Objective: To train the model using Naive Baye's.

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
```

```
C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning:
 detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

Reading Data

In [2]:

```
project_data = pd.read_csv(r'C:\Users\Bhuvana Chandrahasan\train_data.csv')
resource_data = pd.read_csv(r'C:\Users\Bhuvana Chandrahasan\resources.csv')
```

In [3]:

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

```
Number of data points in train data (109248, 17)
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
```

```
her_prefix', 'school_state',
       'project_submitted_datetime', 'project_
grade_category',
       'project_subject_categories', 'project_
subject_subcategories',
       'project_title', 'project_essay_1', 'pr
oject_essay_2',
       'project_essay_3', 'project_essay_4', '
project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved'],
      dtype='object')
```

In [4]:

```
# how to replace elements in list python: https://stackoverfl
ow.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://stackove
rflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_s
ubmitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inpla
ce=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[4]:

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

In [5]:

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

```
Number of data points in resource data (154127
2, 4)
Number of data points in resource data Index([
'id', 'description', 'quantity', 'price'], dty
pe='object')
```

Out[5]:

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

In [6]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a
-dataframes-indexes-for-all-groups-in-one-step
```

```
price_data = resource_data.groupby('id').agg({'price':'sum',
'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[6]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [7]:

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

In [8]:

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

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

In [9]:

```
project_data.shape
```

Out[9]:

(109248, 19)

In [10]:

```
final_appr = project_data[project_data['project_is_approved']
 == 1]
final_appr = final_appr.sample(frac=0.35,random_state=1)
final_appr.shape
```

Out[10]:

(32447, 19)

In [11]:

```
final_rej =  project_data[project_data['project_is_approved']
 == 0]
final_rej = final_rej.sample(n=5000)
final_rej.shape
```

Out[11]:

(5000, 19)

In [12]:

```
final=pd.concat([final_appr,final_rej])
final=final.sort_values('Date', axis=0, ascending=True, inpla
ce=False, kind='quicksort', na_position='last')
final.shape
```

Out[12]:

(37447, 19)

1.1 Preprocessing of project_subject_categories

In [13]:

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

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

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

from collections import Counter
my_counter = Counter()
for word in final['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]))
```

1.2 Preprocessing of project_subject_subcategories

In [14]:

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

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth
, Care & Hunger"
    for j in i.split(','): # it will split it in three parts
["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            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 ' '(s
pace) 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())

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

# count of all the words in corpus python: https://stackoverf
low.com/a/22898595/4084039
my_counter = Counter()
for word in final['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=l
ambda kv: kv[1]))
```

2.Text Preprocessing

2.1 Preprocessing of essay

In [15]:

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

In [16]:

```
```

```
final.head(2)
```

Out[16]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	
4	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	

In [17]:

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

```
Imagine being 8-9 years old. You're in your th
ird grade classroom. You see bright lights, th
e kid next to you is chewing gum, the birds ar
e making noise, the street outside is buzzing
with cars, it's hot, and your teacher is askin
g you to focus on learning. Ack! You need a br
eak! So do my students.Most of my students hav
e autism, anxiety, another disability, or all
of the above. It is tough to focus in school d
```

ue to sensory overload or emotions. My student s have a lot to deal with in school, but I thi nk that makes them the most incredible kids on the planet. They are kind, caring, and sympat hetic. They know what it's like to be overwhel med, so they understand when someone else is s truggling. They are open-minded and compassion ate. They are the kids who will someday change the world.It is tough to do more than one thi ng at a time. When sensory overload gets in th e way, it is the hardest thing in the world to focus on learning. My students need many brea ks throughout the day, and one of the best ite ms we've used is a Boogie Board. If we had a f ew in our own classroom, my students could tak e a break exactly when they need one, regardle ss of which other rooms in the school are occu pied. Many of my students need to do something with their hands in order to focus on the tas k at hand. Putty will give the sensory input t hey need in order to focus, it will calm them when they are overloaded, it will help improve motor skills, and it will make school more fu n.When my students are able to calm themselves down, they are ready to learn. When they are able to focus, they will learn more and retain more. They will get the sensory input they ne ed and it will prevent meltdowns (which are sc ary for everyone in the room). This will lead to a better, happier classroom community that is able to learn the most they can in the best way possible.

==========

Have you ever set in a silent classroom trying to focus on your work? My class will begin hu mming, drumming, or anything to make music whi

le they work. If I can focus them with differe
nt types of music I will. One of the high scho
ol classes favorite types of music is Disney m
usic.Working in a military town at one of thre
e high schools in a classroom where we get no
money to help us with supplies. The small $25
fee that they pay does not even pay for all th
e supplies that we use. I will have over 300 s
tudents work in my classroom during next year
and they all agree to enjoy Disney music.When
students come into a silent classroom they hav
e to make music or drum. With music playing in
 the background, the students just zone into t
heir work and do not notice what or who is wal
king around. With some music in the background
 the students relax and just let their imagina
tion work and create art. By adding music it h
elps students to not be some critical of thems
elves or their art.Music can create a mood. By
 asking for a good set of speakers we can watc
h you tube videos about art or listen to music
. By adding to my collection of Disney music t
he students will have a larger amount of music
 that they all agree too. The music will set t
he mood for a more creative mood.
==========
====
Every morning I am greeted by smiling faces th
at show an eagerness to start a new day of lea
rning. Wether the teaching comes from me or e
ach other, my students are ready to learn new
things. Better yet, they are ready to put wha
t they have learned into practice.My students
are wonderful beings that are growing up in a
rough neighborhood. They might be considered
by some as not having a very bright future. H
owever, their futures are as bright as the sun

, moon, and stars put together. \r\n\r\nMy st
udents come to school everyday ready to learn.
  The are hungry to gain new knowledge and sho
w what they have learned. \r\n\r\nMy school i
s in a very low socio-economic neighborhood.
Many of our families are very impoverished, th
us making it hard to provide adequately for th
eir families. However, just like me, they are
 very interested in their child's success.My s
tudents will use the Chromebook individually t
o practice their literacy skills, as well as m
ath skills. We have a variety of online progr
ams as school for students to use, but we do n
ot have an adequate number of technology tools
 to acces them. In having this resource, my st
udents can access these programs, such as ST M
ath and websites like ABC Mouse and Starfall.\
r\n\r\n It would be great to have at least one
 Chromebook in the classroom for my students t
o use throughout the day and on a daily basis.
  These eager little minds should not have to
wait an entire week to use the computers in th
e computer lab for one hour.Donations towards
this project will allow my students to have in
-class access to technology. Obtaining these
resources will allow me to set up a \"Technolo
gy Center\" where students have access to onli
ne educational programs that will support and
enhance their learning. \r\n\r\nMany of my st
udents have no access to a computer/tablet at
home.j Giving them an opportunity to handle t
hese resources will allow them to be ready for
 future use in school and in the workplace whe
n they are older.
==========
====
A typical day in our classroom starts with a d

elicious breakfast provided by our wonderful s
chool kitchen. Our school district is 100% fre
e breakfast and lunch to all students because
we understand that hungry minds need full bell
ies.\r\nOur school is a wonderful community of
 diverse families who support each other. My s
tudents are kind, responsible, and respectful.
 They uphold these titles with pride and can b
e seen daily trying to make our school and cla
ssroom a better place. We provide a nurturing
environment that creates good citizens, indepe
ndent thinkers, and lifelong learners. We are
a lively group of students who are eager to le
arn new things with innovative tools.With the
LittleBits and Makey Makey Invention Kit, my s
tudents will be able to design, build, and pro
gram their own electronic inventions! The colo
r-coded electronics will make it possible for
all of my students to learn basic programming
skills. My students will be able to collabora
te with each other and use their creativity to
 invent their own projects. We will be using t
he iPad to help research our ideas as well as
continue our learning using the Osmo programin
g of early concepts of coding. \r\nToday's stu
dents are the scientists, engineers, and inven
tors of the future. They need our support toda
y to become the success stories of tomorrow. I
 know they will amaze us with their accomplish
ments! The future begins now, and we're ready
to take learning to the next level starting ri
ght in Kindergarten!nannan
==========
====

In [18]:

```
# 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 [19]:

```
sent = decontracted(final['essay'].values[20000])
print(sent)
print("="*50)
```

```
A typical day in our classroom starts with a d
elicious breakfast provided by our wonderful s
chool kitchen. Our school district is 100% fre
e breakfast and lunch to all students because
we understand that hungry minds need full bell
ies.\r\nOur school is a wonderful community of
 diverse families who support each other. My s
tudents are kind, responsible, and respectful.
 They uphold these titles with pride and can b
e seen daily trying to make our school and cla
ssroom a better place. We provide a nurturing
environment that creates good citizens, indepe
```

```
ndent thinkers, and lifelong learners. We are
a lively group of students who are eager to le
arn new things with innovative tools.With the
LittleBits and Makey Makey Invention Kit, my s
tudents will be able to design, build, and pro
gram their own electronic inventions! The colo
r-coded electronics will make it possible for
all of my students to learn basic programming
skills.  My students will be able to collabora
te with each other and use their creativity to
 invent their own projects. We will be using t
he iPad to help research our ideas as well as
continue our learning using the Osmo programin
g of early concepts of coding. \r\nToday is st
udents are the scientists, engineers, and inve
ntors of the future. They need our support tod
ay to become the success stories of tomorrow.
I know they will amaze us with their accomplis
hments! The future begins now, and we are read
y to take learning to the next level starting
right in Kindergarten!nannan
================================================
====
```

In [20]:

```
# \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 typical day in our classroom starts with a d
elicious breakfast provided by our wonderful s
chool kitchen. Our school district is 100% fre
e breakfast and lunch to all students because
we understand that hungry minds need full bell
```

```
ies.  Our school is a wonderful community of d
iverse families who support each other. My stu
dents are kind, responsible, and respectful. T
hey uphold these titles with pride and can be
seen daily trying to make our school and class
room a better place. We provide a nurturing en
vironment that creates good citizens, independ
ent thinkers, and lifelong learners. We are a
lively group of students who are eager to lear
n new things with innovative tools.With the Li
ttleBits and Makey Makey Invention Kit, my stu
dents will be able to design, build, and progr
am their own electronic inventions! The color-
coded electronics will make it possible for al
l of my students to learn basic programming sk
ills.  My students will be able to collaborate
 with each other and use their creativity to i
nvent their own projects. We will be using the
 iPad to help research our ideas as well as co
ntinue our learning using the Osmo programing
of early concepts of coding.  Today is studen
ts are the scientists, engineers, and inventor
s of the future. They need our support today t
o become the success stories of tomorrow. I kn
ow they will amaze us with their accomplishmen
ts! The future begins now, and we are ready to
 take learning to the next level starting righ
t in Kindergarten!nannan
```

In [21]:

```
#remove spacial character: https://stackoverflow.com/a/584354
7/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
A typical day in our classroom starts with a d
elicious breakfast provided by our wonderful s
```

```
chool kitchen Our school district is 100 free
breakfast and lunch to all students because we
 understand that hungry minds need full bellie
s Our school is a wonderful community of diver
se families who support each other My students
 are kind responsible and respectful They upho
ld these titles with pride and can be seen dai
ly trying to make our school and classroom a b
etter place We provide a nurturing environment
 that creates good citizens independent thinke
rs and lifelong learners We are a lively group
 of students who are eager to learn new things
 with innovative tools With the LittleBits and
 Makey Makey Invention Kit my students will be
 able to design build and program their own el
ectronic inventions The color coded electronic
s will make it possible for all of my students
 to learn basic programming skills My students
 will be able to collaborate with each other a
nd use their creativity to invent their own pr
ojects We will be using the iPad to help resea
rch our ideas as well as continue our learning
 using the Osmo programing of early concepts o
f coding Today is students are the scientists
engineers and inventors of the future They nee
d our support today to become the success stor
ies of tomorrow I know they will amaze us with
 their accomplishments The future begins now a
nd we are ready to take learning to the next l
evel starting right in Kindergarten nannan
```

In [22]:

```
# 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', 'b
etween', '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', "co
uldn'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 [23]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(final['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.append(sent.lower().strip())
```

```
100%|████████████████████████████████████| 3744
7/37447 [01:30<00:00, 402.76it/s]
```

In [24]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[24]:

```
'typical day classroom starts delicious breakf
ast provided wonderful school kitchen school d
istrict 100 free breakfast lunch students unde
rstand hungry minds need full bellies school w
onderful community diverse families support st
udents kind responsible respectful uphold titl
es pride seen daily trying make school classro
om better place provide nurturing environment
creates good citizens independent thinkers lif
elong learners lively group students eager lea
rn new things innovative tools littlebits make
y makey invention kit students able design bui
ld program electronic inventions color coded e
lectronics make possible students learn basic
programming skills students able collaborate u
se creativity invent projects using ipad help
research ideas well continue learning using os
mo programing early concepts coding today stud
ents scientists engineers inventors future nee
```

```
d support today become success stories tomorro
w know amaze us accomplishments future begins
ready take learning next level starting right
kindergarten nannan'
```

2.2 Preprocessing of project_title

In [25]:

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

```
Sensory Tools for Focus
==================================================
====
Setting the Creative Mood with Music
==================================================
====
Technology Is Our Future
==================================================
====
Little Engineers Need Tools Too
==================================================
====
```

In [26]:

```
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 [27]:

```
sent = decontracted(final['project_title'].values[20000])
print(sent)
print("="*50)
```

```
Little Engineers Need Tools Too
==================================================
====
```

In [28]:

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

```
Little Engineers Need Tools Too
```

In [29]:

```
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
```

Little Engineers Need Tools Too

In [30]:

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

```
100%|██████████████████████████████████| 37447
/37447 [00:05<00:00, 6348.52it/s]
```

In [31]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[31]:

'little engineers need tools'

In [32]:

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

Out[32]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_

0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.

In [33]:

```
print(final.shape)
print(final.columns)
```

```
(37447, 20)
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
her_prefix', 'school_state',
       'Date', 'project_grade_category', 'proj
ect_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', '
project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'price', 'quantity', 'clean_categories'
, 'clean_subcategories',
       'essay'],
      dtype='object')
```

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

In [34]:

```
Y = final['project_is_approved'].values
X = final
```

In [35]:

```
final.shape
```

Out[35]:

```
(37447, 20)
```

In [36]:

```
# 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(X, Y, test_size=0.33,stratify=Y) # this is random splitting
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33,stratify=y_train) # this is random splitting
```

In [37]:

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

```
(16809, 20) (16809,)
(8280, 20) (8280,)
(12358, 20) (12358,)
```

4.Vectorizing Text data

4.1 Essay

In [38]:

```
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max
_features=5000)
vectorizer.fit(X_train['essay'].values) # fit has to happen o
nly on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_essay_bow = vectorizer.transform(X_train['essay'].val
ues)
X_cv_essay_bow = vectorizer.transform(X_cv['essay'].values)
X_test_essay_bow = vectorizer.transform(X_test['essay'].value
s)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
```

```
After vectorizations
(16809, 5000) (16809,)
(8280, 5000) (8280,)
(12358, 5000) (12358,)
```

4.2 Project_title

In [39]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10, max_features=5000)
vectorizer.fit(X_train['project_title'].values) # fit has to
happen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
```

```
X_train_title_bow = vectorizer.transform(X_train['project_tit
le'].values)
X_cv_title_bow = vectorizer.transform(X_cv['project_title'].v
alues)
X_test_title_bow = vectorizer.transform(X_test['project_title
'].values)

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
```

```
After vectorizations
(16809, 1008) (16809,)
(8280, 1008) (8280,)
(12358, 1008) (12358,)
```

4.3 Project_resource_summary

In [40]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10, max_features=5000)
vectorizer.fit(X_train['project_resource_summary'].values) #
fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_res_sum_bow = vectorizer.transform(X_train['project_r
esource_summary'].values)
X_cv_res_sum_bow = vectorizer.transform(X_cv['project_resourc
e_summary'].values)
X_test_res_sum_bow = vectorizer.transform(X_test['project_res
ource_summary'].values)

print("After vectorizations")
```

```
print(X_train_res_sum_bow.shape, y_train.shape)
print(X_cv_res_sum_bow.shape, y_cv.shape)
print(X_test_res_sum_bow.shape, y_test.shape)
```

```
After vectorizations
(16809, 2153) (16809,)
(8280, 2153) (8280,)
(12358, 2153) (12358,)
```

5.Catogorical features:one hot encoding

5.1 Clean_categories

In [41]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_category_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_cv_clean_category_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_clean_category_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_clean_category_ohe.shape, y_train.shape)
print(X_cv_clean_category_ohe.shape, y_cv.shape)
print(X_test_clean_category_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(16809, 9) (16809,)
(8280, 9) (8280,)
(12358, 9) (12358,)
['appliedlearning', 'care_hunger', 'health_spo
rts', 'history_civics', 'literacy_language', '
math_science', 'music_arts', 'specialneeds', '
warmth']
```

5.2 Clean_subcategories

In [42]:

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

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_clean_subcategory_ohe = vectorizer.transform(X_train[
'clean_subcategories'].values)
X_cv_clean_subcategory_ohe = vectorizer.transform(X_cv['clean
_subcategories'].values)
X_test_clean_subcategory_ohe = vectorizer.transform(X_test['c
lean_subcategories'].values)

print("After vectorizations")
print(X_train_clean_subcategory_ohe.shape, y_train.shape)
print(X_cv_clean_subcategory_ohe.shape, y_cv.shape)
print(X_test_clean_subcategory_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(16809, 30) (16809,)
(8280, 30) (8280,)
```

```
(12358, 30) (12358,)
['appliedsciences', 'care_hunger', 'charactere
ducation', 'civics_government', 'college_caree
rprep', 'communityservice', 'earlydevelopment'
, 'economics', 'environmentalscience', 'esl',
'extracurricular', 'financialliteracy', 'forei
gnlanguages', 'gym_fitness', 'health_lifescien
ce', 'health_wellness', 'history_geography', '
literacy', 'literature_writing', 'mathematics'
, 'music', 'nutritioneducation', 'other', 'par
entinvolvement', 'performingarts', 'socialscie
nces', 'specialneeds', 'teamsports', 'visualar
ts', 'warmth']
```

5.3 Teacher_prefix

In [43]:

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

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_p
refix'].values)
X_cv_teacher_ohe = vectorizer.transform(X_cv['teacher_prefix'
].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_pre
fix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(16809, 5) (16809,)
(8280, 5) (8280,)
(12358, 5) (12358,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
```

5.4 School_state

In [44]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(16809, 51) (16809,)
(8280, 51) (8280,)
(12358, 51) (12358,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc
', 'de', 'fl', 'ga', 'hi', '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']
```

5.5 Project_grade_category

In [45]:

```
X_train.project_grade_category = X_train.project_grade_catego
ry.str.replace('\s+', '_')
X_train.project_grade_category = X_train.project_grade_catego
ry.str.replace('-', '_')
X_train['project_grade_category'].value_counts()
```

Out[45]:

```
Grades_PreK_2    6753
Grades_3_5       5820
Grades_6_8       2573
Grades_9_12      1663
Name: project_grade_category, dtype: int64
```

In [46]:

```
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values) # fi
t has to happen only on train data
# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_grade_ohe = vectorizer.transform(X_train['project_gra
de_category'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['project_grade_cat
egory'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade
_category'].values)
```

```
print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(16809, 4) (16809,)
(8280, 4) (8280,)
(12358, 4) (12358,)
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'G
rades_PreK_2']
```

6.Numerical features

6.1 Price

In [47]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['price'].values.reshape(-1,1))

X_train_price_std = standard_vec.transform(X_train['price'].v
alues.reshape(-1,1))
X_cv_price_std = standard_vec.transform(X_cv['price'].values.
reshape(-1,1))
```

```
X_test_price_std = standard_vec.transform(X_test['price'].val
ues.reshape(-1,1))

print("After vectorizations")
print(X_train_price_std.shape, y_train.shape)
print(X_cv_price_std.shape, y_cv.shape)
print(X_test_price_std.shape, y_test.shape)
```

```
After vectorizations
(16809, 1) (16809,)
(8280, 1) (8280,)
(12358, 1) (12358,)
```

6.2 Teacher_number_of_previously_posted_projects

In [48]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['teacher_number_of_previously_posted
_projects'].values.reshape(-1,1))

X_train_prev_projects_std = standard_vec.transform(X_train['t
eacher_number_of_previously_posted_projects'].values.reshape(-
1,1))
X_cv_prev_projects_std = standard_vec.transform(X_cv['teacher
_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_prev_projects_std = standard_vec.transform(X_test['tea
cher_number_of_previously_posted_projects'].values.reshape(-1,
```

```
1))

print("After vectorizations")
print(X_train_prev_projects_std.shape, y_train.shape)
print(X_cv_prev_projects_std.shape, y_cv.shape)
print(X_test_prev_projects_std.shape, y_test.shape)
```

```
C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\si
te-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to f
loat64 by StandardScaler.

C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\si
te-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to f
loat64 by StandardScaler.

C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\si
te-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to f
loat64 by StandardScaler.

C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\si
te-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to f
loat64 by StandardScaler.
```

```
After vectorizations
```

```
(16809, 1) (16809,)
(8280, 1) (8280,)
(12358, 1) (12358,)
```

6.3 Quantity

In [49]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['quantity'].values.reshape(-1,1))

X_train_quantity_std = standard_vec.transform(X_train['quanti
ty'].values.reshape(-1,1))
X_cv_quantity_std = standard_vec.transform(X_cv['quantity'].v
alues.reshape(-1,1))
X_test_quantity_std = standard_vec.transform(X_test['quantity
'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_quantity_std.shape, y_train.shape)
print(X_cv_quantity_std.shape, y_cv.shape)
print(X_test_quantity_std.shape, y_test.shape)
```

```
C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\si
te-packages\sklearn\utils\validation.py:595: D
ataConversionWarning:

Data with input dtype int64 was converted to f
loat64 by StandardScaler.
```

```
C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Bhuvana Chandrahasan\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

```
After vectorizations
(16809, 1) (16809,)
(8280, 1) (8280,)
(12358, 1) (12358,)
```

7.Multinomial NaiveBayes

7.1 Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

Merging all the above features

In [50]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_train_bow = hstack((X_train_essay_bow,X_train_title_bow,X_train_res_sum_bow,X_train_clean_category_ohe,X_train_clean_subcategory_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_std,X_train_prev_projects_std,X_train_quantity_std)).tocsr()
X_cv_bow = hstack((X_cv_essay_bow,X_cv_title_bow,X_cv_res_sum_bow,X_cv_clean_category_ohe,X_cv_clean_subcategory_ohe, X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_price_std, X_cv_prev_projects_std,X_cv_quantity_std)).tocsr()
X_test_bow = hstack((X_test_essay_bow,X_test_title_bow,X_test_res_sum_bow,X_test_clean_category_ohe,X_test_clean_subcategory_ohe, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_price_std,X_test_prev_projects_std,X_test_quantity_std)).tocsr()

print("Final Data matrix")
print(X_train_bow.shape, y_train.shape)
print(X_cv_bow.shape, y_cv.shape)
```

```
print(X_test_bow.shape, y_test.shape)
```

```
Final Data matrix
(16809, 8263) (16809,)
(8280, 8263) (8280,)
(12358, 8263) (12358,)
```

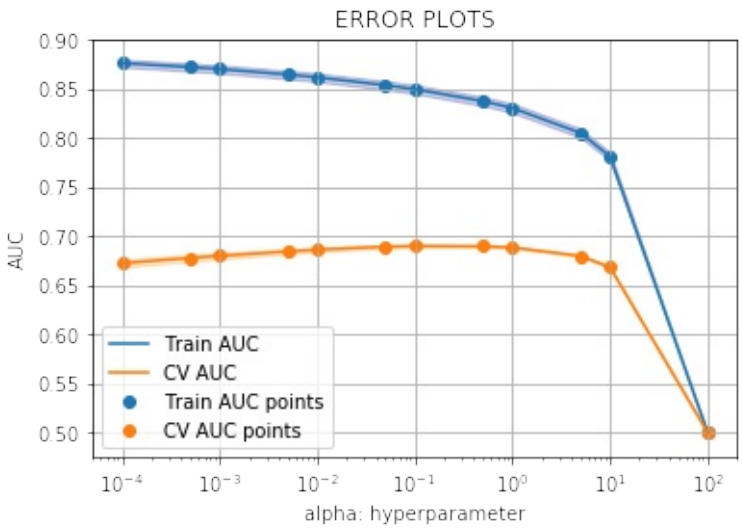
In [51]:

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

    y_data_pred = []
    tr_loop = final.shape[0] - final.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop wil
l be 49041 - 49041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 mul
tiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_log_proba(final[i:i+10
00])[:,1])
    # we will be predicting for the last data points

    y_data_pred.extend(clf.predict_log_proba(final[tr_loop:])
[:,1])

    return y_data_pred
```

In [52]:

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

```
NB_BOW = MultinomialNB()
parameters = {'alpha':[100,10,5,1,0.5,0.1,0.05,0.01,0.005,0.0
01,0.0005,0.0001]}
clf = GridSearchCV(NB_BOW, parameters, cv=3, scoring='roc_auc
')
clf.fit(X_train_bow, 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 AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,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.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [53]:

```
cv_auc = list(cv_auc)
```

In [54]:

```
alpha= [100,10,5,1,0.5,0.1,0.05,0.01,0.005,0.001,0.0005,0.000
1]
best_auc = alpha[cv_auc.index(max(cv_auc))]
print(best_auc)
```

```
0.1
```

from the error plot we choose K such that, we will have maximum AUC on cv data and gap between the train and cv is less

Note: based on the method you use you might get different hyperparameter values as best one

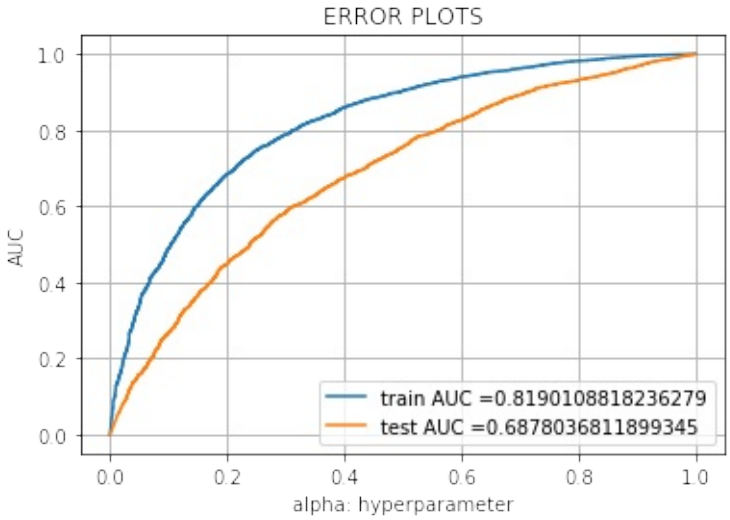
so, you choose according to the method you choose, you use gridsearch if you are having more computing power and note it will take more time

if you increase the cv values in the GridSearchCV you will get more rebust results.

here we are choosing the best_k based on forloop results
best_auc = 0.1

In [55]:

```
# 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=0.01)
NB_BOW.fit(X_train_bow, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(NB_BOW, X_train_bow)
y_test_pred = batch_predict(NB_BOW, X_test_bow)

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("ERROR PLOTS")
plt.grid()
plt.show()
```



In [56]:

```
# we are writing our own function for predict, with defined t
hresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [57]:

```
from sklearn.metrics import confusion_matrix
```

```
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

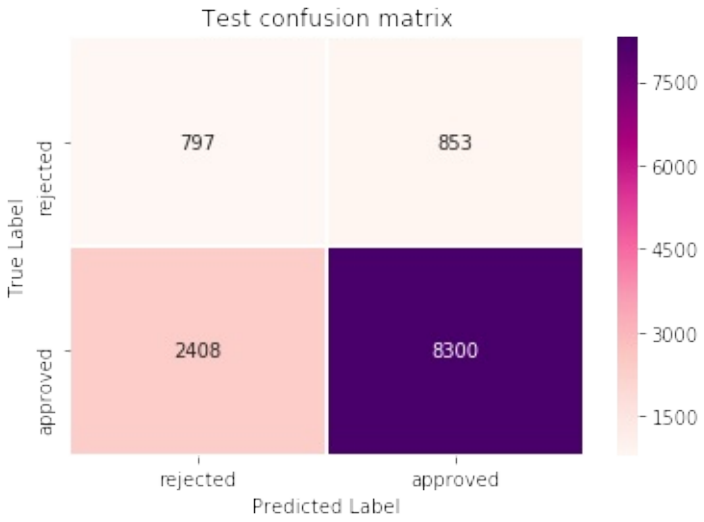
```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -4.646
[[ 1122  1122]
 [ 1419 13146]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.707
[[ 795  855]
 [2400 8308]]
```

In [58]:

```
label = ["rejected", "approved"]
frame_confusion_train = pd.DataFrame(confusion_matrix(y_train
, NB_BOW.predict(X_train_bow)), index = label, columns = labe
l)
frame_confusion_test = pd.DataFrame(confusion_matrix(y_test,
NB_BOW.predict(X_test_bow)), index = label, columns = label)
```

In [59]:

```
sns.heatmap(frame_confusion_test, annot = True, fmt="d", cmap
="RdPu", linewidths=.5)
plt.title("Test confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



Feature Importance

In [60]:

```
feature_bow = [X_train_essay_bow,X_train_title_bow,X_train_res_sum_bow,X_train_clean_category_ohe,X_train_clean_subcategory_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, 'X_train_price_std','X_train_prev_projects_std','X_train_quantity_std']
```

In [61]:

```
type(feature_bow)
```

Out[61]:

```
list
```

In [62]:

```
#https://stackoverflow.com/questions/29867367/sklearn-multinomial-nb-most-informative-features
def show_most_informative_features(feature_names, clf, n=20):
    feature_names = vectorizer.get_feature_names()
```

```
    coefs_with_fns = sorted(zip(clf.coef_[0], feature_names))
    top = zip(coefs_with_fns[:n], coefs_with_fns[:-(n + 1):-1
])
    for (coef_1, fn_1), (coef_2, fn_2) in top:
        print("\t%.4f\t%-15s\t\t%.4f\t%-15s" % (coef_1, fn_1,
 coef_2, fn_2))
```

In [63]:

```
show_most_informative_features(feature_bow,NB_BOW)
```

```
        -9.7403 Grades_9_12             -8.821
1       Grades_6_8
        -9.5380 Grades_PreK_2           -9.293
4       Grades_3_5
        -9.2934 Grades_3_5              -9.538
0       Grades_PreK_2
        -8.8211 Grades_6_8              -9.740
3       Grades_9_12
```

Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

In [64]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['essay'].values) # fit has to happen o
nly on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].v
alues)
X_cv_essay_tfidf = vectorizer.transform(X_cv['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].val
```

```
ues)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
```

```
After vectorizations
(16809, 8102) (16809,)
(8280, 8102) (8280,)
(12358, 8102) (12358,)
```

In [65]:

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

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_title_tfidf = vectorizer.transform(X_train['project_t
itle'].values)
X_cv_title_tfidf = vectorizer.transform(X_cv['project_title']
.values)
X_test_title_tfidf = vectorizer.transform(X_test['project_tit
le'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
```

```
After vectorizations
(16809, 1008) (16809,)
(8280, 1008) (8280,)
(12358, 1008) (12358,)
```

In [66]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_res_sum_tfidf = vectorizer.transform(X_train['project_resource_summary'].values)
X_cv_res_sum_tfidf = vectorizer.transform(X_cv['project_resource_summary'].values)
X_test_res_sum_tfidf = vectorizer.transform(X_test['project_resource_summary'].values)

print("After vectorizations")
print(X_train_res_sum_tfidf.shape, y_train.shape)
print(X_cv_res_sum_tfidf.shape, y_cv.shape)
print(X_test_res_sum_tfidf.shape, y_test.shape)
```

```
After vectorizations
(16809, 2153) (16809,)
(8280, 2153) (8280,)
(12358, 2153) (12358,)
```

In [67]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_train_tfidf = hstack((X_train_essay_tfidf,X_train_title_tfidf,X_train_res_sum_tfidf,X_train_clean_category_ohe,X_train_clean_subcategory_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_std,X_train_prev_projects_std,X_train_quantity_std)).tocsr()
X_cv_tfidf = hstack((X_cv_essay_tfidf,X_cv_title_tfidf,X_cv_res_sum_tfidf,X_cv_clean_category_ohe,X_cv_clean_subcategory_ohe, X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_price_std,X_cv_prev_projects_std,X_cv_quantity_std)).tocsr()
X_test_tfidf = hstack((X_test_essay_tfidf,X_test_title_tfidf,
```

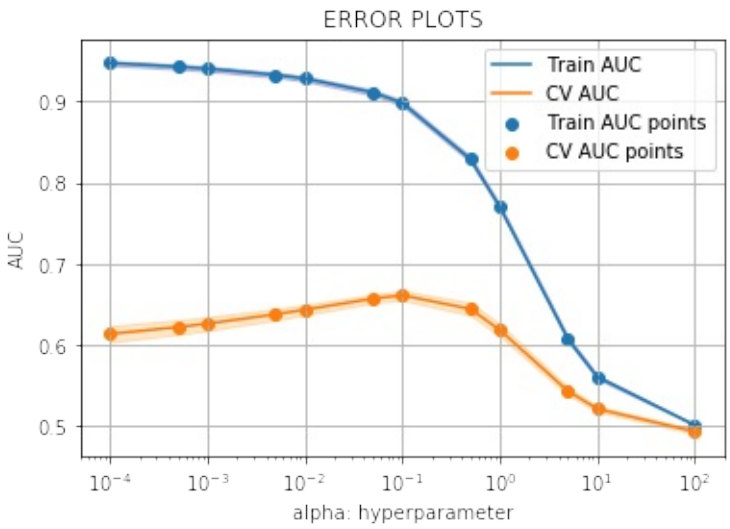
```
X_test_res_sum_tfidf,X_test_clean_category_ohe,X_test_clean_s
ubcategory_ohe, X_test_state_ohe, X_test_teacher_ohe, X_test_
grade_ohe, X_test_price_std,X_test_prev_projects_std,X_test_q
uantity_std)).tocsr()

print("Final Data matrix")
print(X_train_tfidf.shape, y_train.shape)
print(X_cv_tfidf.shape, y_cv.shape)
print(X_test_tfidf.shape, y_test.shape)
```

```
Final Data matrix
(16809, 11365) (16809,)
(8280, 11365) (8280,)
(12358, 11365) (12358,)
```

In [68]:

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


NB_TFIDF = MultinomialNB()
parameters = {'alpha':[100,10,5,1,0.5,0.1,0.05,0.01,0.005,0.0
01,0.0005,0.0001]}
clf = GridSearchCV(NB_BOW, parameters, cv=3, scoring='roc_auc
')
clf.fit(X_train_tfidf, 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 AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
```

```
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,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.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [69]:

```
cv_auc = list(cv_auc)
```

In [70]:

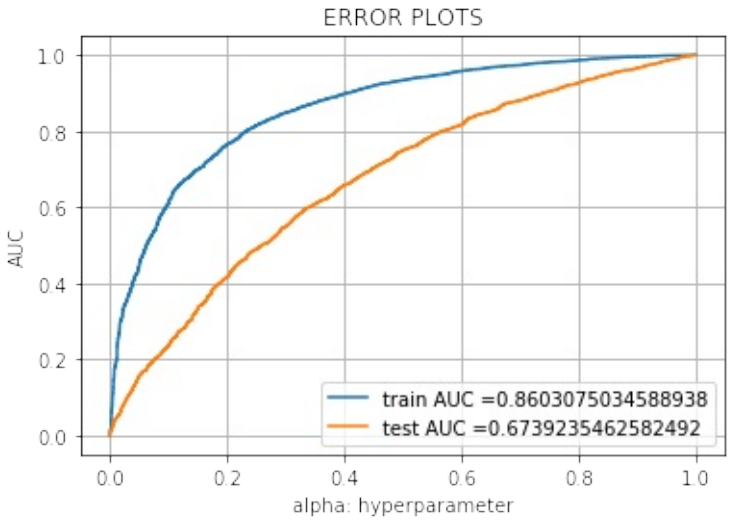
```
best_auc = alpha[cv_auc.index(max(cv_auc))]
print(best_auc)
```

```
0.1
```

In [71]:

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


NB_TFIDF = MultinomialNB(alpha=0.1)
NB_TFIDF.fit(X_train_tfidf, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(NB_TFIDF, X_train_tfidf)
y_test_pred = batch_predict(NB_TFIDF, X_test_tfidf)

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("ERROR PLOTS")
```

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



In [72]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

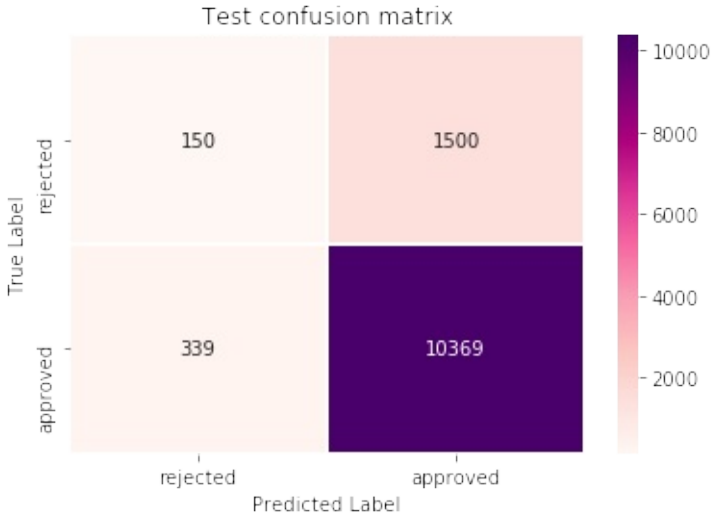
```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999980141
140885 for threshold -0.397
[[ 1121  1123]
 [  970 13595]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.271
[[ 549 1101]
 [1448 9260]]
```

In [73]:

```
label = ["rejected", "approved"]
frame_confusion_train = pd.DataFrame(confusion_matrix(y_train
, NB_TFIDF.predict(X_train_tfidf)), index = label, columns =
label)
frame_confusion_test = pd.DataFrame(confusion_matrix(y_test,
NB_TFIDF.predict(X_test_tfidf)), index = label, columns = lab
el)
```

In [74]:

```
sns.heatmap(frame_confusion_test, annot = True, fmt="d", cmap
="RdPu", linewidths=.5)
plt.title("Test confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



Feature Importance

In [75]:

```
feature_tfidf = [X_train_essay_tfidf,X_train_title_tfidf,X_tr
ain_res_sum_tfidf,X_train_clean_category_ohe,X_train_clean_su
```

```
bcategory_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, 'X_train_price_std','X_train_prev_projects_std', 'X_train_quantity_std']
```

In [76]:

```
show_most_informative_features(feature_tfidf,NB_TFIDF)
```

```
        -12.9284        graphics
        -5.0290 compete
        -12.9217        two
        -5.8968 cycle
        -12.9195        enough
        -6.3515 extension
        -12.9195        awareness
        -6.4809 no
        -12.9091        progress
        -6.5082 diffuser
        -12.8708        cds
        -6.7107 keyboards
        -12.8404        alternatives
        -6.8187 dress
        -12.8151        bars
        -6.8245 graphing
        -12.7709        long
        -6.8452 class
        -12.7501        and
        -6.9954 comfortably
        -12.7497        charge
        -7.0208 strengthen
        -12.7492        working
        -7.0309 nets
        -12.7419        round
        -7.0377 assortment
        -12.7247        called
        -7.0857 pencil
        -12.7016        breaks
        -7.1506 integrate
```

```
-12.6957        rack
-7.1724 internet
-12.6822        requesting
-7.1749 at
-12.6817        phonics
-7.1836 clorox
-12.6642        workbooks
-7.2887 mobile
-12.6623        grow
-7.3105 classmates
```

Summary

Pretty Table

In [77]:

```
#http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer","Hyperparameter", "AUC"]
x.add_row(["Bag of Words", 0.01, 0.6878])
x.add_row(["TFIDF", 0.1, 0.6738])

print(x)
```

```
+--------------+----------------+--------+
|  Vectorizer  | Hyperparameter |  AUC   |
+--------------+----------------+--------+
| Bag of Words |      0.01      | 0.6878 |
|    TFIDF     |      0.1       | 0.6738 |
+--------------+----------------+--------+
```