DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __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 [0]:
```
# Install the PyDrive wrapper & import libraries.
# This only needs to be done once per notebook.
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
# This only needs to be done once per notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

In [0]:
```
%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 [0]:
```
# Download a file based on its file ID.
#https://drive.google.com/file/d/1T48h84GLW3dpy9F6ble5nF_1gQxBO8rx/view?usp=sharing
file_id = '1T48h84GLW3dpy9F6ble5nF_1gQxBO8rx'
downloaded = drive.CreateFile({'id': file_id})
#print('Downloaded content "{}"'.format(downloaded.GetContentString()))
```

In [0]:
```
downloaded.GetContentFile('train_data.csv')
```

In [0]:
```
project_data = pd.read_csv('train_data.csv')
```

In [0]:
```
project_data.shape
```

Out[0]:
```
(109248, 17)
```

In [0]:
```
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 [0]:
```
# Download a file based on its file ID.
#https://drive.google.com/file/d/14OVXWu_SJU-lJD-jKMOCld14EZ21lYYe/view?usp=sharing
# A file ID looks like: laggVyWshwcyP6kEI-y_W3P8D26sz
#https://drive.google.com/file/d/14OVXWu_SJU-lJD-jKMOCld14EZ21lYYe/view?usp=sharing
file_id = '14OVXWu_SJU-lJD-jKMOCld14EZ21lYYe'
downloaded = drive.CreateFile({'id': file_id})
#print('Downloaded content "{}"'.format(downloaded.GetContentString()))
```

In [0]:
```
downloaded.GetContentFile('resources.csv')
```

In [0]:
```
resource_data = pd.read_csv('resources.csv')
```

In [0]:
```
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[0]:

	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 [0]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [0]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

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

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

1.3 Text preprocessing

In [0]:
```
# 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 [0]:
```
project_data.head(2)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educ Supp E Lear
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	W Projec H Le

In [0]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [0]:
```
# 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)
```

```
My students are English learners that are working on English as their second 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 with students at every level of mastery.  We also have over 4
0 countries represented with the families within our school.  Each student brings a wealth of knowledge and experienc
es to us that open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits of you
r world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home that begs for more resourc
es.  Many times our parents are learning to read and speak English along side of their children.  Sometimes this crea
tes barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skill
s.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language e
ven if no one at home is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the English Learner Teach
er and will be sent home regularly to watch.  The videos are to help the child develop early reading skills.\r\n\r\nP
arents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the yea
r.  The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the ti
me. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority
students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is
a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festiva
l with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrat
e 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 colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a tota
l of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety o
f ways. During independent reading time they will be used as special chairs students will each use on occasion. I wil
l utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they wi
ll be used by the students who need the highest amount of movement in their life in order to stay focused on schoo
l.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki Stools. They can't get thei
r fill of the 5 stools we already have. When the students are 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 where 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. Th
e Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools wi
ll help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balan
ce while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a chil
d 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 teach
er in front of the room? A typical 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 day.\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 there is a high enough percentage of f
ree and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there ar
e no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponge
s, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red
throw pillows and the whimsical nautical hanging decor 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 importa
nt 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 Teacher evening. I'll take pictures of each child with th
em, have them developed, and then hung in our classroom ready for their first day of 4th grade.  This kind gesture wi
ll set the tone before even the first day of school! 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, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own po
cket on resources to get our classroom ready. Please consider helping with this project to make our new school year a
very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fi
ne motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitat
ions. \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 o
f the students receive free or reduced price lunch.  Despite their disabilities and limitations, my students love com
ing to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed t
o groove 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 the
y learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances g
ross 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 nu
mber 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 [0]:
```
# 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 [0]:
```
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/fi
ne motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitat
ions. \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 o
f the students receive free or reduced price lunch.  Despite their disabilities and limitations, my students love com
ing to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed t
o groove 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 the
y learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances g
ross motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and d
o 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 will forget they are doing work and just have
the fun a 6 year old deserves.nannan
==================================================
```

In [0]:
```
# \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)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fi
ne motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitat
ions.     The materials we have are the ones I seek out for my students. I teach in a Title I school where most of th
e students receive free or reduced price lunch.  Despite their disabilities and limitations, my students love coming
to school and come eager to learn and explore.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 le
arn or so they say.Wobble chairs 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 learn through games, my kids do not want to sit and do works
heets. 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 will forget they are doing work and just have the fun
a 6 year old deserves.nannan
```

In [0]:
```
#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 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
receive free or reduced price lunch Despite their disabilities and limitations my students love coming to school and
come eager to learn and explore 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 the time The want to be able to move as they learn or so they s
ay Wobble chairs are the answer and I love then because they develop their core which enhances gross motor and in Tur
n fine motor skills They also want to learn through games my kids do not want to sit and do worksheets They want to l
earn to count by jumping and playing Physical engagement is the key to our success The number toss and color and shap
e mats can make that happen My students will forget they are doing work and just have the fun a 6 year old deserves n
annan
```

In [0]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'furthe
r',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'mor
e',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',
 \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "were
n't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:
```
# 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 [01:07<00:00, 1630.13it/s]
```

In [0]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[0]:
```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross fine motor delays
autism they eager beavers always strive work hardest working past limitations the materials ones i seek students i te
ach title i school students receive free reduced price lunch despite disabilities limitations students love coming sc
hool come eager learn explore have ever felt like ants pants needed groove move meeting 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 wa
nt learn games kids not want sit worksheets they want learn count jumping playing physical engagement key success the
number toss color shape mats make happen my students forget work fun 6 year old deserves nannan'
```

1.4 Preprocessing of `project_title`

In [0]:
```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:03<00:00, 33387.40it/s]
```

1.5 Preparing data for models

In [0]:
```
project_data.columns
```

Out[0]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      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
```

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

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

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

   (https://seaborn.pydata.org/generated/seaborn.heatmap.html)
   - 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 (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

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

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

In [0]:
```python
from sklearn.model_selection import train_test_split
#splitting categorical data
# clean_categories
X_train_cc, X_test_cc = train_test_split(project_data['clean_categories'].values,test_size = 0.33,random_state = 0 ,sh
uffle = False)
X_train_cccv, X_test_cccv = train_test_split(X_train_cc,test_size = 0.33,random_state = 0 , shuffle = False)
```

In [0]:
```python
X_train_cc.shape
```

Out[0]:
```
(73196,)
```

In [0]:
```python
#clean_subcategories
X_train_csc, X_test_csc = train_test_split(project_data['clean_subcategories'].values, test_size = 0.33,shuffle = Fals
e)
X_train_csccv, X_test_csccv = train_test_split(X_train_csc,test_size = 0.33 ,shuffle = False)
```

In [0]:
```python
#school_state
X_train_ss, X_test_ss = train_test_split(project_data['school_state'].values,test_size = 0.33,shuffle = False,random_s
tate = 0)
X_train_sscv, X_test_sscv = train_test_split(X_train_ss,test_size = 0.33,shuffle = False,random_state = 0)
```

In [0]:
```python
#teacher_prefix
X_train_tp, X_test_tp = train_test_split(project_data['teacher_prefix'].values,test_size = 0.33,shuffle = False,random
_state = 0)
X_train_tpcv, X_test_tpcv = train_test_split(X_train_tp,test_size = 0.33 , shuffle  = False,random_state = 0)
```

In [0]:
```python
#project_grade_category
X_train_pc, X_test_pc = train_test_split(project_data['project_grade_category'].values,test_size = 0.33,random_state =
0,shuffle = False)
X_train_pccv, X_test_pccv = train_test_split(X_train_pc,test_size = 0.33,shuffle = False,random_state = 0)
```

In [0]:
```python
#splitting text data

#splitting project_essays
X_train_pe, X_test_pe = train_test_split(preprocessed_essays,test_size = 0.33,random_state = 0,shuffle = False)
X_train_pecv, X_test_pecv = train_test_split(X_train_pe,test_size = 0.33,random_state = 0,shuffle = False)
```

In [0]:
```python
#splitting project_titles
X_train_pt, X_test_pt = train_test_split(preprocessed_titles,test_size = 0.33,random_state = 0,shuffle = False)
X_train_ptcv, X_test_ptcv = train_test_split(X_train_pt,test_size = 0.33,random_state = 0,shuffle = False)
```

In [0]:
```python
#splitting Project_is_approved data
Y_train, Y_test = train_test_split(project_data['project_is_approved'].values,test_size = 0.33,random_state = 0,shuffl
e = False)
Y_train_cv, Y_test_cv = train_test_split(Y_train,test_size = 0.33,random_state = 0,shuffle = False)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [0]:
```python
# please write all the code with proper documentation, and proper titles for each subsection
# 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 debugging your code
# 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 reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

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 [0]:
```
#categories
# 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()), lowercase=False, binary=True)
categories_one_hot = vectorizer.fit_transform(X_train_cc)
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)

categories_one_hot_cv = vectorizer.transform(X_train_cccv)
print("Shape of matrix after one hot encodig ",categories_one_hot_cv.shape)

categories_one_hot_te = vectorizer.transform(X_test_cc)
print("Shape of matrix after one hot encodig ",categories_one_hot_te.shape)


categories_one_hot_tecv = vectorizer.transform(X_test_cccv)
print("Shape of matrix after one hot encodig ",categories_one_hot_tecv.shape)
```

```
Shape of matrix after one hot encodig  (73196, 9)
Shape of matrix after one hot encodig  (49041, 9)
Shape of matrix after one hot encodig  (36052, 9)
Shape of matrix after one hot encodig  (24155, 9)
```

In [0]:
```
#subcategories
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(X_train_csc)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)

sub_categories_one_hot_cv = vectorizer.transform(X_train_csccv)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_cv.shape)



sub_categories_one_hot_te = vectorizer.transform(X_test_csc)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_te.shape)

sub_categories_one_hot_tecv = vectorizer.transform(X_test_csccv)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_tecv.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government',
'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEduc
ation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelo
pment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNee
ds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (73196, 30)
Shape of matrix after one hot encodig  (49041, 30)
Shape of matrix after one hot encodig  (36052, 30)
Shape of matrix after one hot encodig  (24155, 30)
```

In [0]:
```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
from collections import Counter
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())
school_state_dict = dict(my_counter)
sorted_school_state_dict = dict(sorted(school_state_dict.items(), key=lambda kv: kv[1]))
```

In [0]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train_ss)
school_state_one_hot = vectorizer.fit_transform(X_train_ss)
print("Shape of matrix after one hot encodig ",school_state_one_hot.shape)


school_state_one_hot_cv = vectorizer.transform(X_train_sscv)
print("Shape of matrix after one hot encodig ",school_state_one_hot_cv.shape)


school_state_one_hot_te = vectorizer.transform(X_test_ss)
print("Shape of matrix after one hot encodig ",school_state_one_hot_te.shape)

school_state_one_hot_tecv = vectorizer.transform(X_test_sscv)
print("Shape of matrix after one hot encodig ",school_state_one_hot_tecv.shape)
```

```
Shape of matrix after one hot encodig  (73196, 51)
Shape of matrix after one hot encodig  (49041, 51)
Shape of matrix after one hot encodig  (36052, 51)
Shape of matrix after one hot encodig  (24155, 51)
```

In [0]:
```
print(project_data["teacher_prefix"])
```

```
0           Mrs.
1            Mr.
2            Ms.
3           Mrs.
4           Mrs.
5           Mrs.
6           Mrs.
7            Ms.
8           Mrs.
9            Ms.
10          Mrs.
11           Ms.
12          Mrs.
13          Mrs.
14           Ms.
15           Ms.
16          Mrs.
17           Ms.
18          Mrs.
19           Ms.
20          Mrs.
21          Mrs.
22           Ms.
23           Mr.
24          Mrs.
25          Mrs.
26           Ms.
27       Teacher
28          Mrs.
29          Mrs.
            ...
109218      Mrs.
109219   Teacher
109220      Mrs.
109221   Teacher
109222       Ms.
109223       Ms.
109224       Ms.
109225      Mrs.
109226       Ms.
109227      Mrs.
109228      Mrs.
109229      Mrs.
109230       Ms.
109231      Mrs.
109232      Mrs.
109233       Ms.
109234       Ms.
109235      Mrs.
109236      Mrs.
109237      Mrs.
109238      Mrs.
109239      Mrs.
109240      Mrs.
109241      Mrs.
109242      Mrs.
109243       Mr.
109244       Ms.
109245      Mrs.
109246      Mrs.
109247       Ms.
Name: teacher_prefix, Length: 109248, dtype: object
```

In [0]:
```
from collections import Counter
my_counter = Counter()
for word in project_data['teacher_prefix'].values:
    if not isinstance(word, float):
      word = word.replace('.',' ')
      my_counter.update(word.split())

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

In [0]:
```
##Vectorizing teacher_prefix
# we use count vectorizer to convert the values into one hot encoded features
#https://blog.csdn.net/ningzhimeng/article/details/80953916
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train_tp.astype('U'))
teacher_prefix_one_hot = vectorizer.fit_transform(X_train_tp.astype("U"))
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)

teacher_prefix_one_hot_cv = vectorizer.transform(X_train_tpcv.astype("U"))
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot_cv.shape)


teacher_prefix_one_hot_te = vectorizer.transform(X_test_tp.astype("U"))
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot_te.shape)

teacher_prefix_one_hot_tecv = vectorizer.transform(X_test_tpcv.astype("U"))
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot_tecv.shape)
```

```
Shape of matrix after one hot encodig  (73196, 5)
Shape of matrix after one hot encodig  (49041, 5)
Shape of matrix after one hot encodig  (36052, 5)
Shape of matrix after one hot encodig  (24155, 5)
```

In [0]:
```
from collections import Counter
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.split())
project_grade_category_dict = dict(my_counter)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), key=lambda kv: kv[1]))
```

In [0]:
```
##Vectorizing project_grade_category
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train_pc)
project_grade_category_one_hot = vectorizer.transform(X_train_pc)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot.shape)

project_grade_category_one_hot_cv = vectorizer.transform(X_train_pccv)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_cv.shape)

project_grade_category_one_hot_te = vectorizer.transform(X_test_pc)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_te.shape)

project_grade_category_one_hot_tecv = vectorizer.transform(X_test_pccv)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_tecv.shape)
```

```
Shape of matrix after one hot encodig  (73196, 5)
Shape of matrix after one hot encodig  (49041, 5)
Shape of matrix after one hot encodig  (36052, 5)
Shape of matrix after one hot encodig  (24155, 5)
```

In [0]:

Vectorizing Numerical features

In [0]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [0]:
```
#splitting numerical features
X_train_p, X_test_p = train_test_split(project_data['price'].values,test_size = 0.33,shuffle = False , random_state =
0)
X_train_pcv, X_test_pcv = train_test_split(X_train_p,test_size = 0.33,shuffle = False , random_state = 0)
```

In [0]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import Normalizer
# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)
#normalized_X = preprocessing.normalize(X)
X_train_p[np.isnan(X_train_p)] = np.median(X_train_p[~np.isnan(X_train_p)])
Normalizer().fit(X_train_p.reshape(-1,1))
price_normalized = Normalizer().transform(X_train_p.reshape(-1,1))


X_train_pcv[np.isnan(X_train_pcv)] = np.median(X_train_pcv[~np.isnan(X_train_pcv)])
price_normalized_cv= Normalizer().transform(X_train_pcv.reshape(-1,1))

X_test_pcv[np.isnan(X_test_pcv)] = np.median(X_test_pcv[~np.isnan(X_test_pcv)])
price_normalized_tecv= Normalizer().transform(X_test_pcv.reshape(-1,1))

X_test_p[np.isnan(X_test_p)] = np.median(X_test_p[~np.isnan(X_test_p)])
price_normalized_te= Normalizer().transform(X_test_p.reshape(-1,1))

print(price_normalized.shape)
print(price_normalized_cv.shape)
print(price_normalized_tecv.shape)
print(price_normalized_te.shape)
```

```
(73196, 1)
(49041, 1)
(24155, 1)
(36052, 1)
```

In [0]:
```
X_train_tnpp, X_test_tnpp = train_test_split(project_data['teacher_number_of_previously_posted_projects'].values,test_
size = 0.33,shuffle = False , random_state = 0)
X_train_tnppcv, X_test_tnppcv = train_test_split(X_train_tnpp,test_size = 0.33,shuffle = False , random_state = 0)
```

In [0]:
```
#teacher_number_of_previously_posted_projects feature
from sklearn.preprocessing import Normalizer
# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)
#normalized_X = preprocessing.normalize(X)
X_train_tnpp[np.isnan(X_train_tnpp)] = np.median(X_train_tnpp[~np.isnan(X_train_tnpp)])
Normalizer().fit(X_train_tnpp.reshape(-1,1))
normalized_tnpp = Normalizer().transform(X_train_tnpp.reshape(-1,1))


X_train_tnppcv[np.isnan(X_train_tnppcv)] = np.median(X_train_tnppcv[~np.isnan(X_train_tnppcv)])
normalized_tnppcv= Normalizer().transform(X_train_tnppcv.reshape(-1,1))

X_test_tnppcv[np.isnan(X_test_tnppcv)] = np.median(X_test_tnppcv[~np.isnan(X_test_tnppcv)])
normalized_tnpptecv= Normalizer().transform(X_test_tnppcv.reshape(-1,1))

X_test_p[np.isnan(X_test_tnpp)] = np.median(X_test_tnpp[~np.isnan(X_test_tnpp)])
normalized_tnppte= Normalizer().transform(X_test_tnpp.reshape(-1,1))

print(normalized_tnpp.shape)
print(normalized_tnppcv.shape)
print(normalized_tnpptecv.shape)
print(normalized_tnppte.shape)
```

```
(73196, 1)
(49041, 1)
(24155, 1)
(36052, 1)
```

In [0]:
```
project_data['project_is_approved'].shape
```

Out[0]:
```
(109248,)
```

In [0]:
```
Y_train.shape
```

Out[0]:
```
(73196,)
```

In [0]:
```
Y_train_cv.shape
```

Out[0]:
```
(49041,)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

```
In [0]: # please write all the code with proper documentation, and proper titles for each subsection
        # 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 debugging your code
        # 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 reader
            # b. Legends if needed
            # c. X-axis label
            # d. Y-axis label
```

Bag of words

```
In [0]: # We are considering only the words which appeared in at least 10 documents(rows or projects).
        vectorizer_b = CountVectorizer()
        text_bow = vectorizer_b.fit(X_train_pe)
        text_bow = vectorizer_b.transform(X_train_pe)
        print("Shape of matrix after one hot encodig ",text_bow.shape)

        text_bow_cv=vectorizer_b.transform(X_train_pecv)
        print("Shape of matrix after one hot encodig ",text_bow_cv.shape)

        text_bow_te = vectorizer_b.transform(X_test_pe)
        print("Shape of matrix after one hot encodig ",text_bow_te.shape)

        text_bow_tecv = vectorizer_b.transform(X_test_pecv)
        print("Shape of matrix after one hot encodig ",text_bow_tecv.shape)
```

```
Shape of matrix after one hot encodig  (73196, 48013)
Shape of matrix after one hot encodig  (49041, 48013)
Shape of matrix after one hot encodig  (36052, 48013)
Shape of matrix after one hot encodig  (24155, 48013)
```

```
In [0]: #bow of Project_titles
```

```
In [0]: vectorizer_t = CountVectorizer()
        titles_bow = vectorizer_t.fit_transform(X_train_pt)
        print("Shape of matrix after one hot encodig ",titles_bow.shape)


        titles_bow_cv = vectorizer_t.transform(X_train_ptcv)
        print("Shape of matrix after one hot encodig ",titles_bow_cv.shape)

        titles_bow_te = vectorizer_t.transform(X_test_pt)
        print("Shape of matrix after one hot encodig ",titles_bow_te.shape)

        titles_bow_tecv = vectorizer_t.transform(X_test_ptcv)
        print("Shape of matrix after one hot encodig ",titles_bow_tecv.shape)
```

combining data

```
In [0]: %time
        from scipy.sparse import hstack
        #with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
        x_train= hstack(( categories_one_hot,sub_categories_one_hot,teacher_prefix_one_hot,school_state_one_hot,project_grade_
        category_one_hot,text_bow,titles_bow,price_normalized,normalized_tnpp)).tocsr()
        #x_train = x_train.toarray()
        #x_train[np.isnan(x_train)] = np.median(x_train[~np.isnan(x_train)])
        x_train.shape
```

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 8.58 µs
```

```
Out[0]: (73196, 62198)
```

```
In [0]: from scipy.sparse import hstack
        # with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
        x_train_cv= hstack((categories_one_hot_cv, sub_categories_one_hot_cv,teacher_prefix_one_hot_cv,school_state_one_hot_cv
        ,project_grade_category_one_hot_cv,text_bow_cv,titles_bow_cv,price_normalized_cv,normalized_tnppcv)).tocsr()
        #x_train_cv = x_train_cv.toarray()
        #x_train_cv[np.isnan(x_train_cv)] = np.median(x_train_cv[~np.isnan(x_train_cv)])
        x_train_cv.shape
```

```
Out[0]: (49041, 62198)
```

In [0]:
```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
x_test= hstack((categories_one_hot_te, sub_categories_one_hot_te,teacher_prefix_one_hot_te,school_state_one_hot_te,pro
ject_grade_category_one_hot_te,text_bow_te,titles_bow_te,price_normalized_te,normalized_tnppte)).tocsr()
#x_test = x_test.toarray()
#x_test[np.isnan(x_test)] = np.median(x_test[~np.isnan(x_test)])
x_test.shape
```

Out[0]:
```
(36052, 62198)
```

In [0]:
```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
x_test_cv= hstack((categories_one_hot_tecv, sub_categories_one_hot_tecv,teacher_prefix_one_hot_tecv,school_state_one_h
ot_tecv,project_grade_category_one_hot_tecv,text_bow_tecv,titles_bow_tecv,price_normalized_tecv,normalized_tnpptecv)).
tocsr()
#x_test_cv= x_test_cv.toarray()
#x_test_cv[np.isnan(x_test_cv)] = np.median(x_test_cv[~np.isnan(x_test_cv)])
x_test_cv.shape
```

Out[0]:
```
(24155, 62198)
```

In [0]:
```
print("Final Data matrix")
print(x_train.shape, Y_train.shape)
print(x_test_cv.shape, Y_test_cv.shape)
print(x_test.shape, Y_test.shape)
```

```
Final Data matrix
(73196, 62198) (73196,)
(24155, 62198) (24155,)
(36052, 62198) (36052,)
```

2.4.1 Applying Naive Bayes on BOW, SET 1

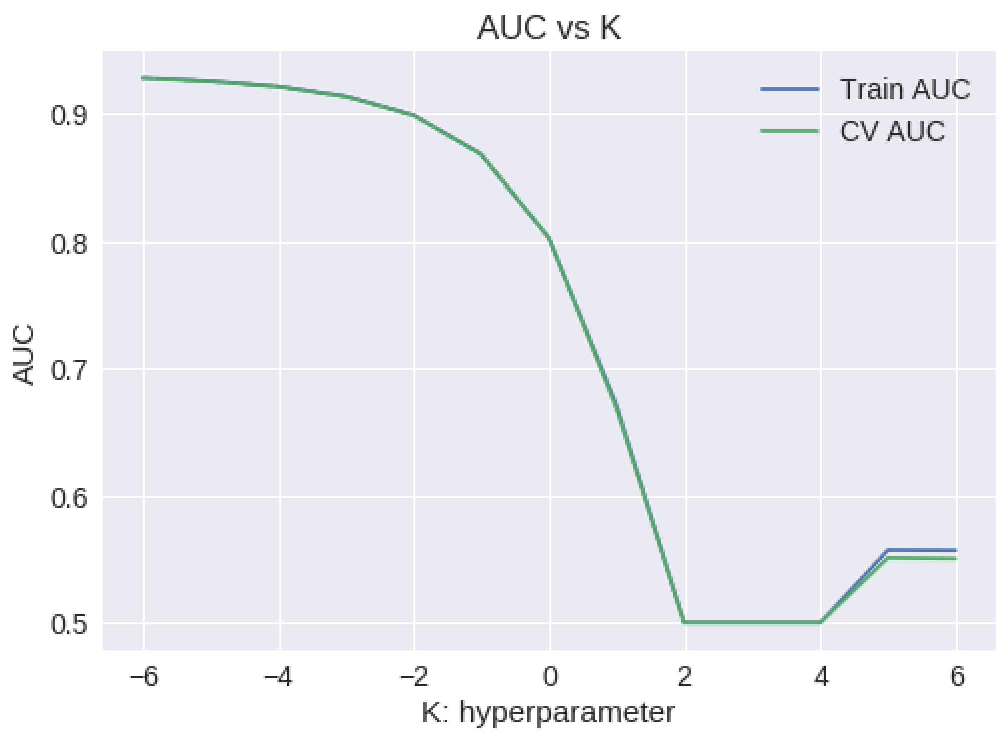
In [0]:
```
x_train_cv.shape
```

Out[0]:
```
(49041, 62198)
```

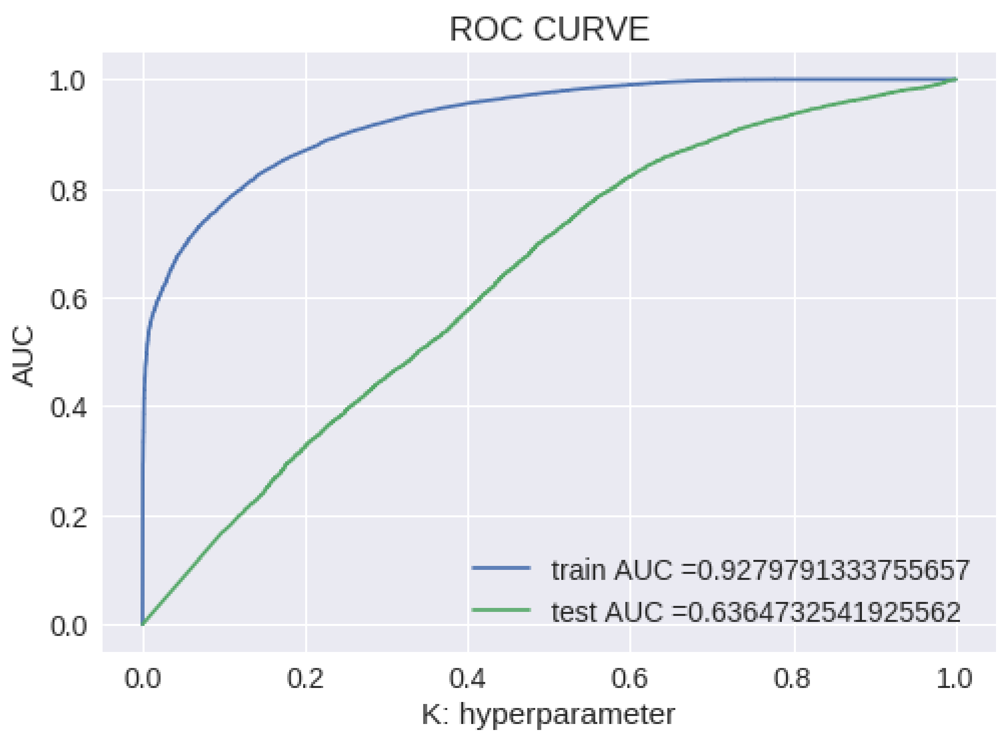
In [0]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
import random
import math
from sklearn import metrics
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,precision_score,recall_score
train_auc = []
cv_auc = []
alpha = [0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000,10000,100000,1000000]
log_a = [math.log10(num) for num in alpha]
for i in alpha:
    clf = MultinomialNB(alpha=i)
    clf.fit(x_train, Y_train)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
    # not the predicted outputs
    y_train_pred =  clf.predict_proba(x_train)[:,1]
    y_cv_pred =  clf.predict_proba(x_test_cv)[:,1]
    train_auc_score = roc_auc_score(Y_train,y_train_pred)
    train_auc.append((train_auc_score))

    cv_auc_score=roc_auc_score(Y_test_cv, y_cv_pred)
    cv_auc.append(cv_auc_score)
    print("alpha=",i,"cv:",cv_auc_score,"train:",train_auc_score)
```

```
alpha= 1e-06 cv: 0.9277545634438902 train: 0.9279791333755657
alpha= 1e-05 cv: 0.9253336944156472 train: 0.9255176331152208
alpha= 0.0001 cv: 0.921016709973987 train: 0.921155603191449
alpha= 0.001 cv: 0.9132464529563333 train: 0.9133387221461959
alpha= 0.01 cv: 0.8985242197763842 train: 0.8985226218484524
alpha= 0.1 cv: 0.8679578976811672 train: 0.867733332659405
alpha= 1 cv: 0.8029154632194226 train: 0.8027542357326178
alpha= 10 cv: 0.6684662600955737 train: 0.6712509477208973
alpha= 100 cv: 0.5 train: 0.5
alpha= 1000 cv: 0.5 train: 0.5
alpha= 10000 cv: 0.5 train: 0.5
alpha= 100000 cv: 0.5508674296985719 train: 0.5570217192382747
alpha= 1000000 cv: 0.5503226289455313 train: 0.5567117906032986
```

In [0]:
```
alpha = [0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000,10000,100000,1000000]
log_a = [math.log10(num) for num in alpha]
plt.plot(log_a, train_auc, label='Train AUC')
plt.plot(log_a, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs K")
plt.show()
```



In [0]:
```
clf = MultinomialNB(alpha=0.000001)
clf.fit(x_train, Y_train)

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

train_fpr, train_tpr, thresholds = roc_curve(Y_train, clf.predict_proba(x_train)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_test, clf.predict_proba(x_test)[:,1])

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("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ROC CURVE")
plt.show()

print("="*100)

from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(Y_train, clf.predict(x_train)))
print("Test confusion matrix")
print(confusion_matrix(Y_test, clf.predict(x_test)))
```

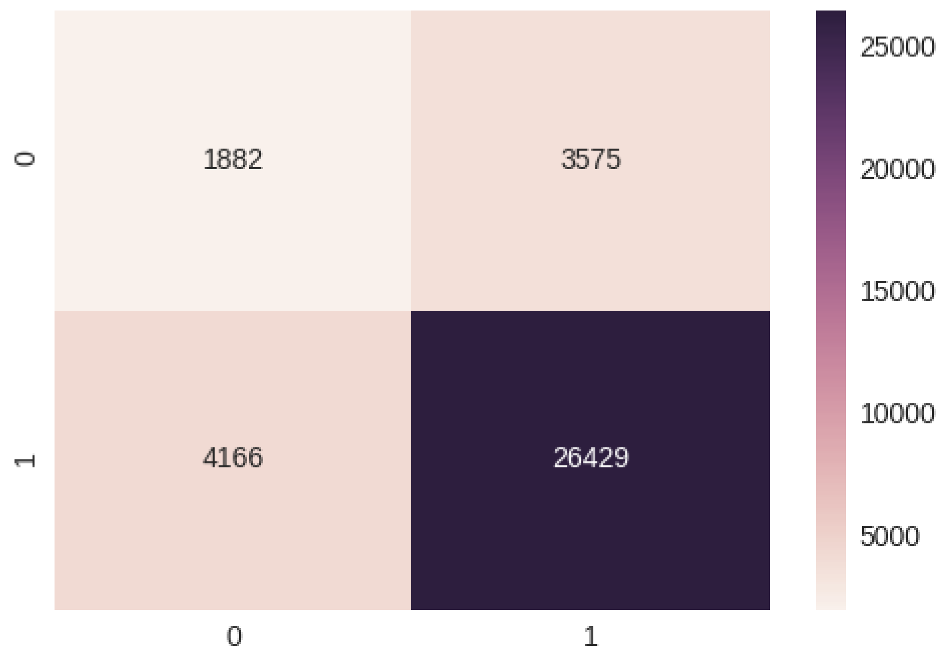


```
====================================================================================================
Train confusion matrix
[[ 8236  2849]
 [ 5959 56152]]
Test confusion matrix
[[ 1882  3575]
 [ 4166 26429]]
```

In [0]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
print('confusion matrix on test data')
y_new_pred  = clf.predict(x_test)
df_cm_bow = pd.DataFrame(confusion_matrix(Y_test, y_new_pred))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm_bow, annot=True,annot_kws={"size": 14}, fmt='g')
```

```
confusion matrix on test data
```

Out[0]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4820a0a048>
```



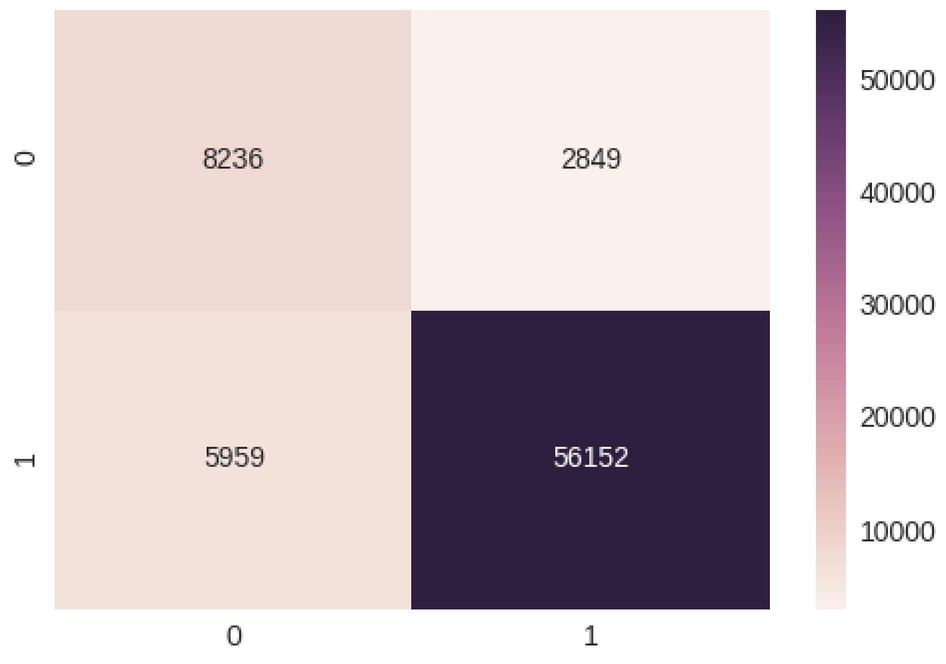
- **From the confusion matrix for test data we can say that,**

  26429+1882 =28311 pouns are correctly classified and 4166+3575 = 7741 points are wrongly classified

In [0]:
```
print("confusion matrix on train data")
y_new_pred_tr  = clf.predict(x_train)
df_cm_bow_tr = pd.DataFrame(confusion_matrix(Y_train, y_new_pred_tr))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm_bow_tr, annot=True,annot_kws={"size": 14}, fmt='g')
```

```
confusion matrix on train data
```

Out[0]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f481179e5f8>
```



- **From the confusion matrix for train data we can say that,**

  56152+8236 = 64388 pouns are correctly classified and 5959+2849= 8808 points are wrongly classified

```
In [0]: from sklearn.metrics import accuracy_score
        from sklearn.metrics import precision_score
        from sklearn.metrics import f1_score
        from sklearn.metrics import recall_score



        print("Accuracy: %f%%"%(accuracy_score(Y_test, y_new_pred)*100))
        print("Precision: %f"%(precision_score(Y_test, y_new_pred)))
        print("Recall: %f"%(recall_score(Y_test, y_new_pred)))
        print("F1-Score: %f"%(f1_score(Y_test, y_new_pred)))
```

```
Accuracy: 78.528237%
Precision: 0.880849
Recall: 0.863834
F1-Score: 0.872259
```

Top 10 postive & Negative Features(Bag of words)

```
In [0]: features_b= vectorizer_b.get_feature_names()
        len(features_b)

Out[0]: 48013
```

```
In [0]: features_t = vectorizer_t.get_feature_names()
        len(features_t)

Out[0]: 14083
```

```
In [0]: features = features_b + features_t + features_t[0:102]
        len(features)

Out[0]: 62198
```

```
In [0]: log_prob=clf.feature_log_prob_
        #https://stackoverflow.com/questions/51209933/convert-list-to-column-in-python-dataframe
        #for converting to column's we require a dataframe to store these list's
        df = pd.DataFrame(log_prob,columns = features)
        print(df.shape)
        df_t = df.T
        print(df_t.shape)
        print(df_t[1].sort_values(ascending = False)[:10])
```

```
Max bow feature_log_probability's -3.0901827930526924

(2, 62198)
(62198, 2)
styled       -3.090183
scoop        -4.232883
namle        -4.548803
leguminous   -4.598899
cleveland    -4.621619
theraputty   -4.853099
thoughout    -4.888674
nourishing   -4.892192
leggings     -4.938370
herring      -4.966418
Name: 1, dtype: float64
```

```
In [0]: print(df_t[0].sort_values(ascending = False)[:10])
```

```
styled       -3.101053
scoop        -4.201234
leguminous   -4.515295
namle        -4.557106
cleveland    -4.672140
nourishing   -4.858780
leggings     -4.874248
thoughout    -4.890511
herring      -4.900174
theraputty   -4.919380
Name: 0, dtype: float64
```

Encoding Project_essay and project_titles using TFIDF vectorizer and applying Multinomial Naive Bayes

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_b = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer_tfidf_b.fit_transform(X_train_pe)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)


text_tfidf_te = vectorizer_tfidf_b.transform(X_test_pe)
print("Shape of matrix after one hot encodig ",text_tfidf_te.shape)

text_tfidf_cv = vectorizer_tfidf_b.transform(X_train_pecv)
print("Shape of matrix after one hot encodig ",text_tfidf_cv.shape)

text_tfidf_tecv = vectorizer_tfidf_b.transform(X_test_pecv)
print("Shape of matrix after one hot encodig ",text_tfidf_tecv.shape)
```

```
Shape of matrix after one hot encodig  (73196, 14214)
Shape of matrix after one hot encodig  (36052, 14214)
Shape of matrix after one hot encodig  (49041, 14214)
Shape of matrix after one hot encodig  (24155, 14214)
```

In [0]:
```
# Similarly you can vectorize for title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_t = TfidfVectorizer(min_df=10)
titles_tfidf = vectorizer_tfidf_t.fit_transform(X_train_pt)
print("Shape of matrix after one hot encodig ",titles_tfidf.shape)

titles_tfidf_cv = vectorizer_tfidf_t.transform(X_train_ptcv)
print("Shape of matrix after one hot encodig ",titles_tfidf_cv.shape)


titles_tfidf_te = vectorizer_tfidf_t.transform(X_test_pt)
print("Shape of matrix after one hot encodig ",titles_tfidf_te.shape)

titles_tfidf_tecv = vectorizer_tfidf_t.transform(X_test_ptcv)
print("Shape of matrix after one hot encodig ",titles_tfidf_tecv.shape)
```

```
Shape of matrix after one hot encodig  (73196, 2600)
Shape of matrix after one hot encodig  (49041, 2600)
Shape of matrix after one hot encodig  (36052, 2600)
Shape of matrix after one hot encodig  (24155, 2600)
```

In [0]:

2.4.1 Combining all features,TFIDF SET 2

In [0]:
```
from scipy.sparse import hstack
#with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
x_train_tfidf= hstack(( categories_one_hot,sub_categories_one_hot,teacher_prefix_one_hot,school_state_one_hot,project_
grade_category_one_hot,text_tfidf,titles_tfidf,price_normalized,normalized_tnpp)).tocsr()
#x_train = x_train.toarray()
#x_train[np.isnan(x_train)] = np.median(x_train[~np.isnan(x_train)])
x_train_tfidf.shape
```

Out[0]:
```
(73196, 16916)
```

In [0]:
```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
x_train_tfidf_cv= hstack((categories_one_hot_cv, sub_categories_one_hot_cv,teacher_prefix_one_hot_cv,school_state_one_
hot_cv,project_grade_category_one_hot_cv,text_tfidf_cv,titles_tfidf_cv,price_normalized_cv,normalized_tnppcv)).tocsr()
#x_train_cv = x_train_cv.toarray()
#x_train_cv[np.isnan(x_train_cv)] = np.median(x_train_cv[~np.isnan(x_train_cv)])
x_train_tfidf_cv.shape
```

Out[0]:
```
(49041, 16916)
```

In [0]:
```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
x_test_tfidf= hstack((categories_one_hot_te, sub_categories_one_hot_te,teacher_prefix_one_hot_te,school_state_one_hot_
te,project_grade_category_one_hot_te,text_tfidf_te,titles_tfidf_te,price_normalized_te,normalized_tnppte)).tocsr()
#x_test = x_test.toarray()
#x_test[np.isnan(x_test)] = np.median(x_test[~np.isnan(x_test)])
x_test_tfidf.shape
```

Out[0]:
```
(36052, 16916)
```

In [0]:
```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
x_test_tfidf_cv= hstack((categories_one_hot_tecv, sub_categories_one_hot_tecv,teacher_prefix_one_hot_tecv,school_state
_one_hot_tecv,project_grade_category_one_hot_tecv,text_tfidf_tecv,titles_tfidf_tecv,price_normalized_tecv,normalized_t
npptecv)).tocsr()
#x_test_cv= x_test_cv.toarray()
#x_test_cv[np.isnan(x_test_cv)] = np.median(x_test_cv[~np.isnan(x_test_cv)])
x_test_tfidf_cv.shape
```

Out[0]:
```
(24155, 16916)
```

In [0]:
```
print("Final Data matrix")
print(x_train_tfidf.shape, Y_train.shape)
print(x_test_tfidf_cv.shape, Y_test_cv.shape)
print(x_test_tfidf.shape, Y_test.shape)
```

```
Final Data matrix
(73196, 16916) (73196,)
(24155, 16916) (24155,)
(36052, 16916) (36052,)
```

In [0]:
```
```

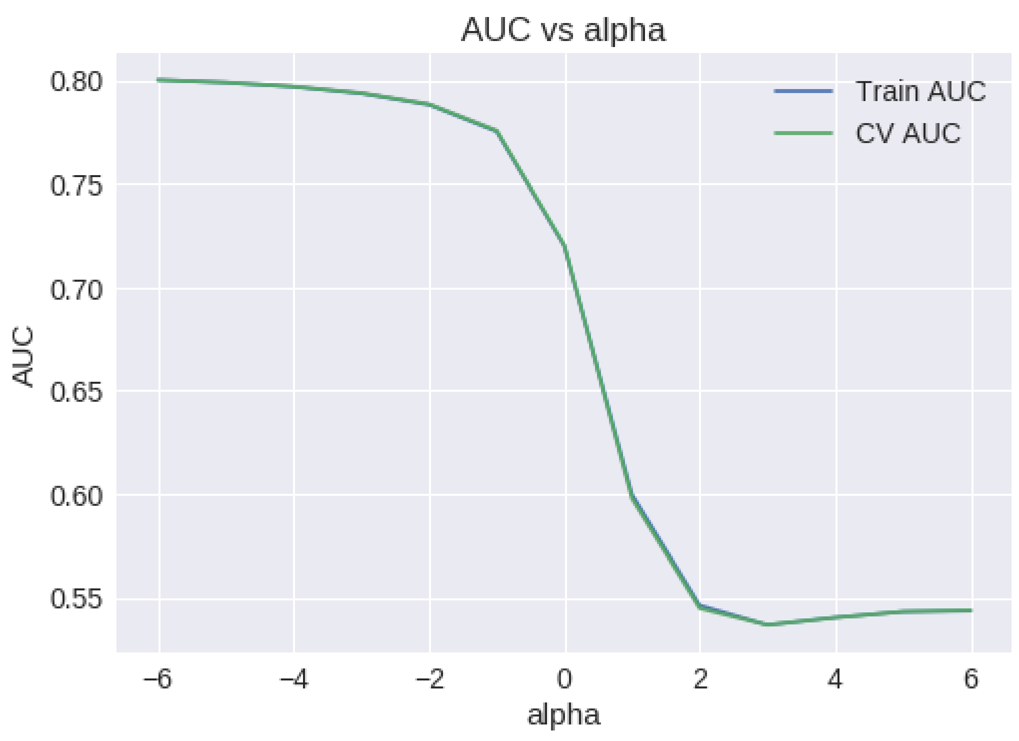
2.4.1 Applying Naive Bayes on TFIDF, SET 2

In [0]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.metrics import roc_auc_score
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
import random
import math
from sklearn import metrics
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,precision_score,recall_score
train_auc = []
cv_auc = []
a = [0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000,10000,100000,1000000]
log_a = [math.log10(num) for num in alpha]
for i in alpha:
    clf = MultinomialNB(alpha=i)
    clf.fit(x_train_tfidf, Y_train)
    y_train_pred =  clf.predict_proba(x_train_tfidf)[:,1]
    y_cv_pred =  clf.predict_proba(x_test_tfidf_cv)[:,1]
    train_auc_score = roc_auc_score(Y_train,y_train_pred)
    train_auc.append((train_auc_score))
    cv_auc.append(roc_auc_score(Y_test_cv, y_cv_pred))
    cv_auc_score=roc_auc_score(Y_test_cv, y_cv_pred)
    print("alpha=",i,"cv:",cv_auc_score,"train:",train_auc_score)
```

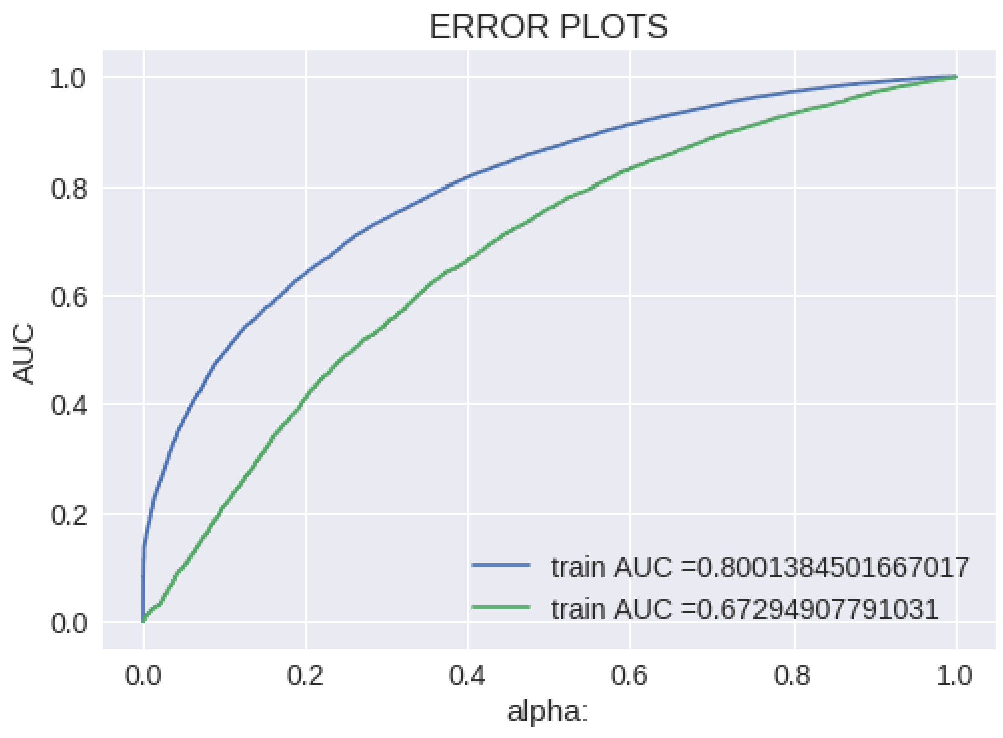
```
alpha= 1e-06 cv: 0.8001838601102305 train: 0.8001384501667017
alpha= 1e-05 cv: 0.7989722564662647 train: 0.7988884988867146
alpha= 0.0001 cv: 0.7969959731316906 train: 0.7968732743066458
alpha= 0.001 cv: 0.7938301895844451 train: 0.7936408725725785
alpha= 0.01 cv: 0.7884599085569906 train: 0.788166374942087
alpha= 0.1 cv: 0.7756720075819034 train: 0.7752264630014358
alpha= 1 cv: 0.7202693100224797 train: 0.7200237426139027
alpha= 10 cv: 0.5977425766994567 train: 0.5997158520313788
alpha= 100 cv: 0.5448399051359791 train: 0.5462610746208171
alpha= 1000 cv: 0.5368008032719791 train: 0.5368375606037199
alpha= 10000 cv: 0.5402883588613523 train: 0.5404242729345552
alpha= 100000 cv: 0.5432399303121589 train: 0.543201549320735
alpha= 1000000 cv: 0.5438586338258368 train: 0.5436539034285433
```

- **Here we took alpha values in the range of 0.000001 to 1000000**
- **The optimal alpha is observed at alpha value of 0.000001**
- **we can observe that for higher values of alpha the cv is decreasing**

In [0]:
```
a = [0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000,10000,100000,1000000]
log_a = [math.log10(num) for num in alpha]
plt.plot(log_a, train_auc, label='Train AUC')
plt.plot(log_a, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("alpha")
plt.ylabel("AUC")
plt.title("AUC vs alpha")
plt.show()
```



In [0]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

clf = MultinomialNB(alpha =0.000001)
clf.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
train_fpr, train_tpr, thresholds = roc_curve(Y_train, clf.predict_proba(x_train_tfidf)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_test, clf.predict_proba(x_test_tfidf)[:,1])
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha:")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(Y_train, clf.predict(x_train_tfidf)))
print("Test confusion matrix")
print(confusion_matrix(Y_test, clf.predict(x_test_tfidf)))
```



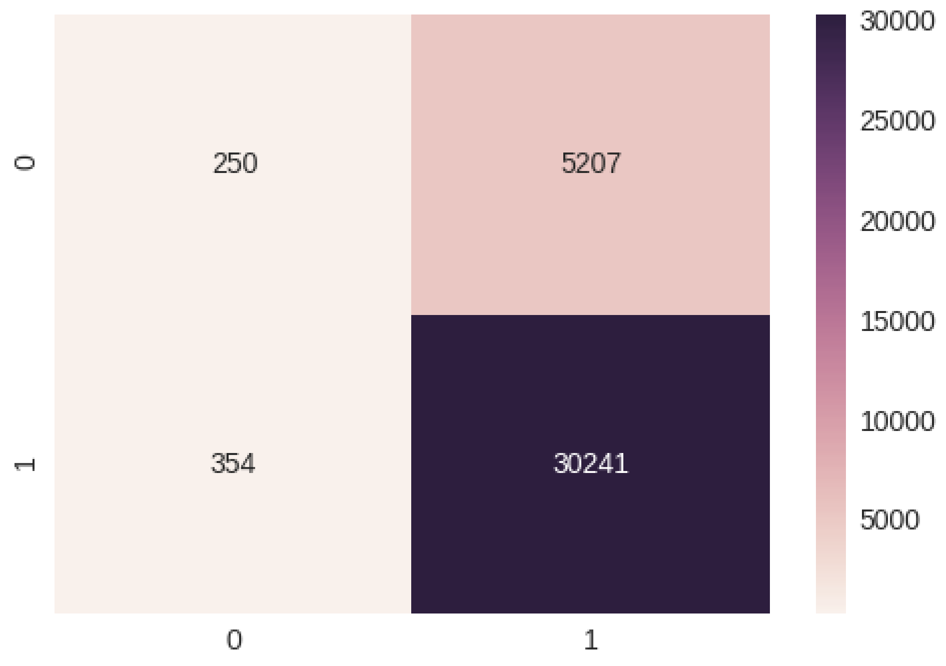
```
====================================================================================================
Train confusion matrix
[[ 1098  9987]
 [  607 61504]]
Test confusion matrix
[[  250  5207]
 [  354 30241]]
```

- **here we got train accuracy as 80% and test accuracy as 67%**

In [0]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
print("confusion matrix on test data")
y_tfidf_pred  = clf.predict(x_test_tfidf)
df_cm_bow = pd.DataFrame(confusion_matrix(Y_test, y_tfidf_pred))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm_bow, annot=True,annot_kws={"size": 14}, fmt='g')
```

```
confusion matrix on test data
```

Out[0]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f481d7f5240>
```



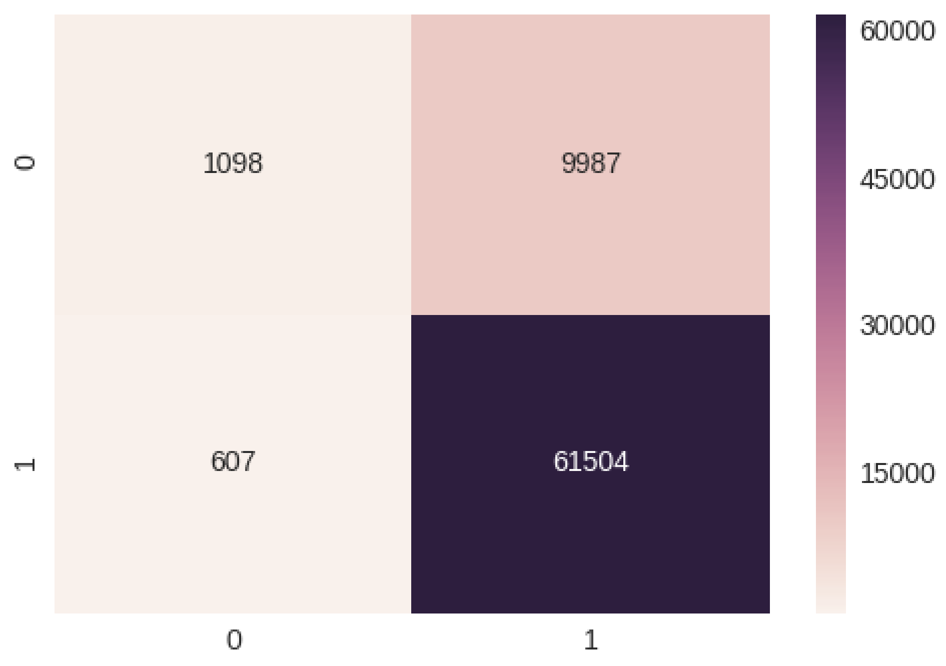
- **From the confusion matrix for test data we can say that,**

  30241 + 250 = 30491 pouns are correctly classified and 5207+354 = 5561 points are wrongly classified

In [0]:
```
print("confusion matrix on train data")
y_new_pred_tr  = clf.predict(x_train_tfidf)
df_cm_bow_tr = pd.DataFrame(confusion_matrix(Y_train, y_new_pred_tr))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm_bow_tr, annot=True,annot_kws={"size": 14}, fmt='g')
```

```
confusion matrix on train data
```

Out[0]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f481d6fad68>
```



- **From the confusion matrix for train data we can say that,**

  61504+1098 = 62612 pouns are correctly classified and 9987+607 = 10594 points are wrongly classified

In [0]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
print("Accuracy: %f%%"%(accuracy_score(Y_test, y_tfidf_pred)*100))
print("Precision: %f"%(precision_score(Y_test, y_tfidf_pred)))
print("Recall: %f"%(recall_score(Y_test, y_tfidf_pred)))
print("F1-Score: %f"%(f1_score(Y_test, y_tfidf_pred)))
```

```
Accuracy: 84.575058%
Precision: 0.853109
Recall: 0.988429
F1-Score: 0.915797
```

Top 10 Positive and Negative Features from TFIDF

In [0]:
```
##https://www.kaggle.com/premvardhan/amazon-fine-food-reviews-analysis-naive-bayes
#https://stackoverflow.com/questions/51209933/convert-list-to-column-in-python-dataframe
#for converting to column's we require a dataframe to store these list's
features_b= vectorizer_tfidf_b.get_feature_names()
len(features_b)
```

Out[0]:
```
14214
```

In [0]:
```
features_t = vectorizer_tfidf_t.get_feature_names()
len(features_t)
```

Out[0]:
```
2600
```

In [0]:
```
features = features_b + features_t + features_t[0:102]
len(features)
```

Out[0]:
```
16916
```

In [0]:
```
log_prob=clf.feature_log_prob_
print("Max bow feature_log_probability's",log_prob.max(),"\n")
df = pd.DataFrame(log_prob,columns = features)
print(df.shape)
df_t = df.T
print(df_t.shape)
print(df_t[1].sort_values(ascending = False)[:10])
```

```
Max bow feature_log_probability's -2.9022060657263733

(2, 16916)
(16916, 2)
animals     -2.916088
225         -2.916088
animation   -3.226310
1600        -3.558492
103         -3.632280
102         -3.895699
160         -3.945459
150         -4.059733
15          -4.276556
14th        -4.497677
Name: 1, dtype: float64
```

In [0]:
```
print(df_t[0].sort_values(ascending = False)[:10])
```

```
animals     -2.902206
225         -2.902206
animation   -3.288506
1600        -3.592741
103         -3.758536
102         -3.824832
160         -3.897300
15          -4.246542
150         -4.253835
14th        -4.560164
Name: 0, dtype: float64
```

In [2]:
```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["vectorizer", "model", "hyperparameter", " Train AUC", "TEST AUC"]

x.add_row(["BOW", "Naive Bayes", "0.000001",0.92 ,0.63 ])
x.add_row(["TFIDF", "Naive Bayes", "0.000001", 0.80,0.67])

print(x)
```

```
+------------+-------------+----------------+------------+----------+
| vectorizer |    model    | hyperparameter |  Train AUC | TEST AUC |
+------------+-------------+----------------+------------+----------+
|    BOW     | Naive Bayes |    0.000001    |    0.92    |   0.63   |
|   TFIDF    | Naive Bayes |    0.000001    |    0.8     |   0.67   |
+------------+-------------+----------------+------------+----------+
```

Observations

- The entire data set is considered which has more than 100k points
- The data was splitted into train and test with size of 0.66 and 0.33
- The traindata is again splitted into train and test cross validated data

Bag Of words

- Tha optimal alpha value is 0.000001
- From the confusion matrix,
- For Train Data,
  - 56152+8236 = 64388 are correctly classified
  - 5959+2849= 8808 are wrongly classified
- For Test Data,
  - 26429+1882 =28311 are correctly classified
  - 4166+3575 = 7741 are wrongly classified
- The perfomance parameters:
- Accuracy using BOW is 78.52%
- Precision using BOW is 0.88
- Recall using BOW is 0.86
- F1 score using BOW is 0.87
- The Top 10 features :
  - The max feature log probality for positive class is -3.09 and the least is -4.966
  - The max feature log probality for negative class is -3.10and the least is -4.91

TFIDF

- The optimal alpha value is 0.000001
- From the confusion matrix,
- For Train Data,
  - 61504+1098 = 62612 are correctly classified
  - 9987+607 = 10594 are wrongly classified
- For Test Data,
  - 30241 + 250 = 30491 are correctly classified
  - 5207+354 = 5561 are wrongly classified
- The perfomance parameters:
- Accuracy using TFIDF is 84.575%
- Precision using TFIDF is 0.85
- Recall using TFIDFis 0.98
- F1 score using TFIDF is 0.91
- The Top 10 features :
  - The max feature log probality for positive class is -2.91 and the least is -4.49
  - The max feature log probality for negative class is -2.90 and the least is -4.50

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]: