealing with numerical values
#considering the value of price standardized values
```

```
#price_standardized_train = pd.DataFrame({'price_standard_train':price_
standardized_train})
#price_standardized_test = pd.DataFrame({'price_standard_test':price_st
andardized_test})
#price_standardized_cv = pd.DataFrame({'price_standard_cv':price_standa
rdized_cv})


#combining numerical ,project_title(BOW)  and preprocessed_essay (BOW)
num_text_train=hstack((text_bow_train,price_standardized_train,title_bo
w_train))
num_text_test=hstack((text_bow_test,price_standardized_test,title_bow_t
est))
num_text_cv=hstack((text_bow_cv,price_standardized_cv,title_bow_cv))

#froming features for set1

set1_train=hstack((X_cat_train,num_text_train))
set1_test=hstack((X_cat_test,num_text_test))
set1_cv=hstack((X_cat_cv,num_text_cv))

#numerical + project_title(TFIDF)+ preprocessed_essay (TFIDF)
num_tfidf_train=hstack((price_standardized_train,text_tfidf_train,title
_tfidf_train))
num_tfidf_test=hstack((price_standardized_test,text_tfidf_test,title_tf
idf_test))
num_tfidf_cv=hstack((price_standardized_cv,text_tfidf_cv,title_tfidf_cv
))

#froming features for set2

set2_train=hstack((X_cat_train,num_tfidf_train))
set2_test=hstack((X_cat_test,num_tfidf_test))
set2_cv=hstack((X_cat_cv,num_tfidf_cv))
```

In [40]:
```
set1_train.shape
```

Out[40]:
```
(49041, 12320)
```

Assignment 4: Naive Bayes

1. **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)

2. **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

3. **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

4. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. 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.

5. **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

Naive Bayes

2.4 Appling NB() on different kind of featurization as mentioned in the instructions

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

In [41]:
```
X_tr=set1_train.tocsr()
X_cr=set1_cv.tocsr()
X_te=set1_test.tocsr()
```

picking random Alpha values

In [42]:
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math
```

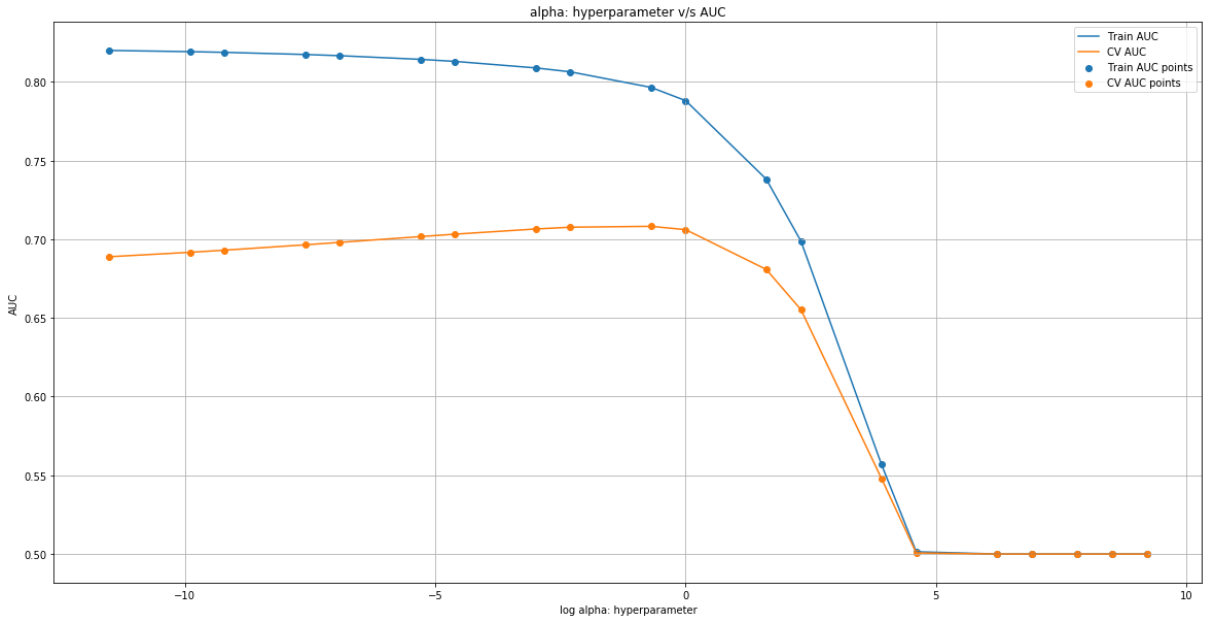
```
train_auc = []
cv_auc = []
log_alphas = []

alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05,
0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior = [0.5, 0.5] )
    nb.fit(X_tr, y_train)

    y_train_pred = nb.predict_log_proba(X_tr)[:,1]
    y_cv_pred = nb.predict_log_proba(X_cr)[:,1]


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probab
ility estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)
```

```
100%|██████████| 20/20 [00:02<00:00,  8.64it/s]
100%|██████████| 20/20 [00:00<?, ?it/s]
```

In [43]:

```
plt.figure(figsize=(20,10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("log alpha: hyperparameter")
```

```
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Summary

1.we have considered alpha values of 10^-4 to 10^4

2.X axis consist of log alpha and ?Y axis consist of AUC

3.very high values and very low values of alpha seems to be ineffective in developing the model

Grid search using CV=10

In [44]:
```
from sklearn.model_selection import GridSearchCV
```

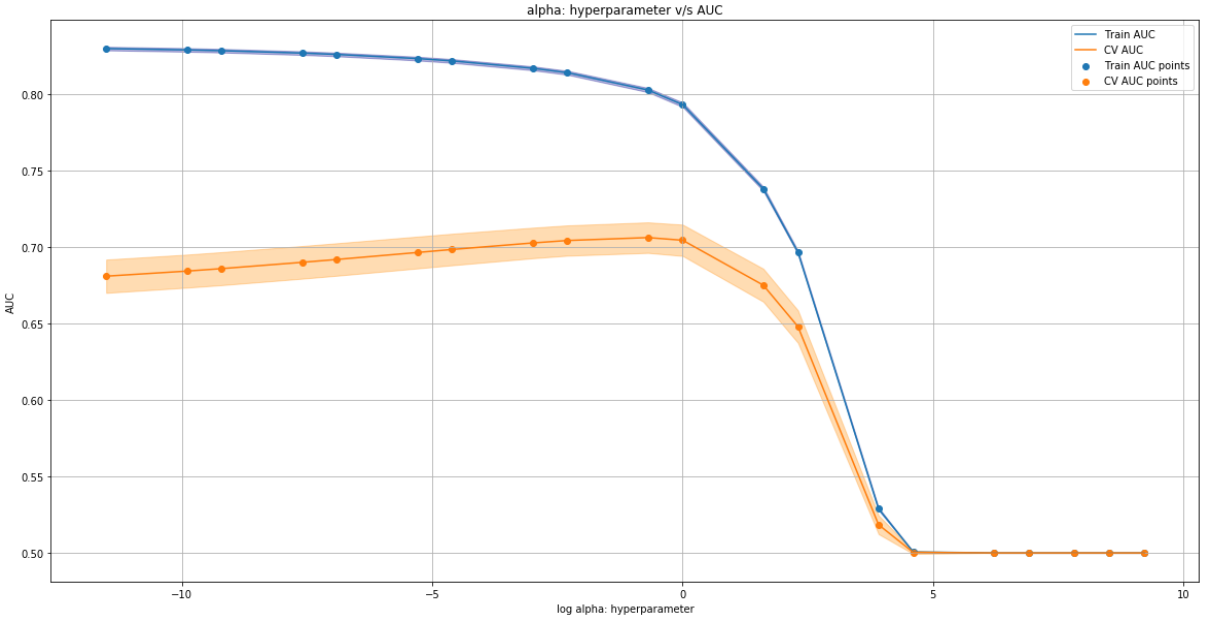
```
nb = MultinomialNB()

parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]}

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

clf.fit(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']
```

In [45]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05,
0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(20,10))

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4
084039
plt.gca().fill_between(log_alphas,train_auc - train_auc_std,train_auc +
 train_auc_std,alpha=0.3,color='darkblue')

plt.plot(log_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4
084039
plt.gca().fill_between(log_alphas,cv_auc - cv_auc_std,cv_auc + cv_auc_s
td,alpha=0.3,color='darkorange')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')
```

```
plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```

```
100%|██████████| 20/20 [00:00<?, ?it/s]
```

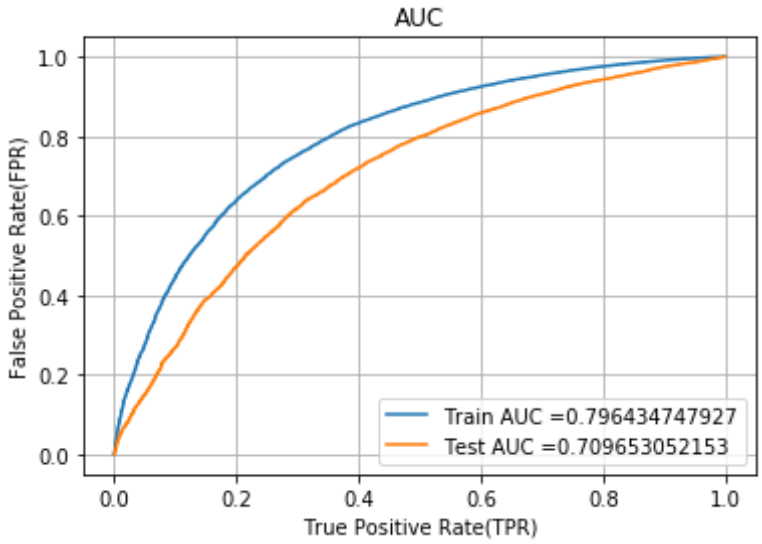


Observations:We see that the negative value of alpha works fine for train data but doesnot seem to be work fine for Cv data. We can choose the best value of (0-1) as maximum AUC on cv data and gap between the train and cv is less

Train model using the best hyperparameter

In [46]:
```
best_k=0.5

# tried with 0 0.5 and 1 we are getting best values for 0.5 so consider
ing it.
```

In [47]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc
_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

nb_bow = MultinomialNB(alpha = best_k,class_prior = [0.5, 0.5])

nb_bow.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probabilit
y estimates of the positive class
# not the predicted outputs

y_train_pred = nb_bow.predict_log_proba(X_tr)[:,1]
y_test_pred = nb_bow.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, t
rain_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_
tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [48]:
```
#Confusion matrix
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(fpr*(1-tpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is
 very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for th
reshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [49]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
```

```
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, tr
ain_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test
_fpr, test_fpr)))
```

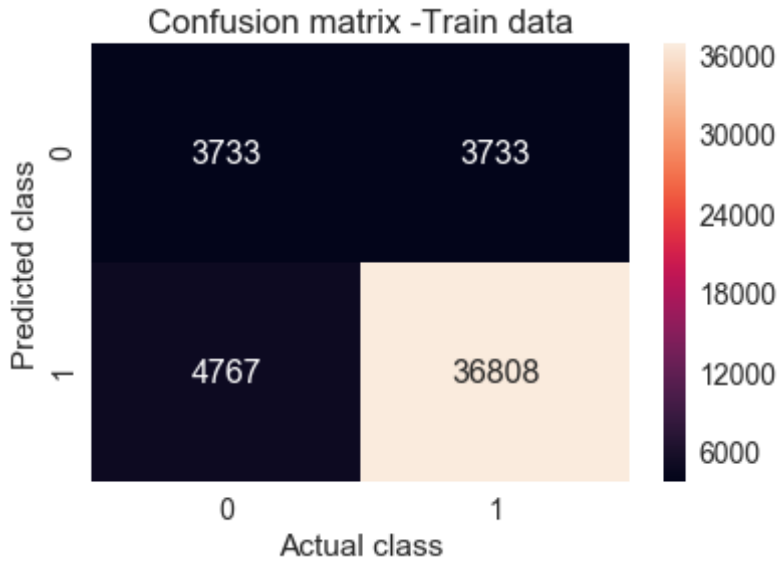
```
==========================================================================
============================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold -3.58
[[ 3733  3733]
 [ 4767 36808]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold -1.718
[[ 2904  2544]
 [ 6832 23772]]
```

In [50]:
```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y
_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold -3.58
```

In [51]:
```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fm
t='g')


plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -Train data")
```

Out[51]:
```
Text(0.5,1,'Confusion matrix -Train data')
```

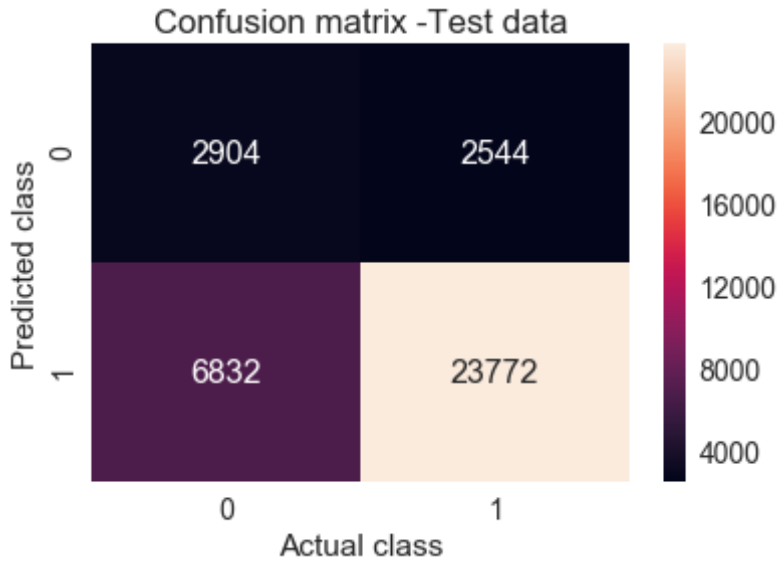


In [52]:
```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold -1.718
```

In [53]:
```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -Test data")
```

Out[53]:
```
Text(0.5,1,'Confusion matrix -Test data')
```



2.4.1.1 Top 10 important features of positive class from SET 1

In [54]:
```
nb_bow = MultinomialNB(alpha = 4,class_prior = [0.5, 0.5])

nb_bow.fit(X_tr, y_train)
```

Out[54]:
```
MultinomialNB(alpha=4, class_prior=[0.5, 0.5], fit_prior=True)
```

In [55]:
```
len(nb_bow.feature_log_prob_[1,:])
```

Out[55]:
```
12320
```

In [56]:
```
bow_features_names = []
```

In [57]:
```
for a in vectorizer_categories.get_feature_names() :
    bow_features_names.append(a)

for b in vectorizer_subcategories.get_feature_names() :
    bow_features_names.append(b)
```

```
for c in vectorizer_schoolstate.get_feature_names() :
    bow_features_names.append(c)

for d in vectorizer_project_grade_category.get_feature_names() :
    bow_features_names.append(d)

for e in vectorizer_teacher_prefix.get_feature_names() :
    bow_features_names.append(e)

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

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


bow_features_names.append("price")
```

In [58]:
```
len(bow_features_names)
```

Out[58]:
```
12320
```

In [59]:
```
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()
pos_class_prob_sorted =pos_class_prob_sorted[::-1]
print(np.take(bow_features_names, pos_class_prob_sorted[:10]))
```

```
['students' 'subscription' 'price' 'magazines' 'learners' 'used' 'thes
e'
 'school' 'real' 'my']
```

2.4.1.2 Top 10 important features of negative class from SET 1

In [60]:
```
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()
neg_class_prob_sorted =neg_class_prob_sorted [::-1]
print(np.take(bow_features_names, neg_class_prob_sorted[:10]))
```

```
['students' 'subscription' 'price' 'learners' 'these' 'used' 'magazine
s'
 'school' 'around' 'includes']
```

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [61]:
```
X_tr=set2_train.tocsr()
X_cr=set2_cv.tocsr()
X_te=set2_test.tocsr()
```

picking random Alpha values

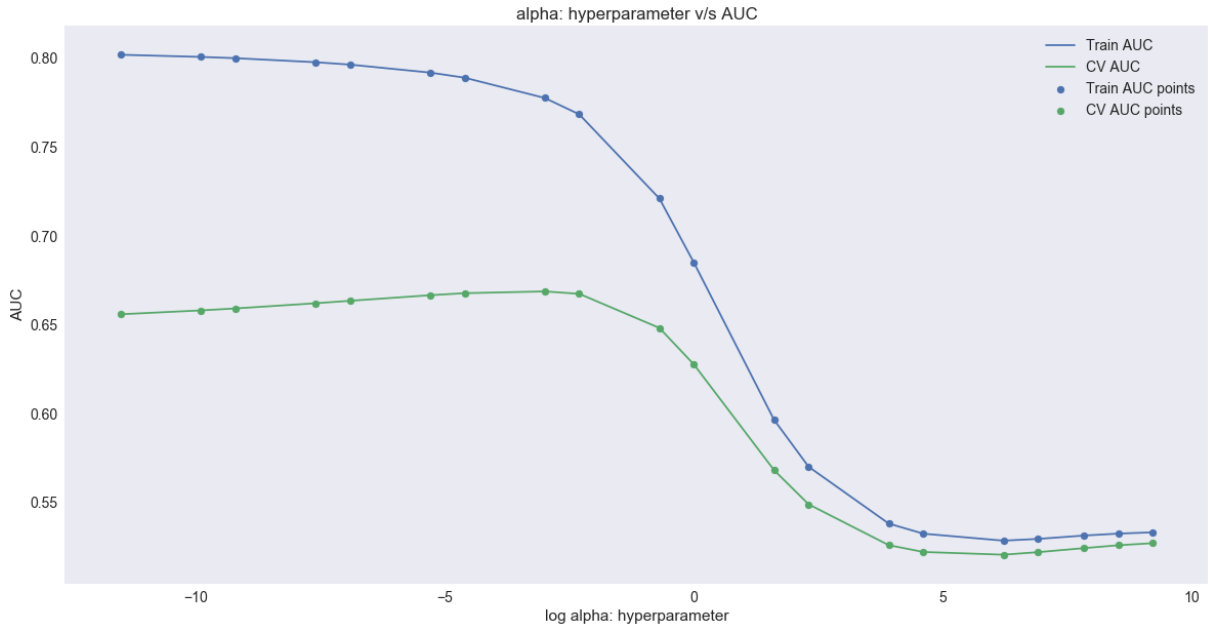
In [62]:
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_alphas = []

alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05,
0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior = [0.5, 0.5])
    nb.fit(X_tr, y_train)

    y_train_pred =nb.predict_log_proba(X_tr)[:,1]
    y_cv_pred = nb.predict_log_proba(X_cr)[:,1]


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

```
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)
```

```
100%|██████████| 20/20 [00:02<00:00,  7.85it/s]
100%|██████████| 20/20 [00:00<?, ?it/s]
```

In [63]:

```
plt.figure(figsize=(20,10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



summary

WE have taken alpha varies from 10^-4 to 10 ^4 and model seems to be ineffective with this values. we can see best values of alpha we check this with cross validation.

Grid search using CV=10

In [64]:
```
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB()

parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]}

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

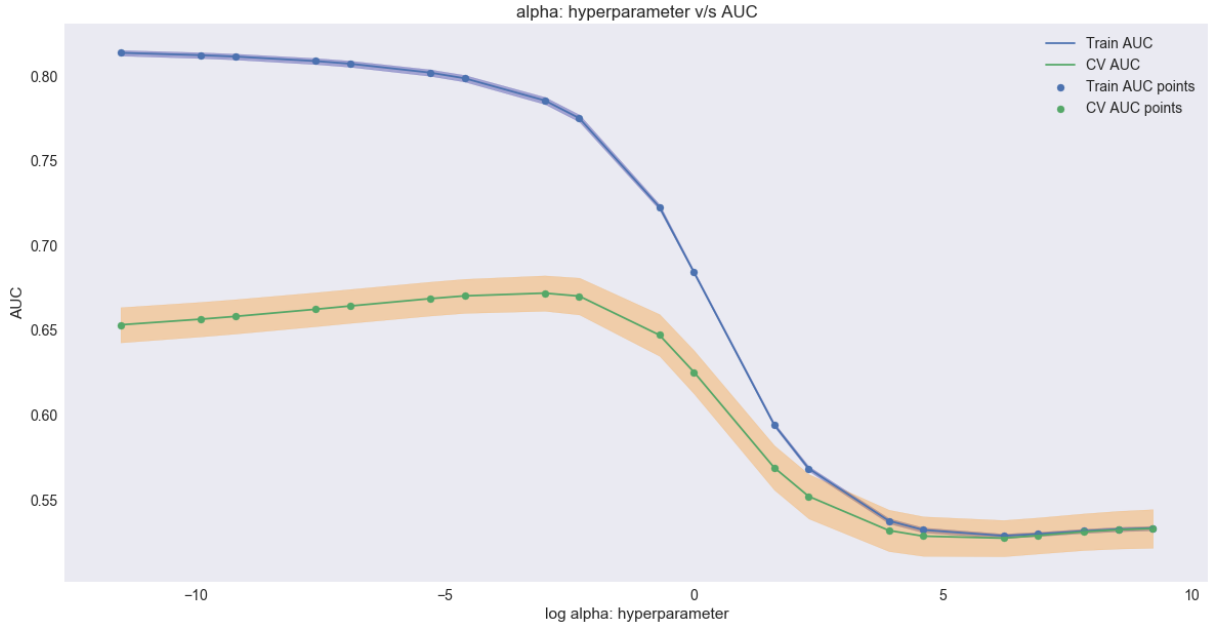
```
clf.fit(X_tr, y_train)

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']
```

In [65]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05,
0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(20,10))

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4
084039
plt.gca().fill_between(log_alphas,train_auc - train_auc_std,train_auc +
 train_auc_std,alpha=0.3,color='darkblue')

plt.plot(log_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4
084039
plt.gca().fill_between(log_alphas,cv_auc - cv_auc_std,cv_auc + cv_auc_s
td,alpha=0.3,color='darkorange')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
```

```
plt.grid()
plt.show()
```

```
100%|██████████| 20/20 [00:00<?, ?it/s]
```



Observation:

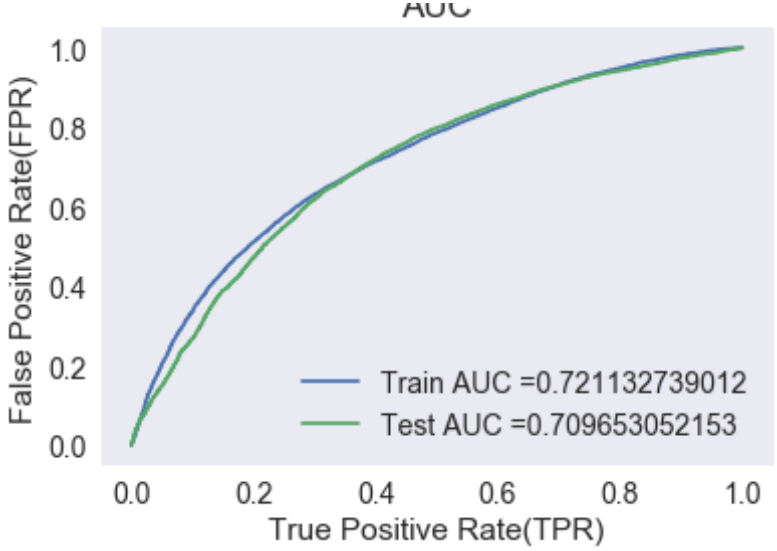
WE can observe that for the k value of -3 but it cannot take negative values so we can see max CV AUC and gap between the train and test AUC at k=0.5 somewhat lesser and it is feasible when compared to other values

In [66]:
```
#best value of k
best_k=0.5
```

In [67]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
```

```
nb_bow = MultinomialNB(alpha = best_k,class_prior = [0.5, 0.5])

nb_bow.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probabilit
y estimates of the positive class
# not the predicted outputs

y_train_pred =nb_bow.predict_log_proba(X_tr)[:,1]
y_cv_pred = nb_bow.predict_log_proba(X_cr)[:,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, t
rain_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_
tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```

AUC



In [68]:
```
#Confusion matrix
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(fpr*(1-tpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is
 very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for th
reshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [69]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
```

```
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, tr
ain_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test
_fpr, test_fpr)))
```

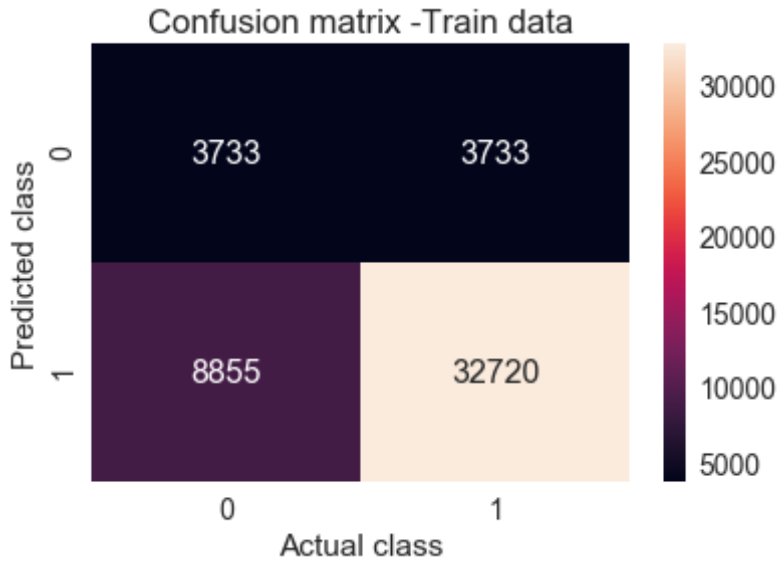
```
========================================================================
============================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold -0.791
[[ 3733  3733]
 [ 8855 32720]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold -0.66
[[ 3474  1974]
 [ 9691 20913]]
```

In [70]:
```
conf_matr_df_train_3 = pd.DataFrame(confusion_matrix(y_train, predict(y
_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold -0.791
```

In [71]:
```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_3, annot=True,annot_kws={"size": 16}, fm
t='g')


plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -Train data")
```

Out[71]:
```
Text(0.5,1,'Confusion matrix -Train data')
```

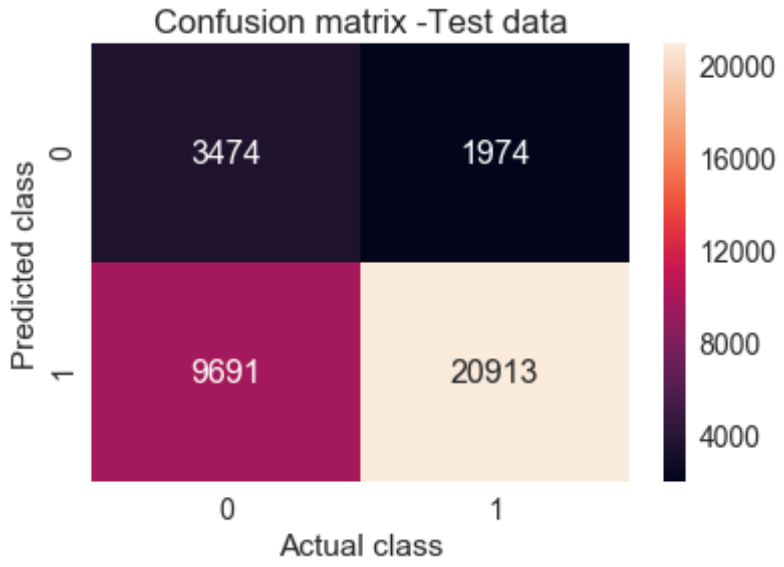


In [72]:
```
conf_matr_df_test_3 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold -0.66
```

In [73]:
```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_3, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -Test data")
```

Out[73]:
```
Text(0.5,1,'Confusion matrix -Test data')
```



2.4.2.1 Top 10 important features of positive class from SET 2

In [74]:
```
nb_tfidf= MultinomialNB(alpha = 4 ,class_prior = [0.5, 0.5])

nb_tfidf.fit(X_tr, y_train)
```

Out[74]:
```
MultinomialNB(alpha=4, class_prior=[0.5, 0.5], fit_prior=True)
```

In [75]:
```
len(nb_tfidf.feature_log_prob_[1,:])
```

Out[75]:
```
12320
```

In [76]:
```
tfidf_features_names=[]
```

In [77]:
```
# Please write all the code with proper documentation
for a in vectorizer_categories.get_feature_names() :
    tfidf_features_names.append(a)

for b in vectorizer_subcategories.get_feature_names() :
```

```
    tfidf_features_names.append(b)

for c in vectorizer_schoolstate.get_feature_names() :
    tfidf_features_names.append(c)

for d in vectorizer_project_grade_category.get_feature_names() :
    tfidf_features_names.append(d)

for e in vectorizer_teacher_prefix.get_feature_names() :
    tfidf_features_names.append(e)

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

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

tfidf_features_names.append('price')
```

In [78]:
```
len(tfidf_features_names)
```

Out[78]: 12320

In [79]:
```
pos_class_prob_sorted = nb_tfidf.feature_log_prob_[1, :].argsort()

pos_class_prob_sorted =pos_class_prob_sorted[::-1]
print(np.take(tfidf_features_names, pos_class_prob_sorted[:10]))
```

```
['00' 'c3' 'id' 'mr' 'la' 'subscription' 'd4' 'ms' 'tx' 'nj']
```

2.4.2.2 Top 10 important features of negative class from SET 2

In [80]:
```
neg_class_prob_sorted = nb_tfidf.feature_log_prob_[0, :].argsort()

neg_class_prob_sorted =neg_class_prob_sorted[::-1]
print(np.take(tfidf_features_names, neg_class_prob_sorted[:10]))
```

```
['00' 'c3' 'id' 'mr' 'la' 'subscription' 'd4' 'ms' 'nj' 'tx']
```

3. Conclusions

In [81]:
```
# Please compare all your models using Prettytable library

# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x12=PrettyTable()

x12.field_names = ["Vectorizer", "model", "hyperparameter", "Train AUC"
,"Test AUC"]

x12.add_row(["BOW", "Naive", 0.5, 0.79, 0.71])

x12.add_row(["TFIDF", "Naive", 0.5, 0.72 ,0.71])

print(x12)
```

```
+------------+-------+----------------+-----------+----------+
| Vectorizer | model | hyperparameter | Train AUC | Test AUC |
+------------+-------+----------------+-----------+----------+
|    BOW     | Naive |      0.5       |    0.79   |   0.71   |
|   TFIDF    | Naive |      0.5       |    0.72   |   0.71   |
+------------+-------+----------------+-----------+----------+
```

Conclusion:

Naive bayes seems to be function better while compared with KNN for both BOW model and TFIDF model and both Train AUC and Test AUC have been improved

Naive bayes takes very less time compared to KNN .

We can conclude that Naive works better than KNN