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	
`project_id`	A unique identifier for the proposed proje
`project_title`	Title • `Art Wi` •
`project_grade_category`	Grade level of students for which the project is targe • • • •

Feature	
project_subject_categories	One or more (comma-separated) subject categorie following e • • • • • I • • • • • • • Literacy & Langua
school_state	State where school is located (Tw (https://en.wikipedia.org/wiki/List_of_U.S._state_abbr
project_subject_subcategories	One or more (comma-separated) subject subc • • Literature & Writin
project_resource_summary	An explanation of the resources needed f • My students need hands on literacy m se
project_essay_1	
project_essay_2	S
project_essay_3	
project_essay_4	
project_submitted_datetime	Datetime when project application was submitted. E
teacher_id	A unique identifier for the teacher of the pro bdf8baa8fedef6
teacher_prefix	Teacher's title. One of the follo • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submi

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

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

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

In [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

1.2 preprocessing of `project_subject_categories`

In [5]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverf

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverf

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunge
    for j in i.split(','): # it will split it in three parts ["Math & Scien
        if 'The' in j.split(): # this will split each of the catogory based
            j=j.replace('The','') # if we have the words "The" we are going
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

# count of all the words in corpus python: https://stackoverflow.com/a/2289
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 [7]:
```
# 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 [8]:
```
project_data.head(2)
```

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [9]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [10]:
```
# 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)
```

```
My students are English learners that are working on English as their sec
ond or third languages. We are a melting pot of refugees, immigrants, and
native-born Americans bringing the gift of language to our school. \r\n\r
\n We have over 24 languages represented in our English Learner program w
ith students at every level of mastery.  We also have over 40 countries r
epresented with the families within our school.  Each student brings a we
alth of knowledge and experiences to us that open our eyes to new culture
s, beliefs, and respect.\"The limits of your language are the limits of y
our world.\"-Ludwig Wittgenstein  Our English learner's have a strong sup
port system at home that begs for more resources.  Many times our parents
are learning to read and speak English along side of their children.  Som
etimes this creates barriers for parents to be able to help their child l
earn phonetics, letter recognition, and other reading skills.\r\n\r\nBy p
roviding these dvd's and players, students are able to continue their mas
tery of the English language even if no one at home is able to assist.  A
ll families with students within the Level 1 proficiency status, will be
a offered to be a part of this program.  These educational videos will be
specially chosen by the English Learner Teacher and will be sent home reg
ularly to watch.  The videos are to help the child develop early reading
skills.\r\n\r\nParents that do not have access to a dvd player will have
the opportunity to check out a dvd player to use for the year.  The plan
is to use these videos and educational dvd's for the years to come for ot
her EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this yea
r all love learning, at least most of the time. At our school, 97.3% of t
he students receive free or reduced price lunch. Of the 560 students, 97.
3% are minority students. \r\nThe school has a vibrant community that lov
es to get together and celebrate. Around Halloween there is a whole schoo
l parade to show off the beautiful costumes that students wear. On Cinco
de Mayo we put on a big festival with crafts made by the students, dance
s, and games. At the end of the year the school hosts a carnival to celeb
rate the hard work put in during the school year, with a dunk tank being
the most popular activity.My students will use these five brightly colore
d Hokki stools in place of regular, stationary, 4-legged chairs. As I wil
l only have a total of ten in the classroom and not enough for each stude
nt to have an individual one, they will be used in a variety of ways. Dur
ing independent reading time they will be used as special chairs students
will each use on occasion. I will utilize them in place of chairs at my s
mall group tables during math and reading times. The rest of the day they
will be used by the students who need the highest amount of movement in t
heir life in order to stay focused on school.\r\n\r\nWhenever asked what
the classroom is missing, my students always say more Hokki Stools. They
can't get their fill of the 5 stools we already have. When the students a
re sitting in group with me on the Hokki Stools, they are always moving,
```

but at the same time doing their work. Anytime the students get to pick w
here they can sit, the Hokki Stools are the first to be taken. There are
always students who head over to the kidney table to get one of the stool
s who are disappointed as there are not enough of them. \r\n\r\nWe ask a
lot of students to sit for 7 hours a day. The Hokki stools will be a comp
romise that allow my students to do desk work and move at the same time.
These stools will help students to meet their 60 minutes a day of movemen
t by allowing them to activate their core muscles for balance while they
sit. For many of my students, these chairs will take away the barrier tha
t exists in schools for a child who can't sit still.nannan
==
How do you remember your days of school? Was it in a sterile environment
with plain walls, rows of desks, and a teacher in front of the room? A ty
pical day in our room is nothing like that. I work hard to create a warm
inviting themed room for my students look forward to coming to each da
y.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of
mixed races in Arkansas.\r\nThey attend a Title I school, which means the
re is a high enough percentage of free and reduced-price lunch to qualif
y. Our school is an \"open classroom\" concept, which is very unique as t
here are no walls separating the classrooms. These 9 and 10 year-old stud
ents are very eager learners; they are like sponges, absorbing all the in
formation and experiences and keep on wanting more.With these resources s
uch as the comfy red throw pillows and the whimsical nautical hanging dec
or and the blue fish nets, I will be able to help create the mood in our
classroom setting to be one of a themed nautical environment. Creating a
classroom environment is very important in the success in each and every
child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teach
er evening. I'll take pictures of each child with them, have them develop
ed, and then hung in our classroom ready for their first day of 4th grad
e. This kind gesture will set the tone before even the first day of scho
ol! The nautical thank you cards will be used throughout the year by the
students as they create thank you cards to their team groups.\r\n\r\nYour
generous donations will help me to help make our classroom a fun, invitin
g, learning environment from day one.\r\n\r\nIt costs lost of money out o
f my own pocket on resources to get our classroom ready. Please consider
helping with this project to make our new school year a very successful o
ne. Thank you!nannan
==
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. Th
ey are eager beavers and always strive to work their hardest working past
their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students r
eceive free or reduced price lunch. Despite their disabilities and limit
ations, my students love coming to school and come eager to learn and exp
lore.Have you ever felt like you had ants in your pants and you needed to
groove and move as you were in a meeting? This is how my kids feel all th
e time. The want to be able to move as they learn or so they say.Wobble c
hairs are the answer and I love then because they develop their core, whi
ch enhances gross motor and in Turn fine motor skills. \r\nThey also want
to learn through games, my kids don't want to sit and do worksheets. They
want to learn to count by jumping and playing. Physical engagement is the
key to our success. The number toss and color and shape mats can make tha
t happen. My students will forget they are doing work and just have the f
un a 6 year old deserves.nannan
==

```
The mediocre teacher tells. The good teacher explains. The superior teach
er demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy s
chool has 803 students which is makeup is 97.6% African-American, making
up the largest segment of the student body. A typical school in Dallas is
made up of 23.2% African-American students. Most of the students are on f
ree or reduced lunch. We aren't receiving doctors, lawyers, or engineers
children from rich backgrounds or neighborhoods. As an educator I am insp
iring minds of young children and we focus not only on academics but one
smart, effective, efficient, and disciplined students with good characte
r.In our classroom we can utilize the Bluetooth for swift transitions dur
ing class. I use a speaker which doesn't amplify the sound enough to rece
ive the message. Due to the volume of my speaker my students can't hear v
ideos or books clearly and it isn't making the lessons as meaningful. But
with the bluetooth speaker my students will be able to hear and I can sto
p, pause and replay it at any time.\r\nThe cart will allow me to have mor
e room for storage of things that are needed for the day and has an extra
part to it I can use.  The table top chart has all of the letter, words a
nd pictures for students to learn about different letters and it is more
accessible.nannan
==================================================
```

In [11]:
```
# 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 [12]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. Th
ey are eager beavers and always strive to work their hardest working past
their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students r
eceive free or reduced price lunch.  Despite their disabilities and limit
ations, my students love coming to school and come eager to learn and exp
lore.Have you ever felt like you had ants in your pants and you needed to
groove and move as you were in a meeting? This is how my kids feel all th
e time. The want to be able to move as they learn or so they say.Wobble c
hairs are the answer and I love then because they develop their core, whi
ch enhances gross motor and in Turn fine motor skills. \r\nThey also want
to learn through games, my kids do not want to sit and do worksheets. The
y want to learn to count by jumping and playing. Physical engagement is t
he key to our success. The number toss and color and shape mats can make
that happen. My students will forget they are doing work and just have th
e fun a 6 year old deserves.nannan
==================================================
```

In [13]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-li
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. Th
ey are eager beavers and always strive to work their hardest working past
their limitations.     The materials we have are the ones I seek out for
my students. I teach in a Title I school where most of the students recei
ve free or reduced price lunch.  Despite their disabilities and limitatio
ns, my students love coming to school and come eager to learn and explor
e.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the
time. The want to be able to move as they learn or so they say.Wobble cha
irs are the answer and I love then because they develop their core, which
enhances gross motor and in Turn fine motor skills.   They also want to l
earn through games, my kids do not want to sit and do worksheets. They wa
nt to learn to count by jumping and playing. Physical engagement is the k
ey to our success. The number toss and color and shape mats can make that
happen. My students will forget they are doing work and just have the fun
a 6 year old deserves.nannan
```

In [14]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and
language delays cognitive delays gross fine motor delays to autism They a
re eager beavers and always strive to work their hardest working past the
ir limitations The materials we have are the ones I seek out for my stude
nts I teach in a Title I school where most of the students receive free o
r reduced price lunch Despite their disabilities and limitations my stude
nts love coming to school and come eager to learn and explore Have you ev
er felt like you had ants in your pants and you needed to groove and move
as you were in a meeting This is how my kids feel all the time The want t
o be able to move as they learn or so they say Wobble chairs are the answ
er and I love then because they develop their core which enhances gross m
otor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count
by jumping and playing Physical engagement is the key to our success The
number toss and color and shape mats can make that happen My students wil
l forget they are doing work and just have the fun a 6 year old deserves
nannan
```

In [15]:
```
# 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', 'y
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', '
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have'
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'be
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on'
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', '
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "sho
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "sh
            'won', "won't", 'wouldn', "wouldn't"]
```

In [16]:
```
# 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 [00:57<00:00, 1886.21it/s]
```

In [17]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[17]:
```
'my kindergarten students varied disabilities ranging speech language del
ays cognitive delays gross fine motor delays autism they eager beavers al
ways strive work hardest working past limitations the materials ones i se
ek students i teach title i school students receive free reduced price lu
nch despite disabilities limitations students love coming school come eag
er learn explore have ever felt like ants pants needed groove move meetin
g this kids feel time the want able move learn say wobble chairs answer i
love develop core enhances gross motor turn fine motor skills they also w
ant learn games kids not want sit worksheets they want learn count jumpin
g playing physical engagement key success the number toss color shape mat
s make happen my students forget work fun 6 year old deserves nannan'
```

1.4 Preprocessing of `project_title`

In [18]:
```
# similarly you can preprocess the titles also
# Combining all the above stundents
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:02<00:00, 47360.16it/s]
```

1.5 Preparing data for models

In [19]:
```
project_data.columns
```

Out[19]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_stat
e',
       'project_submitted_datetime', 'project_grade_category', 'project_t
itle',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approv
ed',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

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

1.5.1 Vectorizing Categorical data

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

In [20]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lower
categories_one_hot = vectorizer.fit_transform(project_data['clean_categorie
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [21]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), l
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subca
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutritio
nEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness',
'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics',
'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [22]:
```
# you can do the similar thing with state, teacher_prefix and project_grade

vectorizer = CountVectorizer(lowercase=False, binary=True)
print (project_data['school_state'].head(5))
vectorizer.fit(project_data['school_state'].values)
print (vectorizer.get_feature_names())

states_one_hot = vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encodig ",states_one_hot.shape)
```

```
0    IN
1    FL
2    AZ
3    KY
4    TX
Name: school_state, dtype: object
['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']
Shape of matrix after one hot encodig  (109248, 51)
```

In [23]:
```
print ('Nan Values:',project_data['teacher_prefix'].isnull().sum())
# Replacing the NaN values with most frequently used value of teacher prefi

project_data.loc[project_data['teacher_prefix'].isnull(),'teacher_prefix']=
print ('After Imputing:',project_data['teacher_prefix'].isnull().sum())


vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'])
print (vectorizer.get_feature_names())

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

```
Nan Values: 3
After Imputing: 0
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (109248, 5)
```

In [24]:
```
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.split(','))

prjctgrd_dict = dict(my_counter)
sorted_prjctgrd_dict = dict(sorted(prjctgrd_dict.items(), key=lambda kv: kv

vectorizer = CountVectorizer(vocabulary=list(sorted_prjctgrd_dict.keys()),l
vectorizer.fit(project_data['project_grade_category'].values)
print (vectorizer.get_feature_names())

project_grade_category_one_hot = vectorizer.transform(project_data['project
print("Shape of matrix after one hot encodig ",project_grade_category_one_h
```

```
['Grades 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2']
Shape of matrix after one hot encodig  (109248, 4)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [25]:
```
# We are considering only the words which appeared in at least 10 documents
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

In [26]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it

vectorizer = CountVectorizer(min_df=10)
titles_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",titles_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3222)
```

1.5.2.2 TFIDF vectorizer

In [27]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [0]:
```
''
 Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084(
ef loadGloveModel(gloveFile):
   print ("Loading Glove Model")
   f = open(gloveFile,'r', encoding="utf8")
   model = {}
   for line in tqdm(f):
       splitLine = line.split()
       word = splitLine[0]
       embedding = np.array([float(val) for val in splitLine[1:]])
       model[word] = embedding
   print ("Done.",len(model)," words loaded!")
   return model
odel = loadGloveModel('glove.42B.300d.txt')

 ============================
utput:

oading Glove Model
917495it [06:32, 4879.69it/s]
one. 1917495  words loaded!

 ============================

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

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

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

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


 stronging variables into pickle files python: http://www.jessicayung.com/ho

mport pickle
ith open('glove_vectors', 'wb') as f:
   pickle.dump(words_courpus, f)


''
```

Out[26]: '\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034

```
9/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGl
oveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(glo
veFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm
(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        mode
l[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n
return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# =======
====================\nOutput:\n    \nLoading Glove Model\n1917495it [06:
32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ====================
=======\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.spli
t(\' \'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))
\nprint("all the words in the coupus", len(words))\nwords = set(words)\np
rint("the unique words in the coupus", len(words))\n\ninter_words = set(m
odel.keys()).intersection(words)\nprint("The number of words that are pre
sent in both glove vectors and our coupus",       len(inter_words),"(",n
p.round(len(inter_words)/len(words)*100,3),"%")\n\nwords_courpus = {}\nw
ords_glove = set(model.keys())\nfor i in words:\n    if i in words_glov
e:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(w
ords_courpus))\n\n\n# stronging variables into pickle files python: htt
p://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-p
ython/\n\nimport (http://www.jessicayung.com/how-to-use-pickle-to-save-an
d-load-variables-in-python/\n\nimport) pickle\nwith open(\'glove_vectors
\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [0]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [0]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in t
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████
██████| 109248/109248 [01:00<00:00, 1806.88it/s]

109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [0]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf
tfidf_words = set(tfidf_model.get_feature_names())
```

In [0]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/re
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the t
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.sp
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████
██████| 109248/109248 [08:03<00:00, 225.81it/s]

109248
300
```

In [0]:
```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

In [28]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'su
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [29]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generate
from sklearn.preprocessing import StandardScaler

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

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.re
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

In [30]:
```
price_standardized
```

Out[30]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

1.5.4 Merging all the above features

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

In [31]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [32]:
```
# 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
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_sta
X.shape
```

Out[32]:
```
(109248, 16663)
```

In [0]:
```
# please write all the code with proper documentation, and proper titles fo
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

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)**

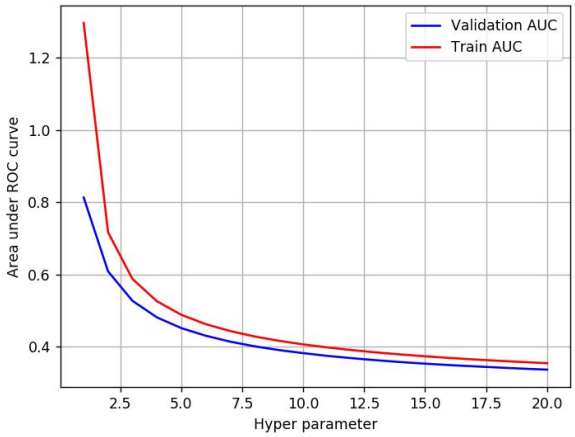
   - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
   - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
   - 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**

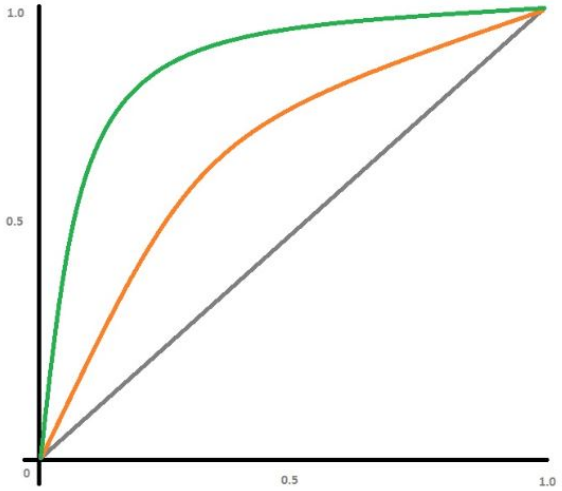
   - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding 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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

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 (http://zetcode.com/python/prettytable/)

Vectorizer	Model	Hyper parameter	AUC
BOW	Brute	7	0.78
TFIDF	Brute	12	0.79
W2V	Brute	10	0.78
TFIDFW2V	Brute	6	0.78

2. Naive Bayes

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

In [80]:
```
please write all the code with proper documentation, and proper titles for
go through documentations and blogs before you start coding
first figure out what to do, and then think about how to do.
reading and understanding error messages will be very much helpfull in debu
when you plot any graph make sure you use
  # a. Title, that describes your plot, this will be very helpful to the re
  # b. Legends if needed
  # c. X-axis label
  # d. Y-axis label
Importing required libraries

om sklearn.model_selection import train_test_split
om sklearn.model_selection import cross_val_score
om sklearn import model_selection
om sklearn.preprocessing import minmax_scale


preparing the data matrix with all the required features


split the data set into train and test with 70% train and 30% test
oject_data_1, project_data_test = model_selection.train_test_split(project_

split the train data set into cross validation train and cross validation t
oject_data_tr, project_data_cv = model_selection.train_test_split(project_d

int (project_data_tr.shape)
int (project_data_cv.shape)
int (project_data_test.shape)
```

```
(53531, 20)
(22942, 20)
(32775, 20)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [81]:
```
# please write all the code with proper documentation, and proper titles for
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debu
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
  # a. Title, that describes your plot, this will be very helpful to the re
  # b. Legends if needed
  # c. X-axis label
  # d. Y-axis label

# ============================Encoding Categorical features only on Train Da

print("="*75)
print("Encoding Categorical features on Train Data (project_data_tr)")
print("="*75)

# ************Encoding categories

from sklearn.feature_extraction.text import CountVectorizer
at_vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lov
ategories_one_hot_train = cat_vectorizer.fit_transform(project_data_tr['clea
# print(cat_vectorizer.get_feature_names())
print("Shape of Train categories (categories_one_hot_train) ",categories_one_

# ************Encoding sub categories

ub_cat_vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys
ub_categories_one_hot_train = sub_cat_vectorizer.fit_transform(project_data_
# print(sub_cat_vectorizer.get_feature_names())
print("Shape of Train subcategories (sub_categories_one_hot_train) ",sub_cate

# ************Encoding school state

chool_state_vectorizer = CountVectorizer(lowercase=False, binary=True)
tates_one_hot_train = school_state_vectorizer.fit_transform(project_data_tr
# print (school_state_vectorizer.get_feature_names())
print("Shape of school state (states_one_hot_train) ",states_one_hot_train.sh


# **************Encoding teacher prefix

# print ('Nan Values:',project_data_tr['teacher_prefix'].isnull().sum())
# Replacing the NaN values with most frequently used value of teacher prefix

# project_data.loc[project_data_tr['teacher_prefix'].isnull(),'teacher_prefi
# print ('After Imputing:',project_data_tr['teacher_prefix'].isnull().sum())


c_prefix_vectorizer = CountVectorizer(lowercase=False, binary=True)
c_prefix_vectorizer.fit(project_data_tr['teacher_prefix'])
# print (tc_prefix_vectorizer.get_feature_names())

eacher_prfx_one_hot_train = tc_prefix_vectorizer.transform(project_data_tr[
print("Shape of teacher prefix (teacher_prfx_one_hot_train) ",teacher_prfx_on
```

```
# **************Encoding project grade category

y_counter = Counter()
or word in project_data_tr['project_grade_category'].values:
    my_counter.update(word.split(','))

rjctgrd_dict = dict(my_counter)
orted_prjctgrd_dict = dict(sorted(prjctgrd_dict.items(), key=lambda kv: kv[

rade_vectorizer = CountVectorizer(vocabulary=list(sorted_prjctgrd_dict.keys
rade_vectorizer.fit(project_data_tr['project_grade_category'].values)
print (grade_vectorizer.get_feature_names())

roject_grade_category_one_hot_train = grade_vectorizer.transform(project_da
rint("Shape of grade (project_grade_category_one_hot_train) ",project_grade_


# ****************Numerical features

rice_normalized_train_data = minmax_scale(project_data_tr['price']).reshape

rint("Shape of price standardized (price_normalized_train_data) ",price_norm




# ============================Encoding Categorical features on cross valida

rint("="*75)
rint("Encoding Categorical features on cross validate Data (project_data_cv
rint("="*75)

# *************Encoding categories

ategories_one_hot_cv = cat_vectorizer.transform(project_data_cv['clean_cate
print(cat_vectorizer.get_feature_names())
rint("Shape of cv categories (categories_one_hot_cv) ",categories_one_hot_cv

# ************Encoding sub categories

ub_categories_one_hot_cv = sub_cat_vectorizer.transform(project_data_cv['cl
print(sub_cat_vectorizer.get_feature_names())
rint("Shape of cv subcategories (sub_categories_one_hot_cv) ",sub_categories

# ************Encoding school state

tates_one_hot_cv = school_state_vectorizer.transform(project_data_cv['schoo
print (school_state_vectorizer.get_feature_names())
rint("Shape of school state (states_one_hot_cv) ",states_one_hot_cv.shape)


# *************Encoding teacher prefix

print ('Nan Values:',project_data_cv['teacher_prefix'].isnull().sum())
Replacing the NaN values with most frequently used value of teacher prefix
```

```
project_data.loc[project_data_cv['teacher_prefix'].isnull(),'teacher_prefi
print ('After Imputing:',project_data_cv['teacher_prefix'].isnull().sum())


eacher_prfx_one_hot_cv = tc_prefix_vectorizer.transform(project_data_cv['tea
rint("Shape of teacher prefix (teacher_prfx_one_hot_cv) ",teacher_prfx_one_

**************Encoding project grade category

y_counter = Counter()
or word in project_data_cv['project_grade_category'].values:
    my_counter.update(word.split(','))

rjctgrd_dict = dict(my_counter)
orted_prjctgrd_dict = dict(sorted(prjctgrd_dict.items(), key=lambda kv: kv[

roject_grade_category_one_hot_cv = grade_vectorizer.transform(project_data_
rint("Shape of grade (project_grade_category_one_hot_cv) ",project_grade_ca


**************Numerical features

rice_normalized_cv_data = minmax_scale(project_data_cv['price']).reshape(-1

rint("Shape of price standardized (price_normalized_cv_data)",price_normali


==============================Encoding Categorical features on Test Data (pr

rint("="*75)
rint("Encoding Categorical features on Test Data (project_data_test)")
rint("="*75)

************Encoding categories

ategories_one_hot_test = cat_vectorizer.transform(project_data_test['clean_c
print(cat_vectorizer.get_feature_names())
rint("Shape of test categories (categories_one_hot_test) ",categories_one_ho

***********Encoding sub categories

ub_categories_one_hot_test = sub_cat_vectorizer.transform(project_data_test
print(sub_cat_vectorizer.get_feature_names())
rint("Shape of test subcategories (sub_categories_one_hot_test) ",sub_categ

***********Encoding school state

tates_one_hot_test = school_state_vectorizer.transform(project_data_test['sc
print (school_state_vectorizer.get_feature_names())
rint("Shape of school state (states_one_hot_test) ",states_one_hot_test.shap


**************Encoding teacher prefix

print ('Nan Values:',project_data_test['teacher_prefix'].isnull().sum())
Replacing the NaN values with most frequently used value of teacher prefix
```

```
 project_data.loc[project_data_test['teacher_prefix'].isnull(),'teacher_pre
 print ('After Imputing:',project_data_test['teacher_prefix'].isnull().sum(


eacher_prfx_one_hot_test = tc_prefix_vectorizer.transform(project_data_test
rint("Shape of teacher prefix (teacher_prfx_one_hot_test) ",teacher_prfx_on


 ***************Encoding project grade category

y_counter = Counter()
or word in project_data_test['project_grade_category'].values:
    my_counter.update(word.split(','))

rjctgrd_dict = dict(my_counter)
orted_prjctgrd_dict = dict(sorted(prjctgrd_dict.items(), key=lambda kv: kv[

roject_grade_category_one_hot_test = grade_vectorizer.transform(project_data
rint("Shape of grade (project_grade_category_one_hot_test)  ",project_grade_


 *****************Numerical features

rice_normalized_test_data = minmax_scale(project_data_test['price']).reshape

rint("Shape of price standardized (price_normalized_test_data) ",price_norma
```

```
==========================================================================
==
Encoding Categorical features on Train Data (project_data_tr)
==========================================================================
==
Shape of Train categories (categories_one_hot_train)  (53531, 9)
Shape of Train subcategories (sub_categories_one_hot_train)  (53531, 30)
Shape of school state (states_one_hot_train)  (53531, 51)
Shape of teacher prefix (teacher_prfx_one_hot_train)  (53531, 5)
Shape of grade (project_grade_category_one_hot_train)  (53531, 4)
Shape of price standardized (price_normalized_train_data)  (53531, 1)
==========================================================================
==
Encoding Categorical features on cross validate Data (project_data_cv)
==========================================================================
==
Shape of cv categories (categories_one_hot_cv)  (22942, 9)
Shape of cv subcategories (sub_categories_one_hot_cv)  (22942, 30)
Shape of school state (states_one_hot_cv)  (22942, 51)
Shape of teacher prefix (teacher_prfx_one_hot_cv)  (22942, 5)
Shape of grade (project_grade_category_one_hot_cv)  (22942, 4)
Shape of price standardized (price_normalized_cv_data) (22942, 1)
==========================================================================
==
Encoding Categorical features on Test Data (project_data_test)
==========================================================================
==
Shape of test categories (categories_one_hot_test)  (32775, 9)
Shape of test subcategories (sub_categories_one_hot_test)  (32775, 30)
Shape of school state (states_one_hot_test)  (32775, 51)
Shape of teacher prefix (teacher_prfx_one_hot_test)  (32775, 5)
```

```
Shape of grade (project_grade_category_one_hot_test)   (32775, 4)
Shape of price standardized (price_normalized_test_data)  (32775, 1)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [35]:
```
# please write all the code with proper documentation, and proper titles fo
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in de
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
# ============================Encoding eassay, and project_title only on T

print("="*75)
print("Encoding eassay, and project_title only on Train Data (project_data_
print("="*75)

# ********** Text prepocessing on essays and titles from Train Datapreproce
preprocessed_train_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_tr['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_train_essays.append(sent.lower().strip())


preprocessed_train_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_tr['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_train_titles.append(sent.lower().strip())


# *************** BOW For Essays

# We are considering only the words which appeared in at least 10 documents
bow_essay_vectorizer = CountVectorizer(min_df=10)
train_text_bow = bow_essay_vectorizer.fit_transform(preprocessed_train_essa
print("BOW == Shape of Train Data Text encoding (train_text_bow) ",train_te

# *************** BOW For Titles
bow_title_vectorizer = CountVectorizer(min_df=10)
train_titles_bow = bow_title_vectorizer.fit_transform(preprocessed_train_ti
print("BOW == Shape of Train Data Title encoding (train_titles_bow) ",train
```

```
# *************** TFIDF For Essays
tfidf_essay_vectorizer = TfidfVectorizer(min_df=10)
train_text_tfidf = tfidf_essay_vectorizer.fit_transform(preprocessed_train_
print("TFID == Shape of Train Data Text encoding (train_text_tfidf) ",train

# *************** TFIDF For Titles
tfidf_title_vectorizer = TfidfVectorizer(min_df=10)
train_title_tfidf = tfidf_title_vectorizer.fit_transform(preprocessed_train
print("TFIDF == Shape of Train Data Title encoding (train_title_tfidf)  ",t




# ============================Encoding eassay, and project_title only on c

print("="*75)
print("Encoding eassay, and project_title only on cv Data (project_data_cv)
print("="*75)

# ********** Text prepocessing on essays and titles from cv Datapreprocesse

preprocessed_cv_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_cv['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_cv_essays.append(sent.lower().strip())


preprocessed_cv_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_cv['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_cv_titles.append(sent.lower().strip())


# *************** BOW For Essays

# We are considering only the words which appeared in at least 10 documents
cv_text_bow = bow_essay_vectorizer.transform(preprocessed_cv_essays)
print("BOW == Shape of cv Data Text encoding (cv_text_bow) ",cv_text_bow.sh

# *************** BOW For Titles
cv_titles_bow = bow_title_vectorizer.transform(preprocessed_cv_titles)
print("BOW == Shape of cv Data Title encoding (cv_titles_bow) ",cv_titles_b
```

```
# *************** TFIDF For Essays
cv_text_tfidf = tfidf_essay_vectorizer.transform(preprocessed_cv_essays)
print("TFID == Shape of cv Data Text encoding (cv_text_tfidf) ",cv_text_tfi

# *************** TFIDF For Titles
cv_title_tfidf = tfidf_title_vectorizer.transform(preprocessed_cv_titles)
print("TFIDF == Shape of cv Data Title encoding (cv_title_tfidf)  ",cv_titl



# ============================Encoding eassay, and project_title only on t

print("="*75)
print("Encoding eassay, and project_title only on test Data (project_data_t
print("="*75)

# ********** Text prepocessing on essays and titles from test Datapreproces
preprocessed_test_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_test['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_test_essays.append(sent.lower().strip())


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


# *************** BOW For Essays

# We are considering only the words which appeared in at least 10 documents
test_text_bow = bow_essay_vectorizer.transform(preprocessed_test_essays)
print("BOW == Shape of test Data Text encoding (test_text_bow) ",test_text_

# *************** BOW For Titles
test_titles_bow = bow_title_vectorizer.transform(preprocessed_test_titles)
print("BOW == Shape of test Data Title encoding (test_titles_bow) ",test_ti

# *************** TFIDF For Essays
test_text_tfidf = tfidf_essay_vectorizer.transform(preprocessed_test_essays
```

```
print("TFID == Shape of test Data Text encoding (test_text_tfidf) ",test_te

# *************** TFIDF For Titles
test_title_tfidf = tfidf_title_vectorizer.transform(preprocessed_test_title
print("TFIDF == Shape of test Data Title encoding (test_title_tfidf)  ",tes
```

```
  1%|          | 312/53531 [00:00<00:35, 1493.50it/s]

==========================================================================
==
Encoding eassay, and project_title only on Train Data (project_data_tr)
==========================================================================
==

100%|██████████| 53531/53531 [00:26<00:00, 1988.81it/s]
100%|██████████| 53531/53531 [00:01<00:00, 42190.13it/s]

BOW == Shape of Train Data Text encoding (train_text_bow)  (53531, 12441)
BOW == Shape of Train Data Title encoding (train_titles_bow)  (53531, 207
8)
TFID == Shape of Train Data Text encoding (train_text_tfidf)  (53531, 124
41)

  2%|          | 386/22942 [00:00<00:11, 1925.95it/s]

TFIDF == Shape of Train Data Title encoding (train_title_tfidf)   (53531,
2078)
==========================================================================
==
Encoding eassay, and project_title only on cv Data (project_data_cv)
==========================================================================
==

100%|██████████| 22942/22942 [00:11<00:00, 1920.27it/s]
100%|██████████| 22942/22942 [00:00<00:00, 38719.76it/s]

BOW == Shape of cv Data Text encoding (cv_text_bow)  (22942, 12441)
BOW == Shape of cv Data Title encoding (cv_titles_bow)  (22942, 2078)

  1%|          | 211/32775 [00:00<00:15, 2103.96it/s]

TFID == Shape of cv Data Text encoding (cv_text_tfidf)  (22942, 12441)
TFIDF == Shape of cv Data Title encoding (cv_title_tfidf)   (22942, 2078)
==========================================================================
==
Encoding eassay, and project_title only on test Data (project_data_test)
==========================================================================
==

100%|██████████| 32775/32775 [00:16<00:00, 2008.85it/s]
100%|██████████| 32775/32775 [00:00<00:00, 43848.09it/s]

BOW == Shape of test Data Text encoding (test_text_bow)  (32775, 12441)
BOW == Shape of test Data Title encoding (test_titles_bow)  (32775, 2078)
TFID == Shape of test Data Text encoding (test_text_tfidf)  (32775, 1244
1)
```

```
TFIDF == Shape of test Data Title encoding (test_title_tfidf)   (32775, 2
078)
```

In [36]:
```
# Data size of encoded essays and titles on Train, CV and Test datas

print ("="*75)
print("Train Data")
print ("="*75)

print ("BOW on Essay (train_text_bow) ",train_text_bow.shape)
print ("BOW on title (train_titles_bow) ",train_titles_bow.shape)

print ("TFIDF on Essay (train_text_tfidf)", train_text_tfidf.shape)
print ("TFIDF on Title (train_title_tfidf)", train_title_tfidf.shape)


print ("="*75)
print("CV Data")
print ("="*75)

print ("BOW on Essay (cv_text_bow) ",cv_text_bow.shape)
print ("BOW on Title (cv_titles_bow) ",cv_titles_bow.shape)

print ("TFIDF on Essay (cv_text_tfidf)", cv_text_tfidf.shape)
print ("TFIDF on Title (cv_title_tfidf)", cv_title_tfidf.shape)


print ("="*75)
print("TEST Data")
print ("="*75)

print ("BOW on Essay (test_text_bow) ",test_text_bow.shape)
print ("BOW on title (test_titles_bow) ",test_titles_bow.shape)

print ("TFIDF on Essay (test_text_tfidf)", test_text_tfidf.shape)
print ("TFIDF on Title (test_title_tfidf)", test_title_tfidf.shape)
```

```
===========================================================================
==
Train Data
===========================================================================
==
BOW on Essay (train_text_bow)  (53531, 12441)
BOW on title (train_titles_bow)  (53531, 2078)
TFIDF on Essay (train_text_tfidf) (53531, 12441)
TFIDF on Title (train_title_tfidf) (53531, 2078)
===========================================================================
==
CV Data
===========================================================================
==
BOW on Essay (cv_text_bow)  (22942, 12441)
BOW on Title (cv_titles_bow)  (22942, 2078)
TFIDF on Essay (cv_text_tfidf) (22942, 12441)
TFIDF on Title (cv_title_tfidf) (22942, 2078)
===========================================================================
==
TEST Data
===========================================================================
```

```
==
BOW on Essay (test_text_bow)  (32775, 12441)
BOW on title (test_titles_bow)  (32775, 2078)
TFIDF on Essay (test_text_tfidf) (32775, 12441)
TFIDF on Title (test_title_tfidf) (32775, 2078)
```

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

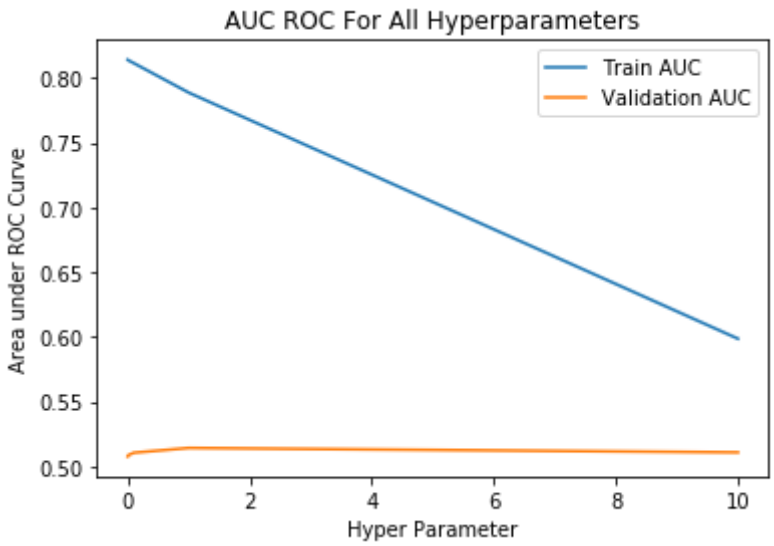
In [100]:
```
# Preparing the data matrix with the given features
from scipy.sparse import hstack
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn import metrics

# method to plot the graph for Hyperparameter Vs AUC
def plot_hyper_vs_auc(train_auc_score, cv_auc_score, hyper_parameters):

    plt.plot(hyper_parameters ,train_auc_score, label= "Train AUC")
    plt.plot(hyper_parameters,cv_auc_score, label="Validation AUC")
    plt.title("AUC ROC For All Hyperparameters")
    plt.xlabel("Hyper Parameter")
    plt.ylabel("Area under ROC Curve")
    plt.legend()
    plt.show()

# method to get auc score based on prediction
def get_auc_score (train_data, train_val, cv_data, cv_val):

    auc_score = []

    for i in ([10**x for x in range (-5,2) ]):

        mnb = MultinomialNB(alpha=i,class_prior=(0.5,0.5))


        # fitting the model on crossvalidation train
        mnb.fit(train_data, train_val)

        # predict the response on the given data (train/cv)
        pred = mnb.predict_proba(cv_data)
        postv_class_test_prob = [item[1] for item in pred]

        # Appending the score to a list
        auc_score.append(roc_auc_score(cv_val, postv_class_test_prob))

    return auc_score

def plot_roc (fpr_test,tpr_test,fpr_train,tpr_train):

    plt.plot(fpr_test,tpr_test, label="ROC Curve for Test Data")
    plt.plot(fpr_train,tpr_train, label="ROC Curve for Train Data")
    plt.title("Area under ROC Curve")
    plt.xlabel("FPR")
    plt.ylabel("TPR")
    plt.legend()
    plt.show()
```

2.4.1 Applying Naive Bayes on BOW, SET 1

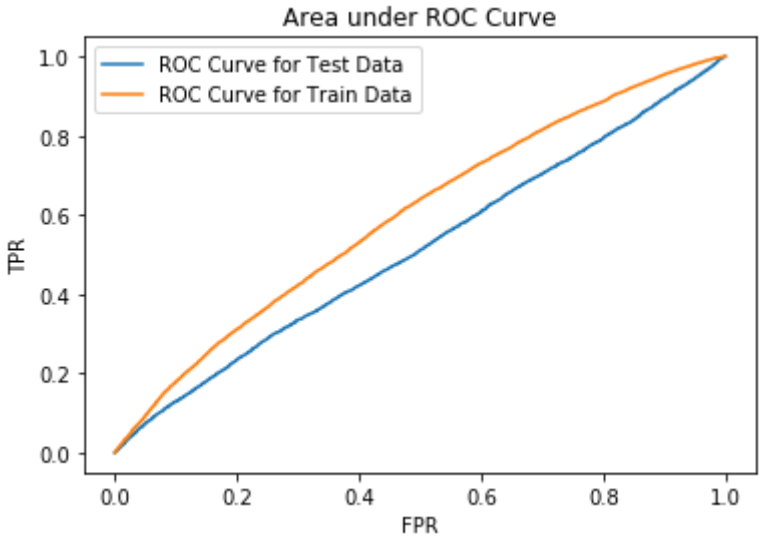
In [101]:

```
# Applying Naive Bayes on Set 1: categorical, numerical features + project_

set1_train_data = hstack((categories_one_hot_train,sub_categories_one_hot_t
print ("Shape of BOW Train Dataset ",set1_train_data.shape)

set1_cv_data = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,stat
print ("Shape of BOW CV Dataset ",set1_cv_data.shape)

set1_test_data = hstack((categories_one_hot_test,sub_categories_one_hot_tes
print ("Shape of BOW Test Dataset ",set1_test_data.shape)

# get CV AUC Score
cv_auc_score = get_auc_score (set1_train_data,project_data_tr["project_is_a

# Get Train AUC Score
train_auc_score = get_auc_score (set1_train_data,project_data_tr["project_i

hyper_parameters = list([10**x for x in range (-5,2) ])

# Plot Graph for Hyperparameter VS AUC
plot_hyper_vs_auc (train_auc_score,cv_auc_score,hyper_parameters)
```

```
Shape of BOW Train Dataset  (53531, 14619)
Shape of BOW CV Dataset  (22942, 14619)
Shape of BOW Test Dataset  (32775, 14619)
```



In [102]:
```
# training the model with best alpha
mnb = MultinomialNB(alpha=10,fit_prior=True)
mnb.fit(set1_train_data, project_data_tr["project_is_approved"])

# predict probabilities for test data
pred_prob = mnb.predict_proba(set1_test_data)
postv_class_test_prob = [item[1] for item in pred_prob]

fpr_test, tpr_test, thresholds = metrics.roc_curve(project_data_test["proje

# predict probabilities for train data
pred_prob = mnb.predict_proba(set1_train_data)
postv_class_train_prob = [item[1] for item in pred_prob]

fpr_train, tpr_train, thresholds = metrics.roc_curve(project_data_tr["proje

# Plot ROC Curve for Train and Test data
plot_roc (fpr_test,tpr_test,fpr_train,tpr_train)
```



In [113]:
```
# Confusion matrix for testData
from sklearn.metrics import confusion_matrix

y_pred = mnb.predict(set1_test_data)

conf_mtrx = confusion_matrix(project_data_test["project_is_approved"], y_pr
print('Test Data Confusion matrix')
print (conf_mtrx)

print('Train Data Confusion matrix')
y_pred = mnb.predict(set1_train_data)
conf_mtrx = confusion_matrix(project_data_tr["project_is_approved"], y_pred
print (conf_mtrx)
```

```
Test Data Confusion matrix
[[  231  4768]
 [ 1408 26368]]
Train Data Confusion matrix
[[  762  7314]
 [ 1901 43554]]
```

2.4.1.1 Top 10 important features of positive class from SET 1

In [0]:
```
# Please write all the code with proper documentation
```

2.4.1.2 Top 10 important features of negative class from SET 1

In [0]:
```
# Please write all the code with proper documentation
```

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [0]:
```
# Please write all the code with proper documentation
```

2.4.2.1 Top 10 important features of positive class from SET 2

In [0]:
```
# Please write all the code with proper documentation
```

2.4.2.2 Top 10 important features of negative class from SET 2

In [0]:
```
# Please write all the code with proper documentation
```

3. Conclusions

In [0]:
```
# Please compare all your models using Prettytable library
```