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	Desc
`project_id`	A unique identifier for the proposed project. **Example:** `p0`
`project_title`	Title of the project. **Exa** • `Art Will Make You H` • `First Grad`
`project_grade_category`	Grade level of students for which the project is targeted. One of the fo enumerated v • `Grades P` • `Grade` • `Grade` • `Grades`
`project_subject_categories`	One or more (comma-separated) subject categories for the project fr following enumerated list of v • `Applied Lea` • `Care & H` • `Health & S` • `History & C` • `Literacy & Lan` • `Math & Sc` • `Music & The` • `Special` • `W` **Exar** • `Music & The` • `Literacy & Language, Math & Sc`

Feature	Desc
`school_state`	State where school is located ([Two-letter U.S. posta (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_c](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_c) **Exampl**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the **Exar** • `Lit` • `Literature & Writing, Social Sci`
`project_resource_summary`	An explanation of the resources needed for the project. **Exa** • `My students need hands on literacy materials to ma sensory needs!<`
`project_essay_1`	First application
`project_essay_2`	Second application
`project_essay_3`	Third application
`project_essay_4`	Fourth application
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-( 12:43:5`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Exa** `bdf8baa8fedef6bfeec7ae4ff1c`
`teacher_prefix`	Teacher's title. One of the following enumerated v • • • • • • `Tea`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same te **Examp**

* 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]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code (https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:
··········
Mounted at /content/drive
```

In [2]:

```
%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

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]:

```
project_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/train_data.csv', nrows=
resource_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/resources.csv')
```

In [4]:

```
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 (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 's
chool_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 [5]:

```
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[5]:

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

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/473019

# 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", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        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/473019

# 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", '
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        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 [9]:

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

Out[9]:

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

In [0]:

```
#### 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])
```

```
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 hav
e over 24 languages represented in our English Learner program with students
at every level of mastery.  We also have over 40 countries represented with
the families within our school.  Each student brings a wealth of knowledge a
nd experiences to us that open our eyes to new cultures, beliefs, and respec
t.\"The limits of your language are the limits of your world.\"-Ludwig Wittg
enstein  Our English learner's have a strong support system at home that beg
s for more resources.  Many times our parents are learning to read and speak
English along side of their children.  Sometimes this creates barriers for p
arents to be able to help their child learn phonetics, letter recognition, a
nd other reading skills.\r\n\r\nBy providing these dvd's and players, studen
ts are able to continue their mastery of the English language even if no one
at home is able to assist.  All families with students within the Level 1 pr
oficiency status, will be a offered to be a part of this program.  These edu
cational videos will be specially chosen by the English Learner Teacher and
will be sent home regularly to watch.  The videos are to help the child deve
lop early reading skills.\r\n\r\nParents that do not have access to a dvd pl
ayer 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 c
ome for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year a
ll love learning, at least most of the time. At our school, 97.3% of the stu
dents receive free or reduced price lunch. Of the 560 students, 97.3% are mi
nority students. \r\nThe school has a vibrant community that loves to get to
gether and celebrate. Around Halloween there is a whole school parade to sho
w off the beautiful costumes that students wear. On Cinco de Mayo we put on
a big festival with crafts made by the students, dances, and games. At the e
nd of the year the school hosts a carnival to celebrate 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 regul
ar, stationary, 4-legged chairs. As I will only have a total of ten in the c
lassroom and not enough for each student to have an individual one, they wil
l be used in a variety of ways. During independent reading time they will be
used as special chairs students will each use on occasion. I will utilize th
em in place of chairs at my small group tables during math and reading time
s. The rest of the day they will be used by the students who need the highes
t amount of movement in their 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. Whe
n the students are sitting in group with me on the Hokki Stools, they are al
ways 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. Ther
e are always students who head over to the kidney table to get one of the st
ools 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 comprom
ise 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 movement by allo
wing them to activate their core muscles for balance while they sit. For man
y of my students, these chairs will take away the barrier that exists in sch
ools for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment wit
h plain walls, rows of desks, and a teacher 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 cl
ass is made up of 28 wonderfully unique boys and girls of mixed races in Ark
ansas.\r\nThey attend a Title I school, which means there is a high enough p
ercentage of free and reduced-price lunch to qualify. Our school is an \"ope
n classroom\" concept, which is very unique as there are no walls separating
the classrooms. These 9 and 10 year-old students are very eager learners; th
ey are like sponges, 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 nauti
cal environment. Creating a classroom environment is very important in the s
uccess 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 them, hav
e them developed, and then hung in our classroom ready for their first day o
f 4th grade.  This kind gesture will 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\nYou
r 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 of m
y own pocket on resources to get our classroom ready. Please consider helpin
g 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 la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations. \r\n\r\nThe materials we have are the ones I seek out for my stu
dents. 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 ever 
felt like you had ants in your pants and you needed to groove and move as yo
u were in a meeting? This is how my kids feel all the time. The want to be a
ble to move as they learn 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. \r\nThey also want to learn through games, my kids d
on't want to sit and do worksheets. They want to learn to count by jumping a
nd playing. Physical engagement is the key to our success. The number toss a
nd color and shape mats can make that happen. My students will forget they a
re doing work and just have the 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 [12]:

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

```
My kindergarten students have varied disabilities ranging from speech and la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations. \r\n\r\nThe materials we have are the ones I seek out for my stu
dents. 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 ever
felt like you had ants in your pants and you needed to groove and move as yo
u were in a meeting? This is how my kids feel all the time. The want to be a
ble to move as they learn 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. \r\nThey also want to learn through games, my kids d
o 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 will forget they
are doing work and just have the fun a 6 year old deserves.nannan
==================================================
```

In [13]:

```
# \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 la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations.     The materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or re
duced 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 wer
e in a meeting? This is how my kids feel all the time. The want to be able t
o move as they learn 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 w
ant to sit and do worksheets. They want to learn to count by jumping and pla
ying. Physical engagement is the key to our success. The number toss and col
or and shape mats can make that happen. My students will forget they are doi
ng 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 la
nguage delays cognitive delays gross fine motor delays to autism They are ea
ger beavers and always strive to work their hardest working past their limit
ations 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 a
nts 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 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 work
sheets They want to learn to count by jumping and playing Physical engagemen
t is the key to our success The number toss and color and shape mats can mak
e that happen My students will forget they are doing work and just have the 
fun a 6 year old deserves nannan
```

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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            '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.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:27<00:00, 1830.79it/s]
```

In [17]:

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

Out[17]:

```
'kindergarten students varied disabilities ranging speech language delays co
gnitive delays gross fine motor delays autism eager beavers always strive wo
rk hardest working past limitations materials ones seek students teach title
school students receive free reduced price lunch despite disabilities limita
tions students love coming school come eager learn explore ever felt like an
ts pants needed groove move meeting kids feel time want able move learn say
wobble chairs answer love develop core enhances gross motor turn fine motor
skills also want learn games kids not want sit worksheets want learn count j
umping playing physical engagement key success number toss color shape mats
make happen students forget work fun 6 year old deserves nannan'
```

In [0]:

```
project_data['essay'] = preprocessed_essays
```

1.4 Preprocessing of `project_title`

In [0]:

```
# similarly you can preprocess the titles also
```

In [20]:

```
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%|██████████| 50000/50000 [00:01<00:00, 42874.93it/s]
```

In [0]:

```
project_data['project_title'] = preprocessed_titles
```

In [0]:

```
#Preprocessing project_grade_category

#reference link: https://stackoverflow.com/questions/28986489/python-pandas-how-to-replace-

project_data['project_grade_category'] = project_data['project_grade_category'].str.replace
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace
```

1.5 Preparing data for models

In [23]:

```
project_data.columns
```

Out[23]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       '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 [24]:

```
# 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, bina
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (50000, 9)
```

In [25]:

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducat
ion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Characte
rEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', '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  (50000, 30)
```

In [0]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

In [26]:

```
#school state
#Using CountVectorizer to convert values into one hot encoded
vectorizer = CountVectorizer(lowercase=False , binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())

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

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'I
A', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO',
'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'O
R', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV',
'WY']
Shape of matrix after one hot encoding (50000, 51)
```

In [27]:

```
#teacher_prefix
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values.astype('U'))
#While running this i got an error:np.nan is an invalid document, expected byte or unicode
#I fixed it by using stackoverflow.com
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np
print(vectorizer.get_feature_names())


teacher_prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].values.astype(
print('Shape of matrix of one hot encoding', teacher_prefix_one_hot.shape)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher', 'nan']
Shape of matrix of one hot encoding (50000, 6)
```

In [28]:

```
#project_grade_category
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values.astype('U'))
print(vectorizer.get_feature_names())


project_grade_category_one_hot = vectorizer.fit_transform(project_data['project_grade_categ
print('Shape of matrix of one hot encoding', project_grade_category_one_hot.shape)
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of matrix of one hot encoding (50000, 4)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [29]:

```
# We are considering only the words which appeared in at least 10 documents(rows or project
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  (50000, 12101)
```

In [0]:

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

In [30]:

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

```
Shape of matrix after one hot encodig  (50000, 2039)
```

1.5.2.2 TFIDF vectorizer

In [31]:

```
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  (50000, 12101)
```

In [32]:

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

```
Shape of matrix after one hot encodig  (50000, 2039)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [33]:

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

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

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

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

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

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

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

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


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickl

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


'''
```

Out[33]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4
```

```
084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGloveMod
el(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFil
e,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n
splitLine = line.split()\n        word = splitLine[0]\n        embedding = n
p.array([float(val) for val in splitLine[1:]])\n        model[word] = embedd
ing\n    print ("Done.",len(model)," words loaded!")\n    return model\nmode
l = 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.split(\' \'))\n\nfor i in preproce
d_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the cou
pus", len(words))\nwords = set(words)\nprint("the unique words in the coupu
s", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprin
t("The number of words that are present in both glove vectors and our coupu
s",      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,
3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in wo
rds:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint
("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pi
ckle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-
load-variables-in-python/\n\nimport (http://www.jessicayung.com/how-to-use-p
ickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith open(\'g
love_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/how-to-use-pickl
# make sure you have the glove_vectors file
with open('/content/drive/My Drive/Colab Notebooks/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [35]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
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%|██████████| 50000/50000 [00:14<00:00, 3481.03it/s]

50000
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 [37]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
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/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            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%|██████████| 50000/50000 [01:30<00:00, 550.56it/s]

50000
300
```

In [0]:

```
# Similarly you can vectorize for title also
```

In [38]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stored in this lis
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            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_titles.append(vector)

print(len(tfidf_w2v_vectors_titles))
print(len(tfidf_w2v_vectors_titles[0]))
```

```
100%|██████████| 50000/50000 [00:01<00:00, 25332.89it/s]

50000
300
```

1.5.3 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 [41]:

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

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

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

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

```
Mean : 299.33367619999996, Standard deviation : 378.20927190421384
```

In [42]:

```
price_standardized
```

Out[42]:

```
array([[-0.38268146],
       [-0.00088225],
       [ 0.57512161],
       ...,
       [-0.65382764],
       [-0.52109689],
       [ 0.54492668]])
```

1.5.4 Merging all the above features

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

In [43]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(50000, 9)
(50000, 30)
(50000, 12101)
(50000, 1)
```

In [44]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[44]:

```
(50000, 12141)
```

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsectic
# 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
```

__ Computing Sentiment Scores__

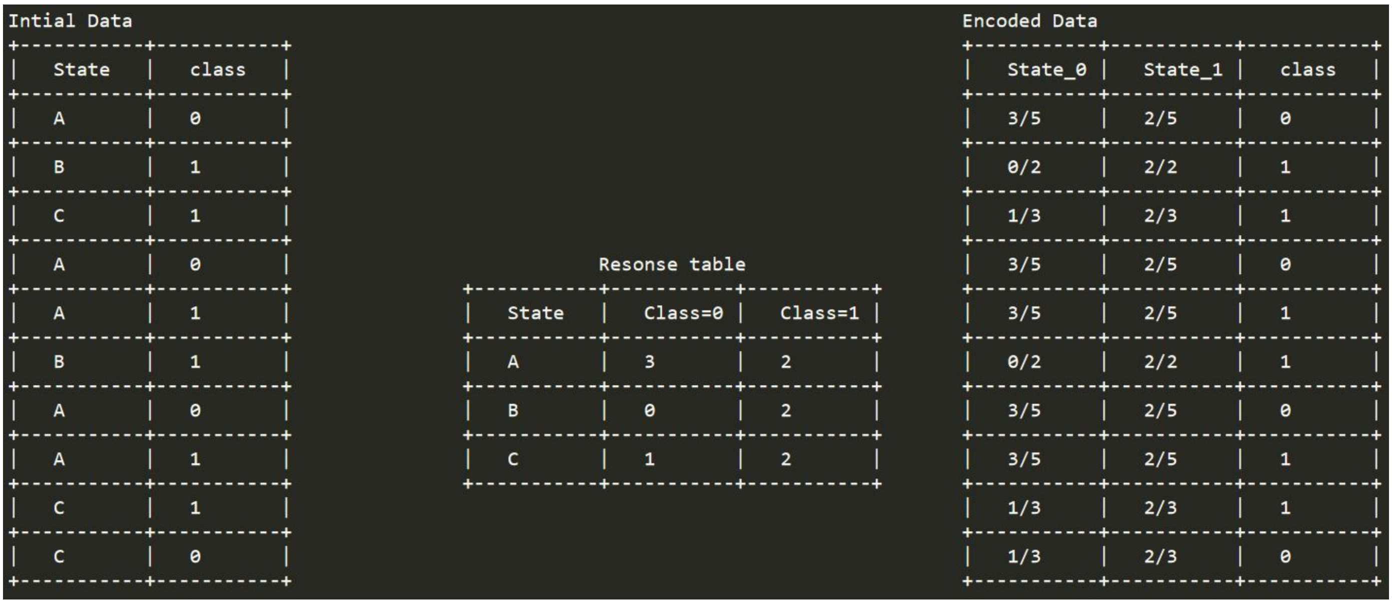
In [45]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
nltk.download()

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest stu
for learning my students learn in many different ways using all of our senses and multiple
of techniques to help all my students succeed students in my class come from a variety of d
for wonderful sharing of experiences and cultures including native americans our school is
learners which can be seen through collaborative student project based learning in and out
in my class love to work with hands on materials and have many different opportunities to p
mastered having the social skills to work cooperatively with friends is a crucial aspect of
montana is the perfect place to learn about agriculture and nutrition my students love to r
in the early childhood classroom i have had several kids ask me can we try cooking with rea
and create common core cooking lessons where we learn important math and writing concepts w
food for snack time my students will have a grounded appreciation for the work that went in
of where the ingredients came from as well as how it is healthy for their bodies this proje
nutrition and agricultural cooking recipes by having us peel our own apples to make homemad
and mix up healthy plants from our classroom garden in the spring we will also create our c
shared with families students will gain math and literature skills as well as a life long e
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> d

Download which package (l=list; x=cancel)?
  Identifier> vader_lexicon
    Downloading package vader_lexicon to /root/nltk_data...

---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> q
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

Assignment 9: RF and GBDT

Response Coding: Example



```
Intial Data
+----------+----------+
|  State   |  class   |
+----------+----------+
|  A       |  0       |
+----------+----------+
|  B       |  1       |
+----------+----------+
|  C       |  1       |
+----------+----------+
|  A       |  0       |
+----------+----------+
|  A       |  1       |
+----------+----------+
|  B       |  1       |
+----------+----------+
|  A       |  0       |
+----------+----------+
|  A       |  1       |
+----------+----------+
|  C       |  1       |
+----------+----------+
|  C       |  0       |
+----------+----------+

           Resonse table
+----------+----------+----------+
|  State   |  Class=0 |  Class=1 |
+----------+----------+----------+
|  A       |  3       |  2       |
+----------+----------+----------+
|  B       |  0       |  2       |
+----------+----------+----------+
|  C       |  1       |  2       |
+----------+----------+----------+

Encoded Data
+----------+----------+----------+
|  State_0 |  State_1 |  class   |
+----------+----------+----------+
|  3/5     |  2/5     |  0       |
+----------+----------+----------+
|  0/2     |  2/2     |  1       |
+----------+----------+----------+
|  1/3     |  2/3     |  1       |
+----------+----------+----------+
|  3/5     |  2/5     |  0       |
+----------+----------+----------+
|  3/5     |  2/5     |  1       |
+----------+----------+----------+
|  0/2     |  2/2     |  1       |
+----------+----------+----------+
|  3/5     |  2/5     |  0       |
+----------+----------+----------+
|  3/5     |  2/5     |  1       |
+----------+----------+----------+
|  1/3     |  2/3     |  1       |
+----------+----------+----------+
|  1/3     |  2/3     |  0       |
+----------+----------+----------+
```

> The response tabel is built only on train dataset. For a category which is not there in train data and present in test data, we will encode them with default values Ex: in our test data if have State: D then we encode it as [0.5, 0.05]

1. **Apply both Random Forrest and GBDT on these feature sets**

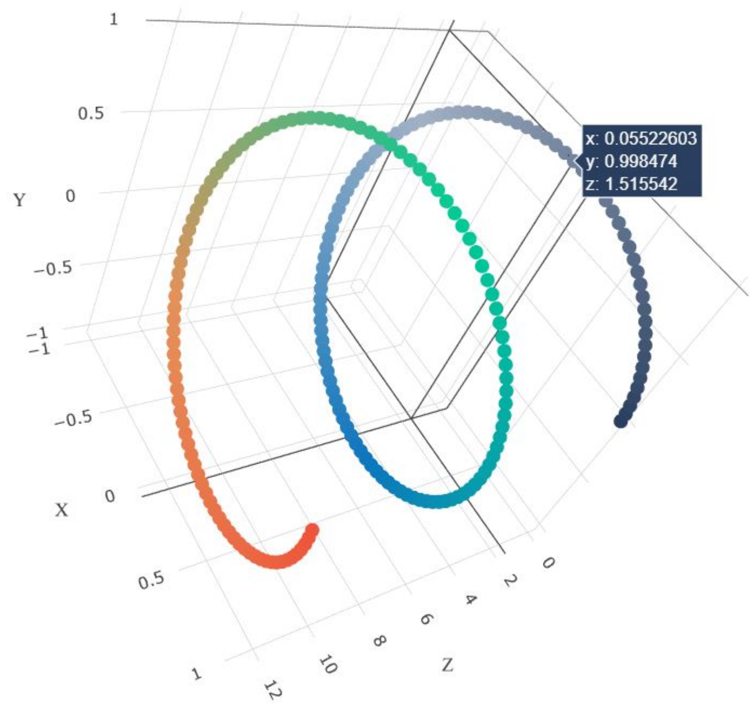
    - Set 1: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

2. **The hyper paramter tuning (Consider any two hyper parameters preferably n_estimators, max_depth)**

    - Consider the following range for hyperparameters **n_estimators** = [10, 50, 100, 150, 200, 300, 500, 1000], **max_depth** = [2, 3, 4, 5, 6, 7, 8, 9, 10]
    - 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
    - find the best hyper paramter using k-fold cross validation/simple cross validation data
    - use gridsearch cv or randomsearch cv or you can write your own for loops to do this task

3. **Representation of results**

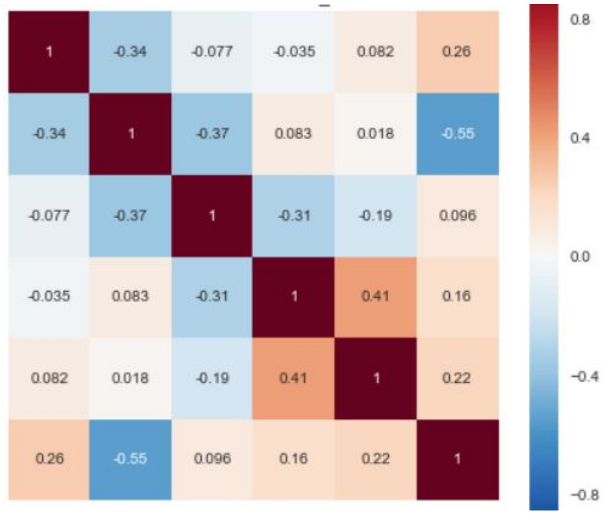
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



with X-axis as **n_estimators**, Y-axis as **max_depth**, and Z-axis as **AUC Score** , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

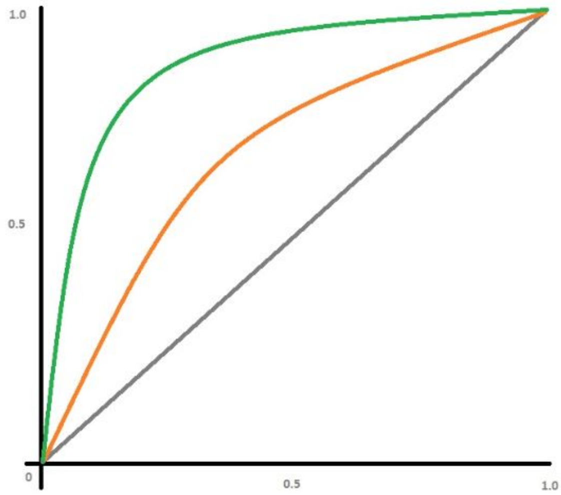
or

- 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

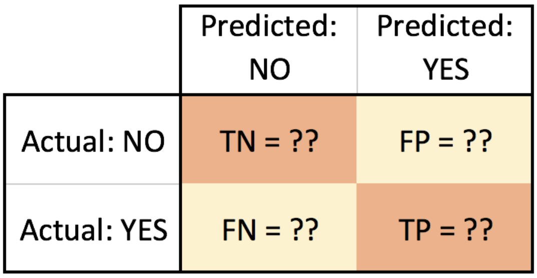


[seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**

- You can choose either of the plotting techniques: 3d plot or heat map
- 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.



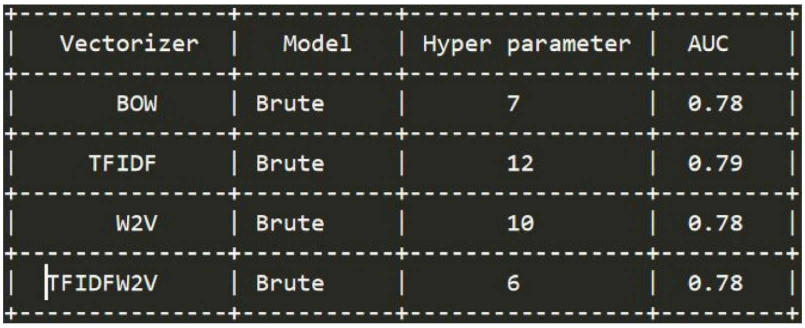
- Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points



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

4. **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  |
+--------------+-----------+-----------------+--------+
```

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. Random Forest and GBDT

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 subsectio
# 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 cod
# 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]:

```
y = project_data['project_is_approved'].values
x = project_data.drop(['project_is_approved'], axis=1)
```

In [0]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, stratify=y)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# 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 cod
# 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
```

In [48]:

```
#Encoding numerical features

#Price
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

normalizer.fit(x_train['price'].values.reshape(1, -1))
x_train_price = normalizer.transform(x_train['price'].values.reshape(1, -1))
x_test_price = normalizer.transform(x_test['price'].values.reshape(1, -1))

print(x_train_price.shape, y_train.shape)
print(x_test_price.shape, y_test.shape)
```

```
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [49]:

```
#Teacher_number_of_previously_posted_projects

normalizer.fit(x_train['teacher_number_of_previously_posted_projects'].values.reshape(1, -1
x_train_previous_projects = normalizer.transform(x_train['teacher_number_of_previously_post
x_test_previous_projects = normalizer.transform(x_test['teacher_number_of_previously_postec

print(x_train_previous_projects.shape, y_train.shape)
print(x_test_previous_projects.shape, y_test.shape)
```

```
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [50]:

```
#Quantity

normalizer.fit(x_train['quantity'].values.reshape(1, -1))
x_train_quantity = normalizer.transform(x_train['quantity'].values.reshape(1, -1))
x_test_quantity = normalizer.transform(x_test['quantity'].values.reshape(1, -1))

print(x_train_quantity.shape, y_train.shape)
print(x_test_quantity.shape, y_test.shape)
```

```
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [0]:

```
#Response Coding For Train Data

#I got this code from personalized cancer diagnosis case study


def get_rs_train(feature, df):
    value_count = df[feature].value_counts()

    rs = dict()
    for i, denominator in value_count.items():
        vec = []
        for j in range(0, 2):
            class_counts  = df.loc[(y_train==j) & (df[feature]==i)]
            vec.append(class_counts.shape[0]/denominator)
        rs[i]=vec
    return rs

def get_cate_train(feature, df):

    gv_dict = get_rs_train(feature, df)

    value_count = df[feature].value_counts()

    cat_fea = []

    for i , row in df.iterrows():
        if row[feature] in dict(value_count).keys():
            cat_fea.append(gv_dict[row[feature]])
        else:
            cat_fea.append([0.5, 0.5])

    return cat_fea
```

In [0]:

```
#Response coding for test data

#I got this code from personalized cancer diagnosis case study

def get_rs_test(feature, df):
    value_count = df[feature].value_counts()

    rs = dict()
    for i, denominator in value_count.items():
        vec = []
        for j in range(0, 2):
            class_counts  = df.loc[(y_test==j) & (df[feature]==i)]
            vec.append(class_counts.shape[0]/denominator)
        rs[i]=vec
    return rs

def get_cate_test(feature, df):

    gv_dict = get_rs_test(feature, df)

    value_count = df[feature].value_counts()

    cat_fea = []

    for i , row in df.iterrows():
        if row[feature] in dict(value_count).keys():
            cat_fea.append(gv_dict[row[feature]])
        else:
            cat_fea.append([0.5, 0.5])

    return cat_fea
```

In [53]:

```
#School_state

x_train_state = np.array(get_cate_train('school_state', x_train))
x_test_state = np.array(get_cate_test('school_state', x_test))

print(x_test_state.shape)
print(x_train_state.shape)
```

```
(16500, 2)
(33500, 2)
```

In [54]:

```
#Teacher_prefix

x_train_teacher = np.array(get_cate_train('teacher_prefix', x_train))
x_test_teacher = np.array(get_cate_test('teacher_prefix', x_test))

print(x_train_teacher.shape)
print(x_test_teacher.shape)
```

```
(33500, 2)
(16500, 2)
```

In [55]:

```
#project_grade_category

x_train_grade = np.array(get_cate_train('project_grade_category', x_train))
x_test_grade = np.array(get_cate_test('project_grade_category', x_test))

print(x_train_grade.shape)
print(x_test_grade.shape)
```

```
(33500, 2)
(16500, 2)
```

In [56]:

```
#project_subject_categories

x_train_categories = np.array(get_cate_train('clean_categories', x_train))
x_test_categories = np.array(get_cate_test('clean_categories', x_test))

print(x_train_categories.shape)
print(x_test_categories.shape)
```

```
(33500, 2)
(16500, 2)
```

In [57]:

```
#project_subject_subcategories

x_train_subcategories = np.array(get_cate_train('clean_subcategories', x_train))
x_test_subcategories = np.array(get_cate_test('clean_subcategories', x_test))

print(x_train_subcategories.shape)
print(x_test_subcategories.shape)
```

```
(33500, 2)
(16500, 2)
```

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 subsectic
# 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 coa
# make sure you featurize train and test data separtly

# 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
```

BOW: Project_title, essay

In [58]:

```
vectorizer = CountVectorizer(min_df=10, ngram_range=(2, 2), max_features=5000)
vectorizer.fit(x_train['project_title'].values)

x_train_title_bow = vectorizer.transform(x_train['project_title'].values)
x_test_title_bow = vectorizer.transform(x_test['project_title'].values)

print(x_train_title_bow.shape)
print(x_test_title_bow.shape)
```

```
(33500, 787)
(16500, 787)
```

In [59]:

```
vectorizer.fit(project_data['essay'].values)

x_train_essay_bow = vectorizer.transform(x_train['essay'].values)
x_test_essay_bow = vectorizer.transform(x_test['essay'].values)

print(x_train_essay_bow.shape)
print(x_test_essay_bow.shape)
```

```
(33500, 5000)
(16500, 5000)
```

TFIDF: Project_title, essay

In [60]:

```
vectorizer = TfidfVectorizer(min_df=10, ngram_range=(2, 2), max_features=5000)
vectorizer.fit(x_train['project_title'].values)

x_train_title_tfidf = vectorizer.transform(x_train['project_title'].values)
x_test_title_tfidf = vectorizer.transform(x_test['project_title'].values)

print(x_train_title_tfidf.shape)
print(x_test_title_tfidf.shape)
```

```
(33500, 787)
(16500, 787)
```

In [61]:

```
vectorizer.fit(x_train['essay'].values)

x_train_essay_tfidf = vectorizer.transform(x_train['essay'].values)
x_test_essay_tfidf = vectorizer.transform(x_test['essay'].values)

print(x_train_essay_tfidf.shape)
print(x_test_essay_tfidf.shape)
```

```
(33500, 5000)
(16500, 5000)
```

AVG W2V: project_title, essay

In [0]:

```
with open('/content/drive/My Drive/Colab Notebooks/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

Project_title

In [63]:

```
x_train_title_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_train['project_title'].values): # 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
    x_train_title_avg_w2v.append(vector)

print(len(x_train_title_avg_w2v))
print(len(x_train_title_avg_w2v[0]))
```

```
100%|██████████| 33500/33500 [00:00<00:00, 71306.18it/s]

33500
300
```

In [64]:

```
x_test_title_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_test['project_title'].values): # 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
    x_test_title_avg_w2v.append(vector)

print(len(x_test_title_avg_w2v))
print(len(x_test_title_avg_w2v[0]))
```

```
100%|██████████| 16500/16500 [00:00<00:00, 67510.42it/s]

16500
300
```

Essay

In [65]:

```
# average Word2Vec
# compute average word2vec for each review.
x_train_essay_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_train['essay'].values): # 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
    x_train_essay_avg_w2v.append(vector)

print(len(x_train_essay_avg_w2v))
print(len(x_train_essay_avg_w2v[0]))
```

```
100%|██████████| 33500/33500 [00:09<00:00, 3645.20it/s]

33500
300
```

In [66]:

```
x_test_essay_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_test['essay'].values): # 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
    x_test_essay_avg_w2v.append(vector)
```

```
100%|██████████| 16500/16500 [00:04<00:00, 3639.74it/s]
```

TFIDF W2V: Project_title, essay

Project_title

In [0]:

```
#project_title

tfidf_model = TfidfVectorizer()
tfidf_model.fit(x_train['project_title'])
# 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 [68]:

```
x_train_title_tfidf_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_train['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    x_train_title_tfidf_w2v.append(vector)

print(len(x_train_title_tfidf_w2v))
```

```
100%|██████████| 33500/33500 [00:01<00:00, 30965.55it/s]

33500
```

In [69]:

```
x_test_title_tfidf_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_test['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    x_test_title_tfidf_w2v.append(vector)
```

```
100%|██████████| 16500/16500 [00:00<00:00, 33333.52it/s]
```

Essay

In [0]:

```
#essay

tfidf_model = TfidfVectorizer()
tfidf_model.fit(x_train['essay'])
# 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 [71]:

```
x_train_essay_tfidf_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_train['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    x_train_essay_tfidf_w2v.append(vector)

print(len(x_train_essay_tfidf_w2v))
```

```
100%|██████████| 33500/33500 [00:57<00:00, 584.44it/s]

33500
```

In [72]:

```
x_test_essay_tfidf_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_test['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    x_test_essay_tfidf_w2v.append(vector)
```

```
100%|██████████| 16500/16500 [00:28<00:00, 582.05it/s]
```

2.4 Applying Random Forest

Apply Random Forest 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 [0]:

```
#Before merging, re-shape some features. if we dont reshape, we'll get error
#Re-shaping
x_train_price = x_train_price.reshape(-1,1)
x_train_previous_projects = x_train_previous_projects.reshape(-1,1)
x_train_quantity = x_train_quantity.reshape(-1, 1)

x_test_price = x_test_price.reshape(-1,1)
x_test_previous_projects = x_test_previous_projects.reshape(-1,1)
x_test_quantity = x_test_quantity.reshape(-1, 1)
```

2.4.1 Applying Random Forests on BOW, SET 1

In [0]:

```
# Please write all the code with proper documentation
```

In [0]:

```
#Merging Features

from scipy.sparse import hstack

x_train_bow = hstack((x_train_price, x_train_previous_projects, x_train_quantity, x_train_s
                     x_train_categories, x_train_subcategories, x_train_essay_bow, x_train_

x_test_bow = hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_test_state,
                     x_test_categories, x_test_subcategories, x_test_essay_bow, x_test_titl
```

In [0]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

tuned_parameters = {'n_estimators':[10, 50, 100, 200, 500], 'max_depth':[2, 4, 5, 6, 8]}

RF = RandomForestClassifier(n_jobs=-1, class_weight='balanced')
clf = GridSearchCV(RF, tuned_parameters, cv=5, scoring='roc_auc', return_train_score=True)

clf.fit(x_train_bow, y_train)
```

Out[133]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True,
                                              class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=-
1,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [2, 4, 5, 6, 8],
                         'n_estimators': [10, 50, 100, 200, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [0]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])



#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [74]:

```
%matplotlib inline
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [0]:

```
def enable_plotly_in_cell():
    import IPython
    from plotly.offline import init_notebook_mode
    display(IPython.core.display.HTML('''<script src="/static/components/requirejs/require.
    init_notebook_mode(connected=False)
```

In [0]:

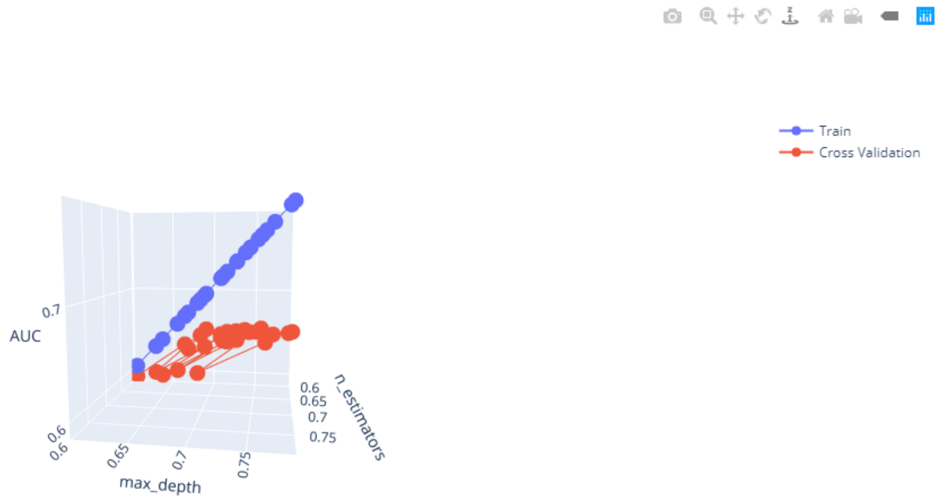
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [0]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```



In [0]:

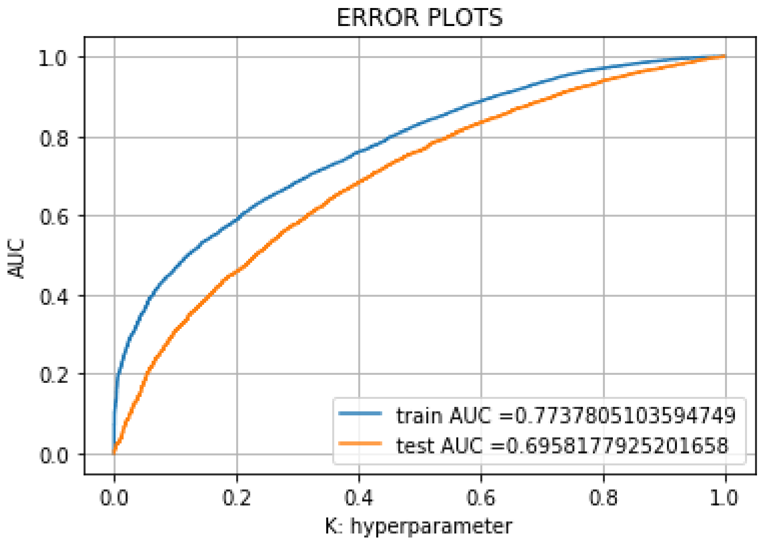
```
clf.best_params_
```

Out[84]:

```
{'max_depth': 8, 'n_estimators': 500}
```

In [0]:

```
from sklearn.metrics import roc_curve, auc

RF = RandomForestClassifier(max_depth=8, n_estimators=500, class_weight='balanced')
RF.fit(x_train_bow, y_train)

y_train_pred = clf.predict_proba(x_train_bow)[:, 1]
y_test_pred = clf.predict_proba(x_test_bow)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.grid()
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("ERROR PLOTS")

plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [0]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

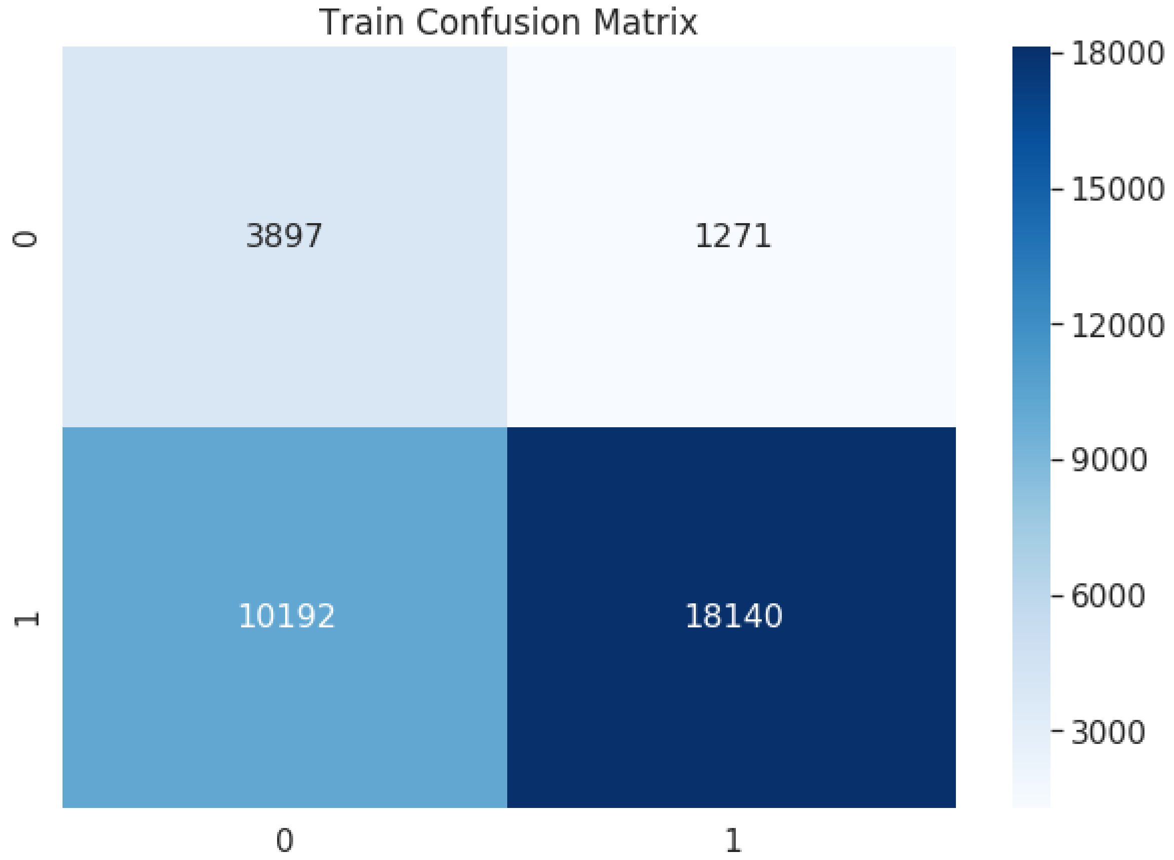
```
the maximum value of tpr*(1-fpr) 0.4828007659293237 for threshold 0.505
Train confusion matrix
[[ 3897  1271]
 [10192 18140]]
Test confusion matrix
[[1779  767]
 [5845 8109]]
```

In [0]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c53549160>
```

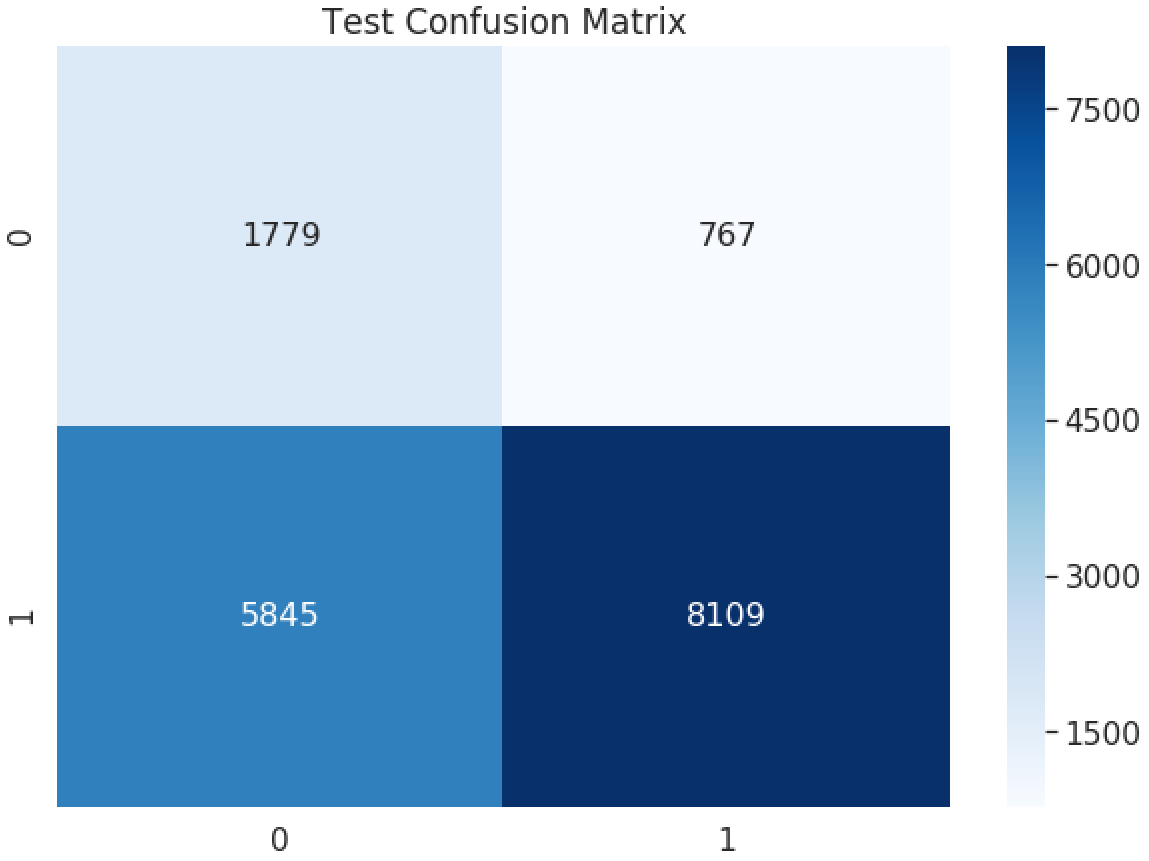


In [0]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c5447d668>
```



2.4.2 Applying Random Forests on TFIDF, SET 2

In [0]:

```
# Please write all the code with proper documentation
```

In [0]:

```
#Merging Features

x_train_tfidf = hstack((x_train_price, x_train_previous_projects, x_train_quantity, x_trair
                     x_train_categories, x_train_subcategories, x_train_essay_tfidf, x_trai

x_test_tfidf = hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_test_stat
                     x_test_categories, x_test_subcategories, x_test_essay_tfidf, x_test_ti
```

In [0]:

```
tuned_parameters = {'n_estimators':[10, 50, 100, 200, 500], 'max_depth':[2, 4, 5, 6, 8]}

RF  = RandomForestClassifier(n_jobs=-1, class_weight='balanced')
clf = GridSearchCV(RF, tuned_parameters, cv=5, scoring='roc_auc', return_train_score=True)

clf.fit(x_train_tfidf, y_train)
```

Out[91]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True,
                                              class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=-
1,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [2, 4, 5, 6, 8],
                         'n_estimators': [10, 50, 100, 200, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [0]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])


#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [0]:

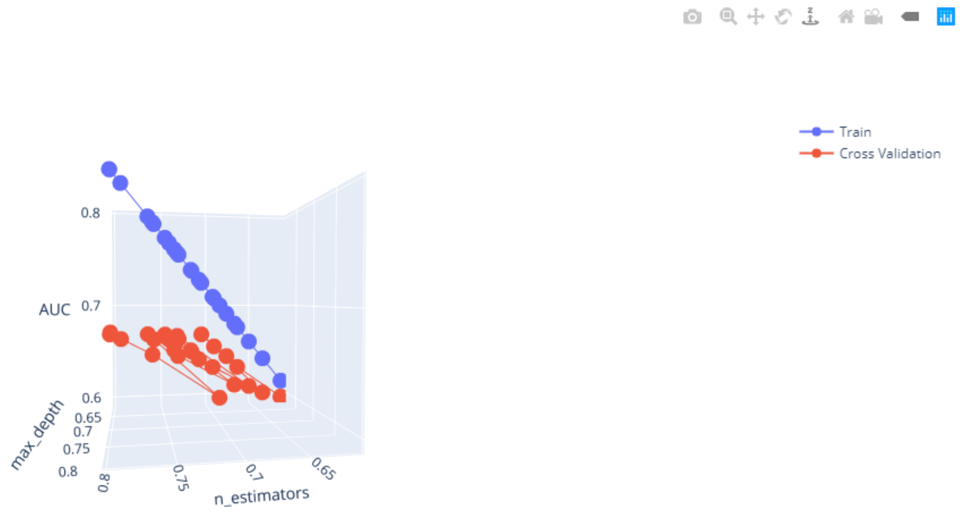
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [0]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```



In [0]:

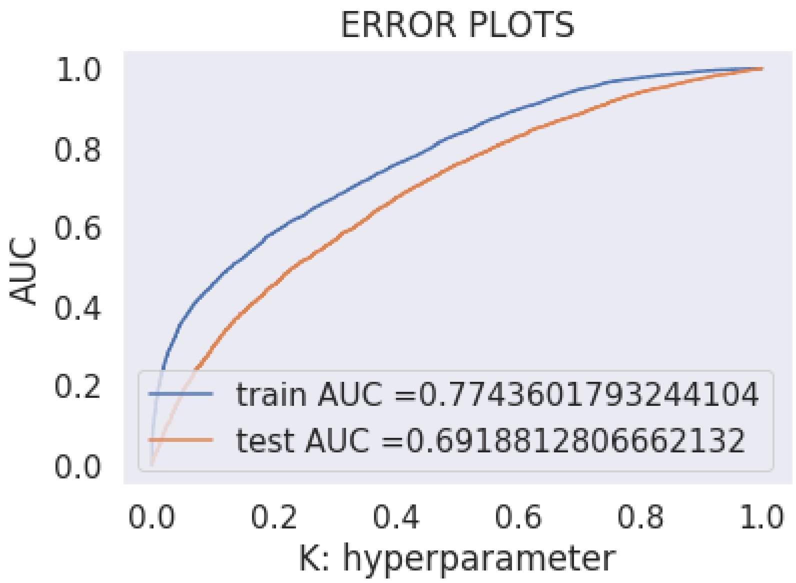
```
clf.best_params_
```

Out[95]:

```
{'max_depth': 8, 'n_estimators': 500}
```

In [0]:

```
from sklearn.metrics import roc_curve, auc

RF = RandomForestClassifier(max_depth=8, n_estimators=500, class_weight='balanced')
RF.fit(x_train_tfidf, y_train)

y_train_pred = clf.predict_proba(x_train_tfidf)[:, 1]
y_test_pred = clf.predict_proba(x_test_tfidf)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)


plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.grid()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")

plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [0]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

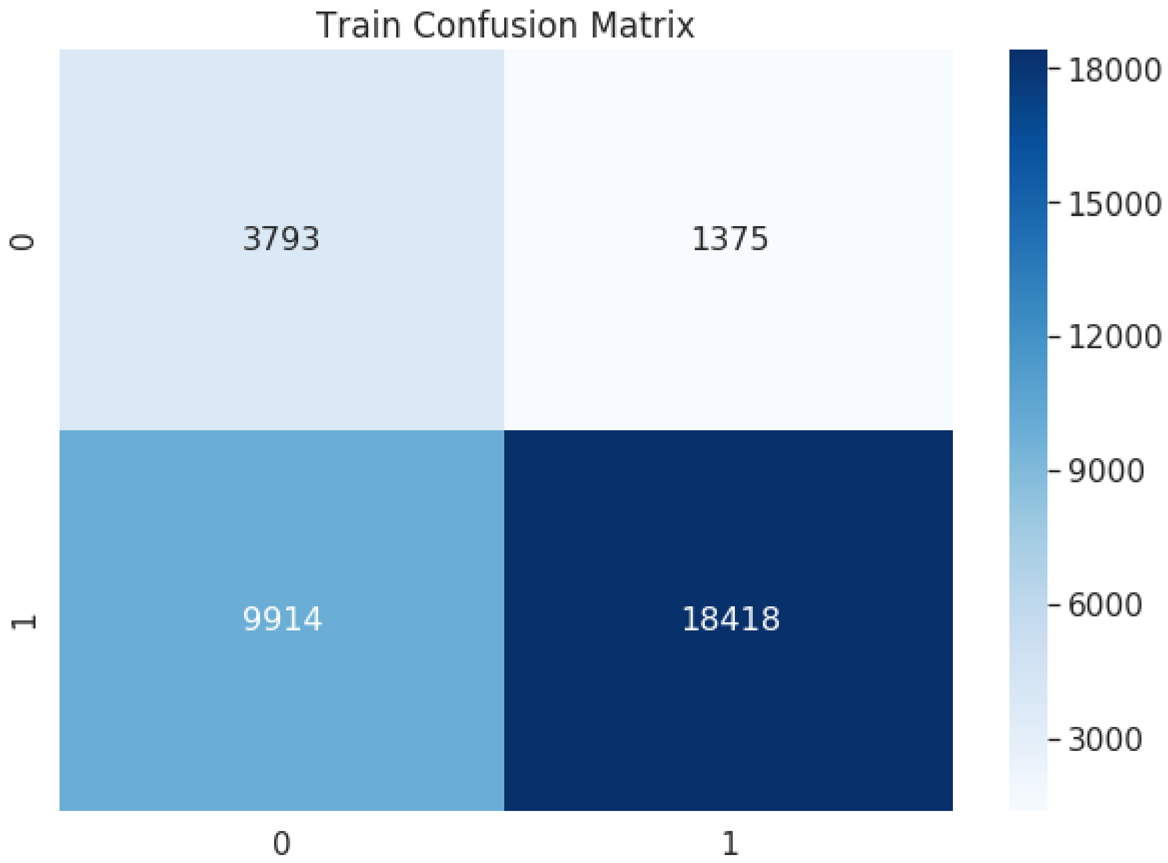
```
the maximum value of tpr*(1-fpr) 0.47711774944936397 for threshold 0.506
Train confusion matrix
[[ 3793  1375]
 [ 9914 18418]]
Test confusion matrix
[[1711  835]
 [5637 8317]]
```

In [0]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c5dad5cf8>
```

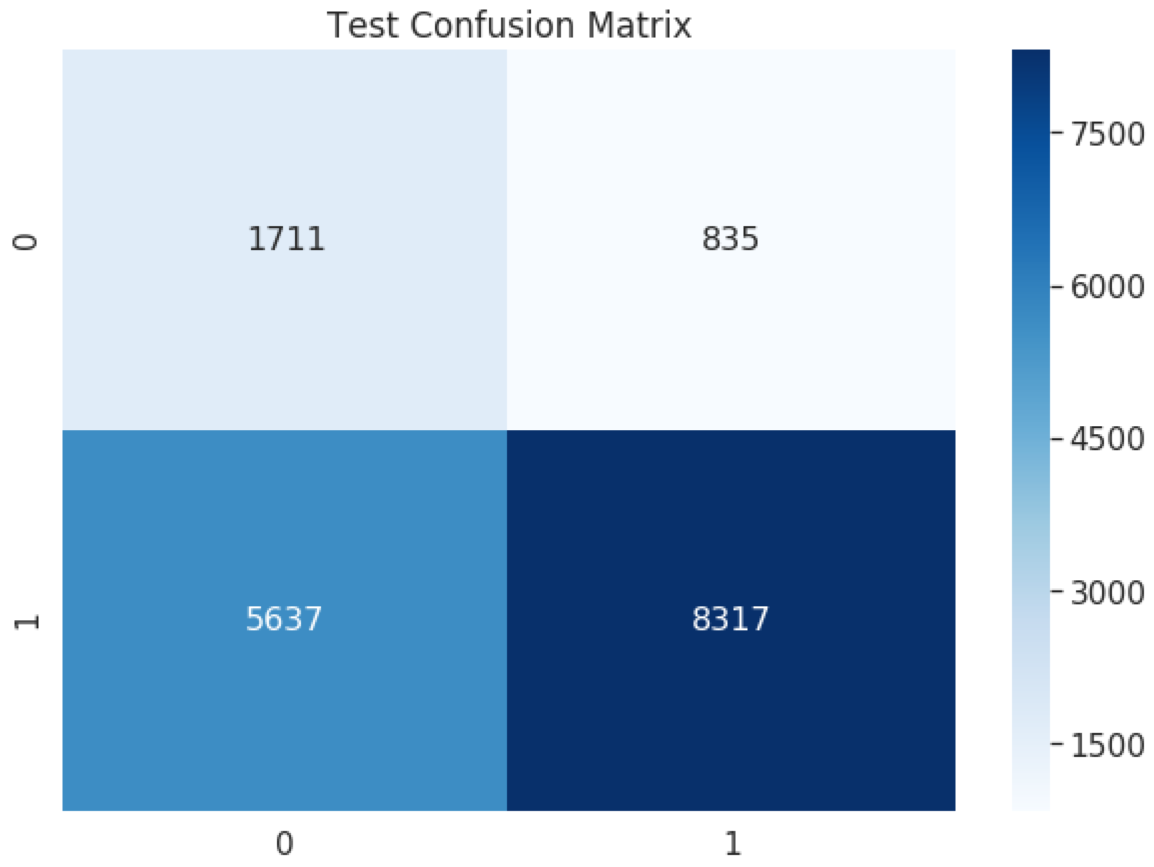


In [0]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[100]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c53504748>
```



2.4.3 Applying Random Forests on AVG W2V, SET 3

In [0]:

```
# Please write all the code with proper documentation
```

In [0]:

```
#Merging Features

x_train_avg_w2v = np.hstack((x_train_price, x_train_previous_projects, x_train_quantity, x_
                     x_train_categories, x_train_subcategories, x_train_essay_avg_w2v, x_tr

x_test_avg_w2v = np.hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_test
                     x_test_categories, x_test_subcategories, x_test_essay_avg_w2v, x_test_
```

In [0]:

```
#Hyperparameter Tuning
tuned_parameters = {'n_estimators':[10, 50, 100, 200, 500], 'max_depth':[2, 4, 5, 6, 8]}

RF = RandomForestClassifier(n_jobs=-1, class_weight='balanced')
clf = GridSearchCV(RF, tuned_parameters, cv=5, scoring='roc_auc', return_train_score=True)

clf.fit(x_train_avg_w2v, y_train)
```

Out[107]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True,
                                              class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=-
1,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [2, 4, 5, 6, 8],
                         'n_estimators': [10, 50, 100, 200, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [0]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])


#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [0]:

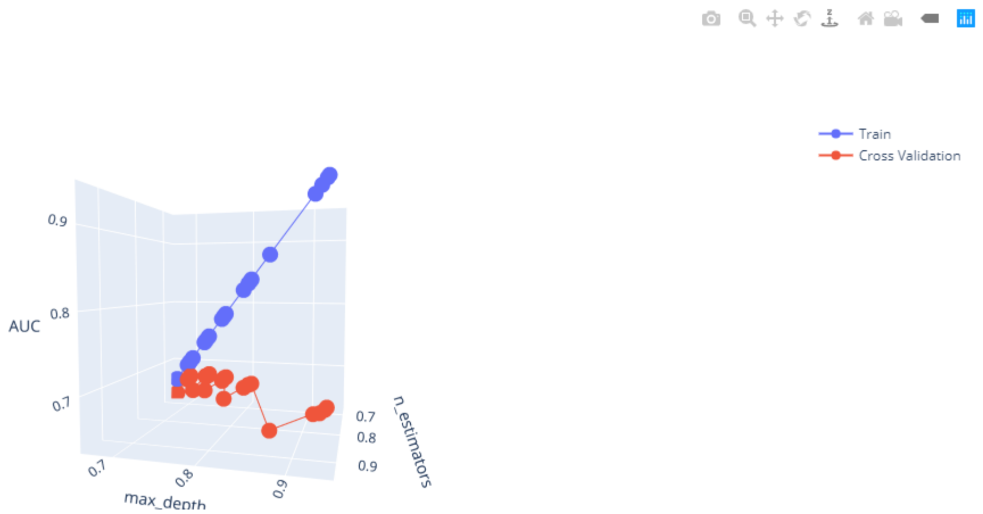
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [0]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```



In [0]:

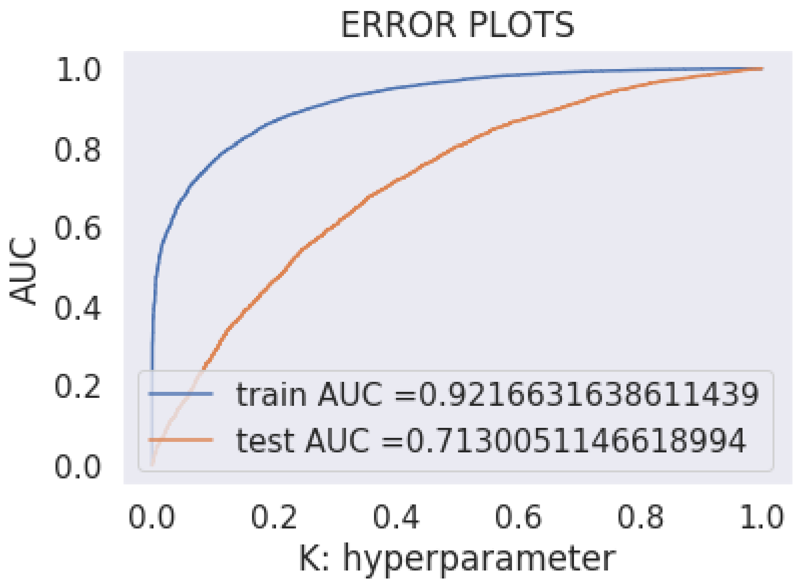
```
clf.best_params_
```

Out[115]:

```
{'max_depth': 8, 'n_estimators': 500}
```

In [0]:

```
from sklearn.metrics import roc_curve, auc

RF = RandomForestClassifier(max_depth=8, n_estimators=500, class_weight='balanced')
RF.fit(x_train_avg_w2v, y_train)

y_train_pred = clf.predict_proba(x_train_avg_w2v)[:, 1]
y_test_pred = clf.predict_proba(x_test_avg_w2v)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.grid()
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("ERROR PLOTS")

plt.show()
```



In [0]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

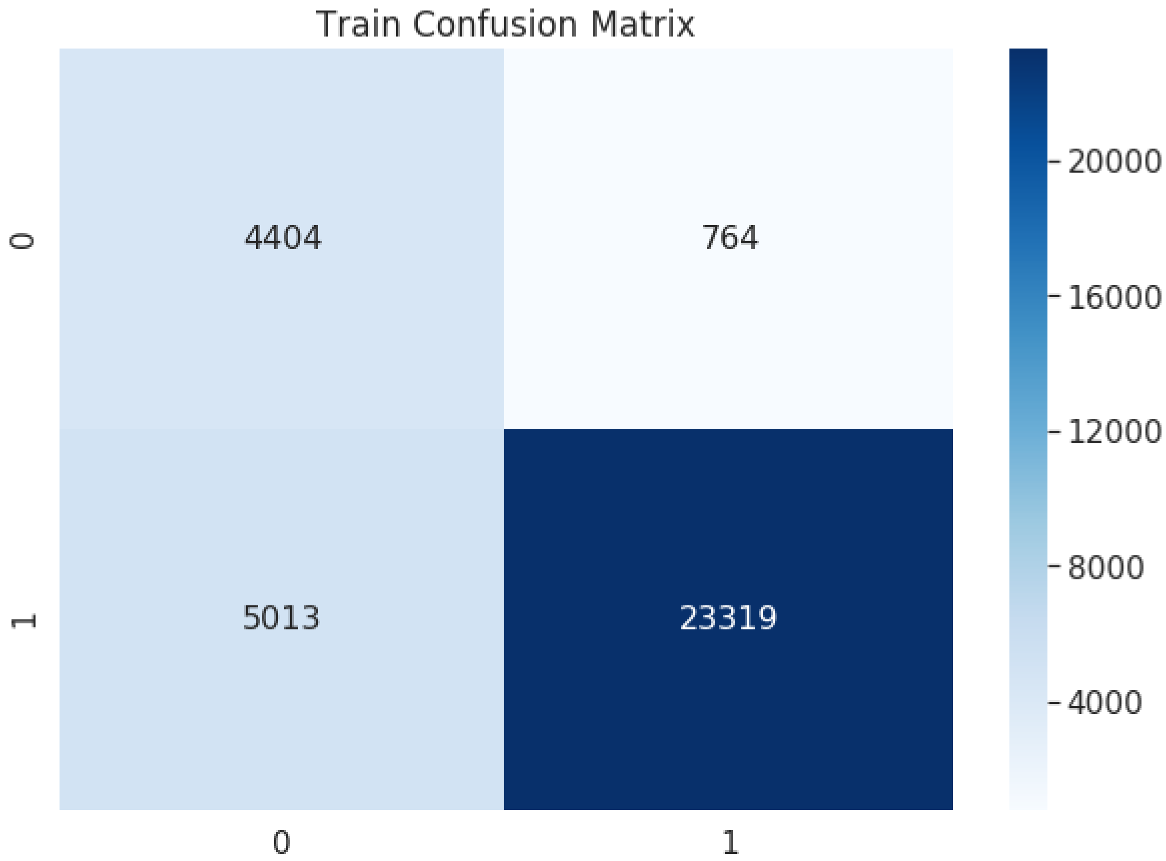
```
the maximum value of tpr*(1-fpr) 0.7013866487543322 for threshold 0.525
Train confusion matrix
[[ 4404   764]
 [ 5013 23319]]
Test confusion matrix
[[ 1416  1130]
 [ 3397 10557]]
```

In [0]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c4dd44550>
```

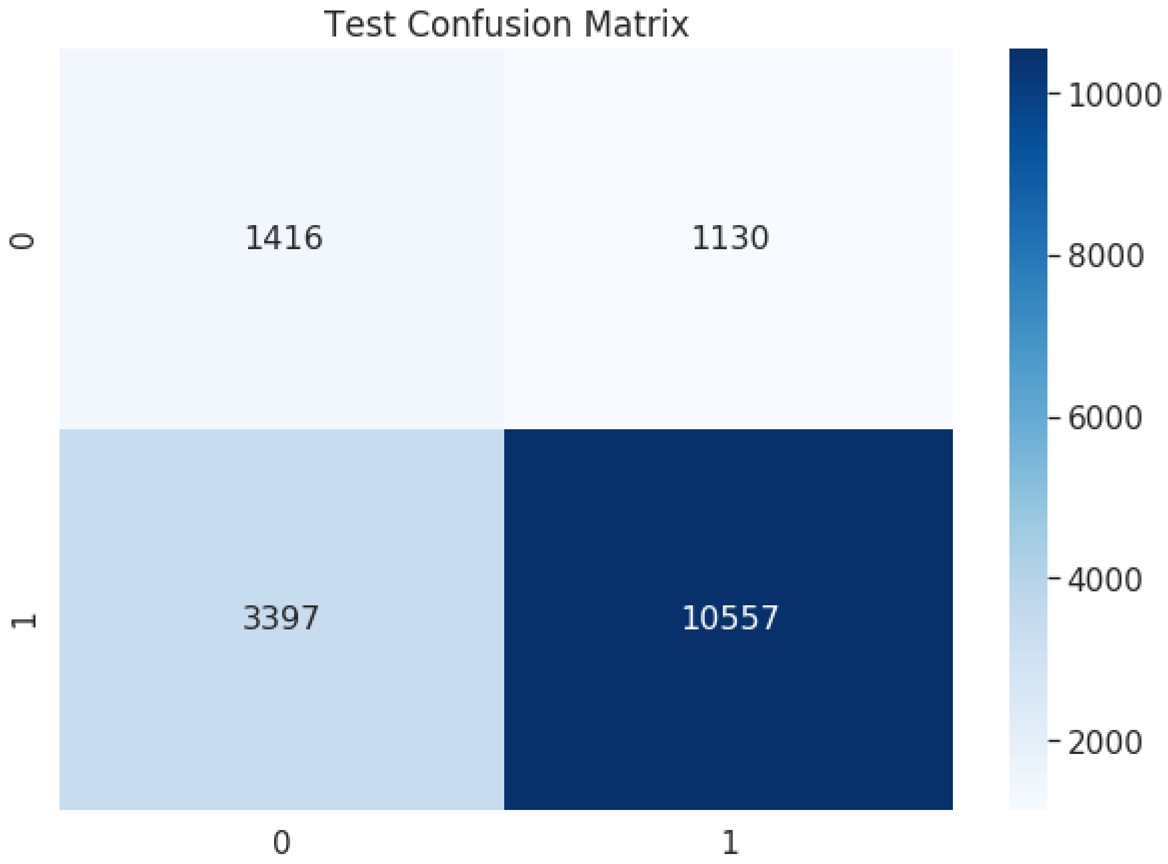


In [0]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[121]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c50ad9c88>
```



In [0]:

```

```

2.4.4 Applying Random Forests on TFIDF W2V, SET 4

In [0]:

```
# Please write all the code with proper documentation
```

In [0]:

```
#Merging Features

x_train_tfidf_w2v = np.hstack((x_train_price, x_train_previous_projects, x_train_quantity, 
                     x_train_categories, x_train_subcategories, x_train_essay_tfidf_w2v, x_

x_test_tfidf_w2v = np.hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_te
                     x_test_categories, x_test_subcategories, x_test_essay_tfidf_w2v, x_tes
```

In [0]:

```
#Hyperparameter Tuning
tuned_parameters = {'n_estimators':[10, 50, 100, 200, 500], 'max_depth':[2, 4, 5, 6, 8]}

RF = RandomForestClassifier(n_jobs=-1, class_weight='balanced')
clf = GridSearchCV(RF, tuned_parameters, cv=5, scoring='roc_auc', return_train_score=True)

clf.fit(x_train_tfidf_w2v, y_train)
```

Out[123]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True,
                                              class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=-
1,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [2, 4, 5, 6, 8],
                         'n_estimators': [10, 50, 100, 200, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [0]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])


#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [0]:

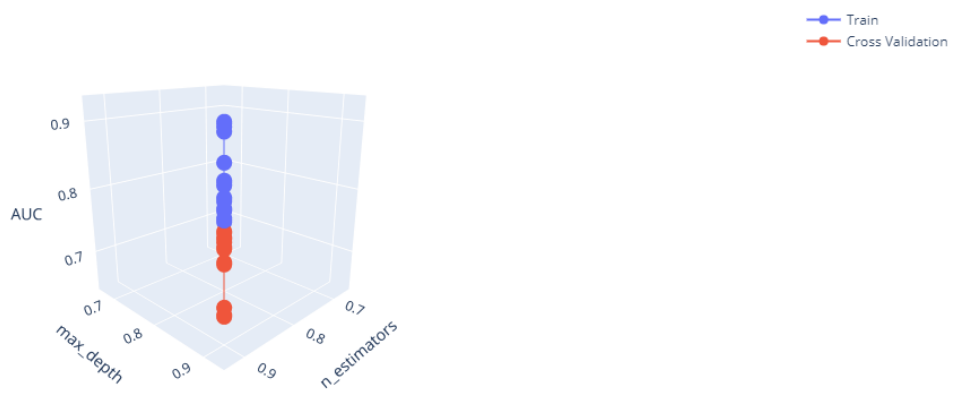
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [0]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```



In [0]:

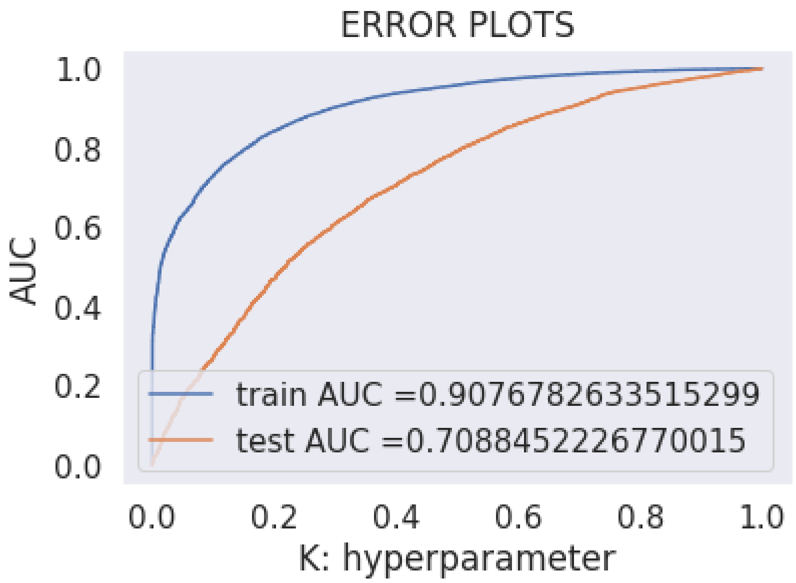
```
clf.best_params_
```

Out[127]:

```
{'max_depth': 8, 'n_estimators': 500}
```

In [0]:

```
from sklearn.metrics import roc_curve, auc

RF = RandomForestClassifier(max_depth=8, n_estimators=500, class_weight='balanced')
RF.fit(x_train_tfidf_w2v, y_train)

y_train_pred = clf.predict_proba(x_train_tfidf_w2v)[:, 1]
y_test_pred = clf.predict_proba(x_test_tfidf_w2v)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.grid()
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("ERROR PLOTS")

plt.show()
```



In [0]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

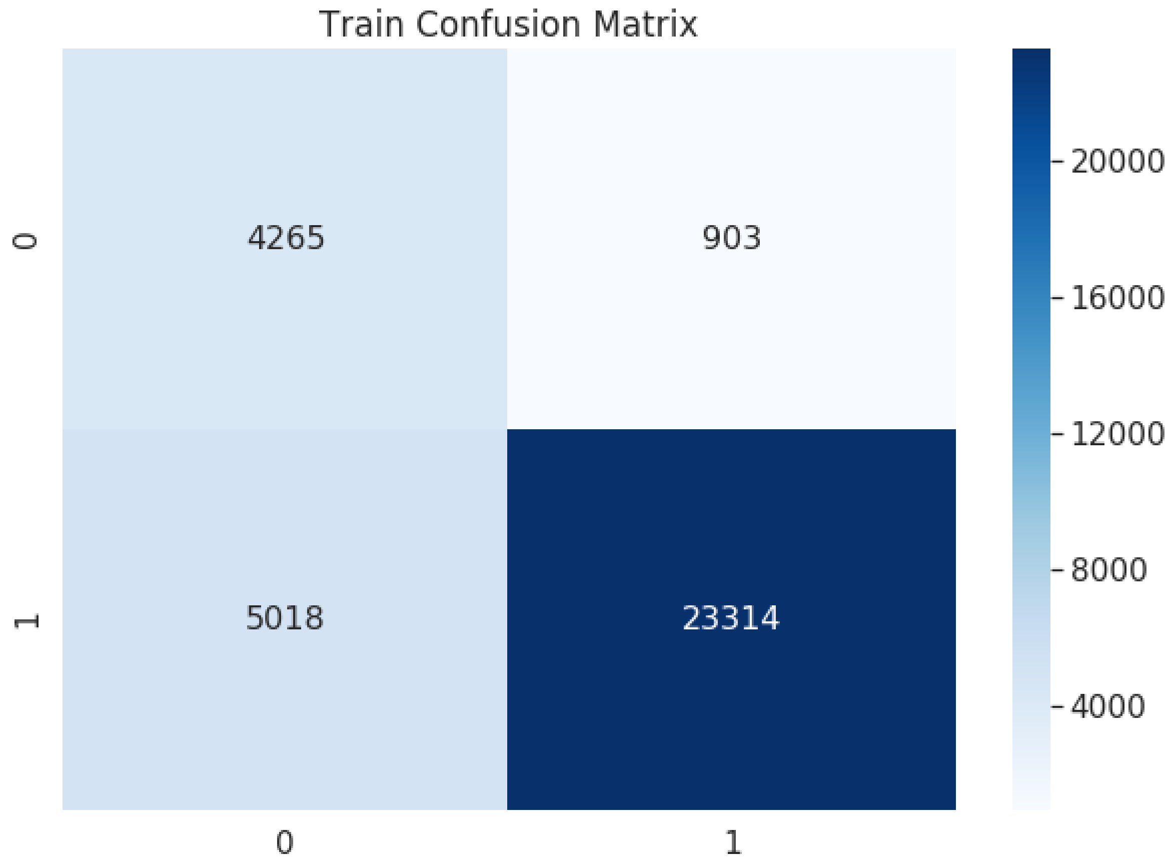
```
the maximum value of tpr*(1-fpr) 0.6791036888350384 for threshold 0.512
Train confusion matrix
[[ 4265   903]
 [ 5018 23314]]
Test confusion matrix
[[ 1355  1191]
 [ 3264 10690]]
```

In [0]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[131]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c530b8278>
```

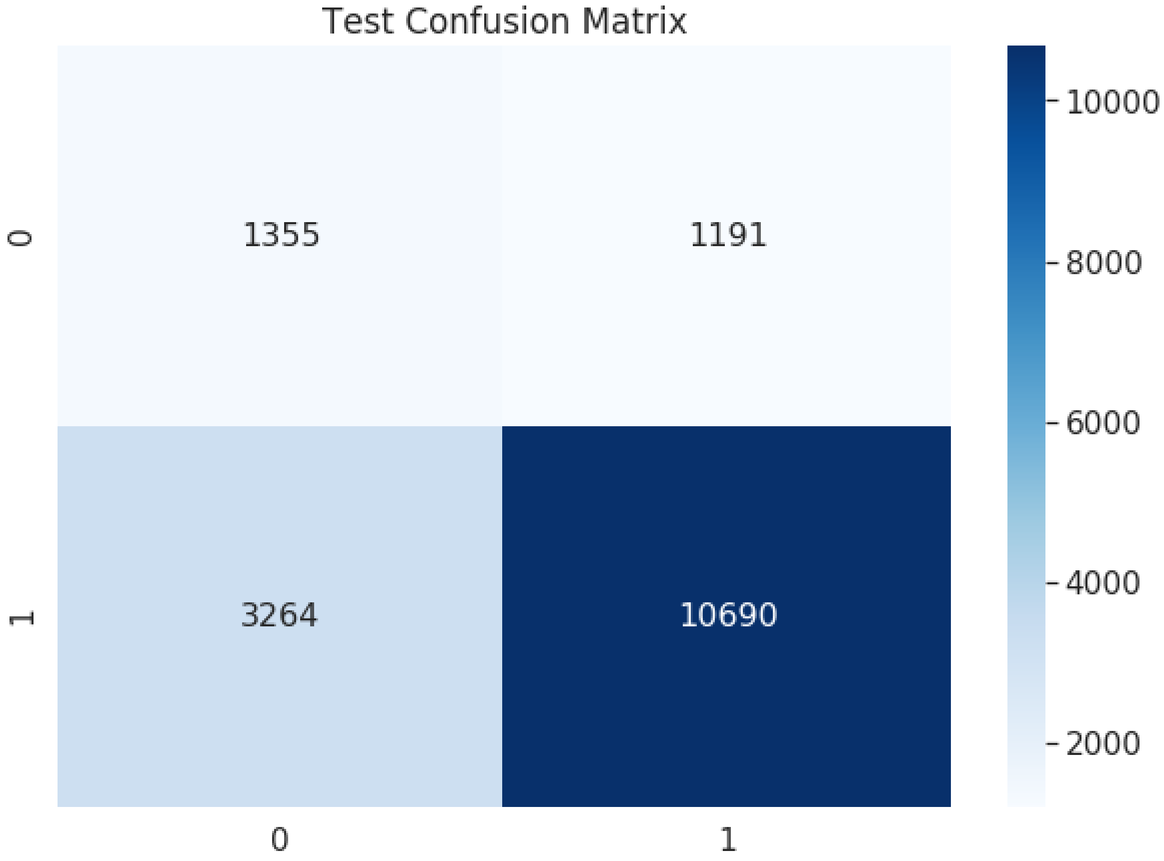


In [0]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[132]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c530b8550>
```



In [0]:

```

```

2.5 Applying GBDT

Apply GBDT 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

2.5.1 Applying XGBOOST on BOW, SET 1

In [0]:

```
# Please write all the code with proper documentation
```

In [0]:

```
import xgboost as xgb
```

In [0]:

```
#Hyperparameter Tuning

tuned_parameters = {'n_estimators':[10, 50, 100, 150, 300, 500], 'max_depth':[2, 4, 5, 6, 8
xgb_model = xgb.XGBClassifier()
clf = GridSearchCV(xgb_model, tuned_parameters, cv=5, scoring='roc_auc', return_train_score
clf.fit(x_train_bow, y_train)
```

Out[134]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                     colsample_bylevel=1, colsample_bynode=
1,
                                     colsample_bytree=1, gamma=0,
                                     learning_rate=0.1, max_delta_step=0,
                                     max_depth=3, min_child_weight=1,
                                     missing=None, n_estimators=100, n_jobs=
1,
                                     nthread=None, objective='binary:logisti
c',
                                     random_state=0, reg_alpha=0, reg_lambda
=1,
                                     scale_pos_weight=1, seed=None, silent=N
one,
                                     subsample=1, verbosity=1),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [2, 4, 5, 6, 8],
                         'n_estimators': [10, 50, 100, 150, 300, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [0]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])


#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [0]:

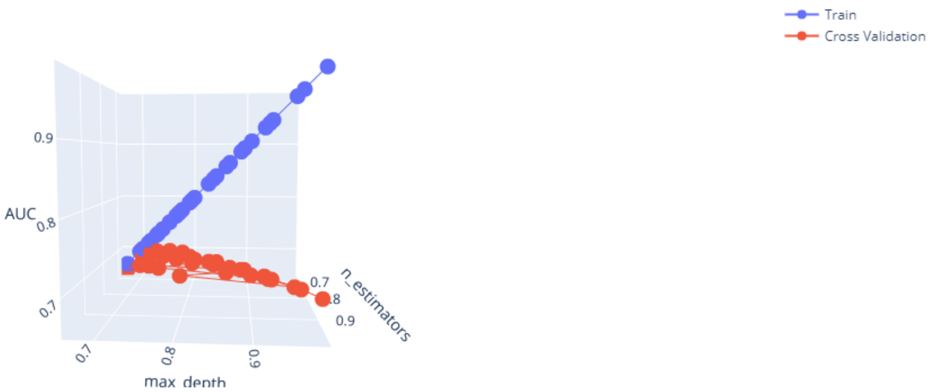
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [0]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```

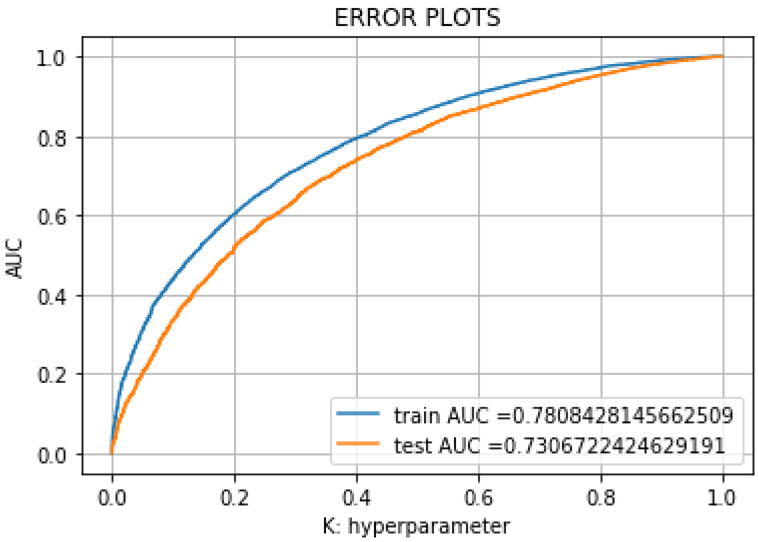


In [ ]:

```
clf.best_params_
```

In [0]:

```
from sklearn.metrics import roc_curve, auc

xgb_model = xgb.XGBClassifier(max_depth=2, n_estimators=500)
xgb_model.fit(x_train_bow, y_train)

y_train_pred = clf.predict_proba(x_train_bow)[:, 1]
y_test_pred = clf.predict_proba(x_test_bow)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.grid()
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("ERROR PLOTS")

plt.show()
```



In [0]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

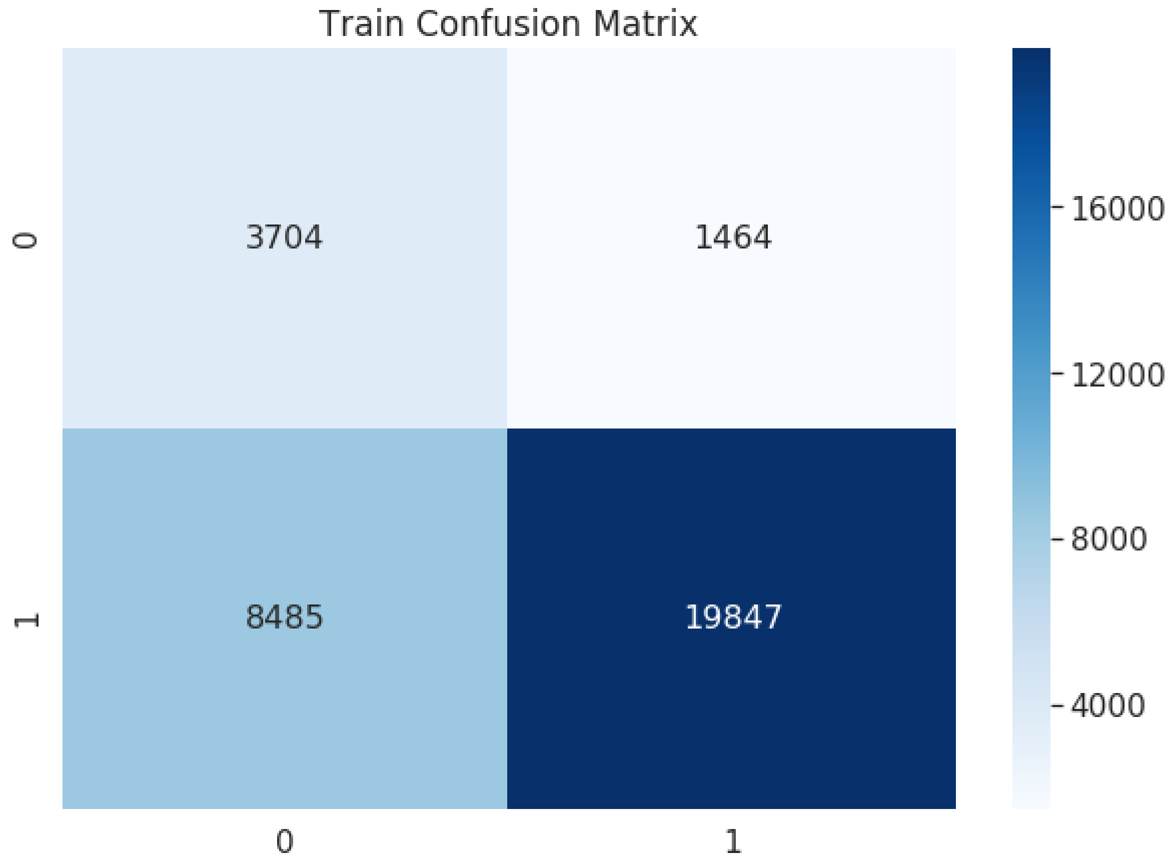
```
the maximum value of tpr*(1-fpr) 0.5020721244649358 for threshold 0.834
Train confusion matrix
[[ 3704  1464]
 [ 8485 19847]]
Test confusion matrix
[[1833  713]
 [5352 8602]]
```

In [0]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[146]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7faefec6c940>
```

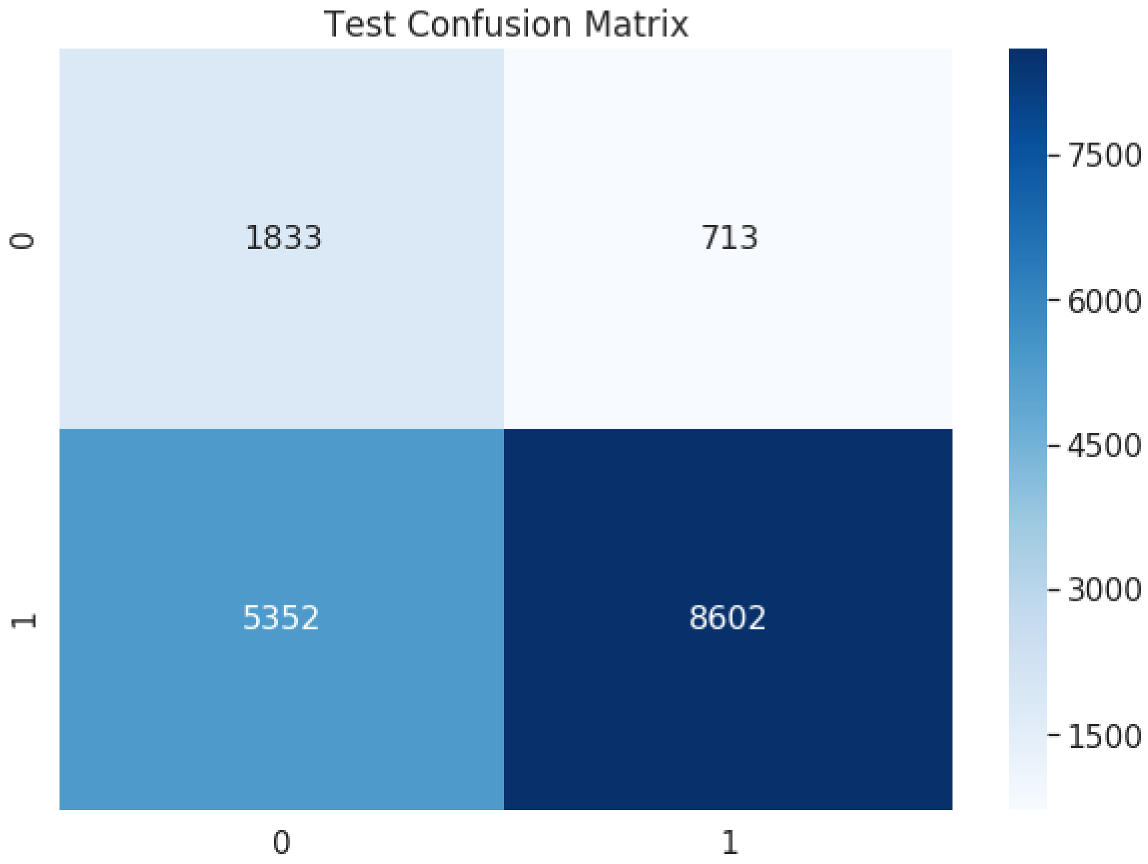


In [0]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[147]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7faefe20d4e0>
```



2.5.2 Applying XGBOOST on TFIDF, SET 2

In [0]:

```
# Please write all the code with proper documentation
```

In [0]:

```
#Hyperparameter Tuning
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV

tuned_parameters = {'n_estimators':[10, 50, 100, 150, 300, 500], 'max_depth':[2, 4, 5, 6, 8
xgb_model = xgb.XGBClassifier()
clf = RandomizedSearchCV(xgb_model, tuned_parameters, cv=5, scoring='roc_auc', return_train
clf.fit(x_train_tfidf, y_train)
```

Out[148]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                     colsample_bylevel=1, colsample_bynode=
1,
                                     colsample_bytree=1, gamma=0,
                                     learning_rate=0.1, max_delta_step=0,
                                     max_depth=3, min_child_weight=1,
                                     missing=None, n_estimators=100, n_jobs=
1,
                                     nthread=None, objective='binary:logisti
c',
                                     random_state=0, reg_alpha=0, reg_lambda
=1,
                                     scale_pos_weight=1, seed=None, silent=N
one,
                                     subsample=1, verbosity=1),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [2, 4, 5, 6, 8],
                         'n_estimators': [10, 50, 100, 150, 300, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [0]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])


#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [0]:

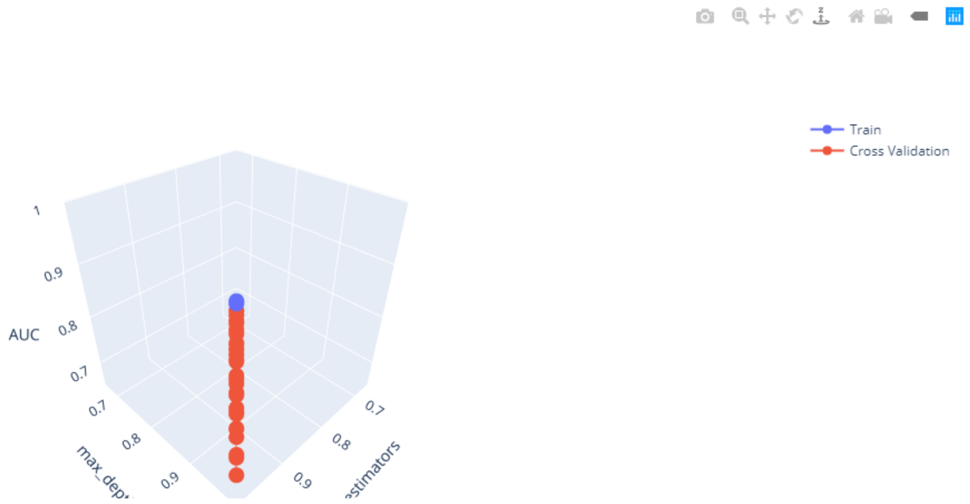
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [0]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```



In [0]:

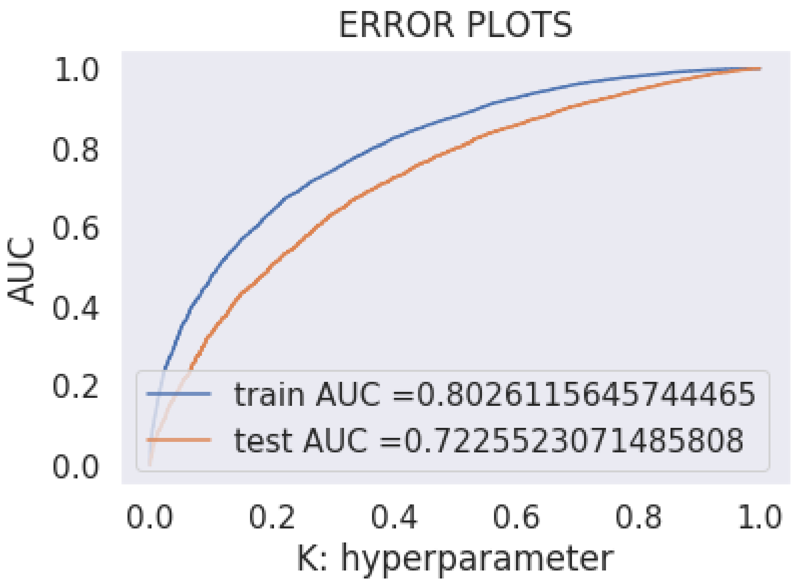
```
clf.best_params_
```

Out[152]:

```
{'max_depth': 2, 'n_estimators': 500}
```

In [0]:

```
from sklearn.metrics import roc_curve, auc

xgb_model = xgb.XGBClassifier(max_depth=2, n_estimators=500)
xgb_model.fit(x_train_tfidf, y_train)

y_train_pred = clf.predict_proba(x_train_tfidf)[:, 1]
y_test_pred = clf.predict_proba(x_test_tfidf)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.grid()
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("ERROR PLOTS")

plt.show()
```



In [0]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

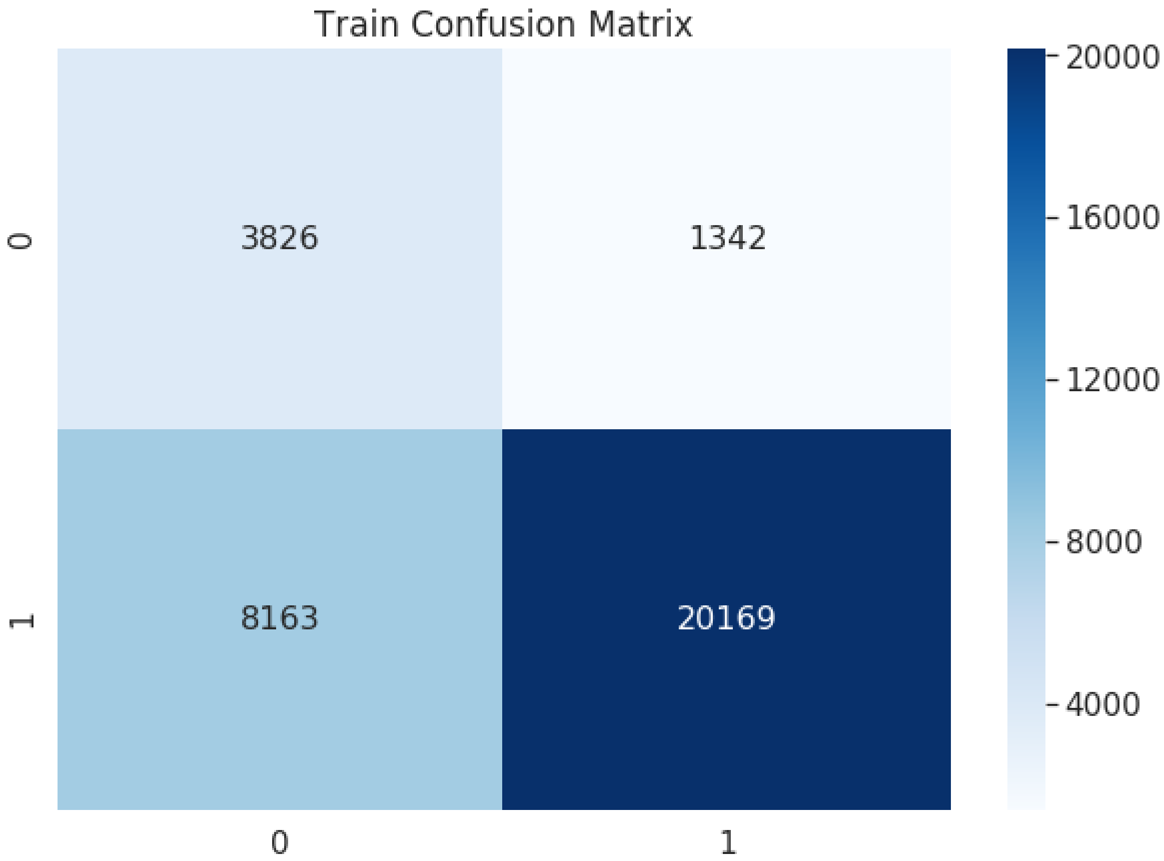
```
the maximum value of tpr*(1-fpr) 0.5270230300038159 for threshold 0.833
Train confusion matrix
[[ 3826  1342]
 [ 8163 20169]]
Test confusion matrix
[[1816  730]
 [5329 8625]]
```

In [0]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[157]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7faefdd00668>
```

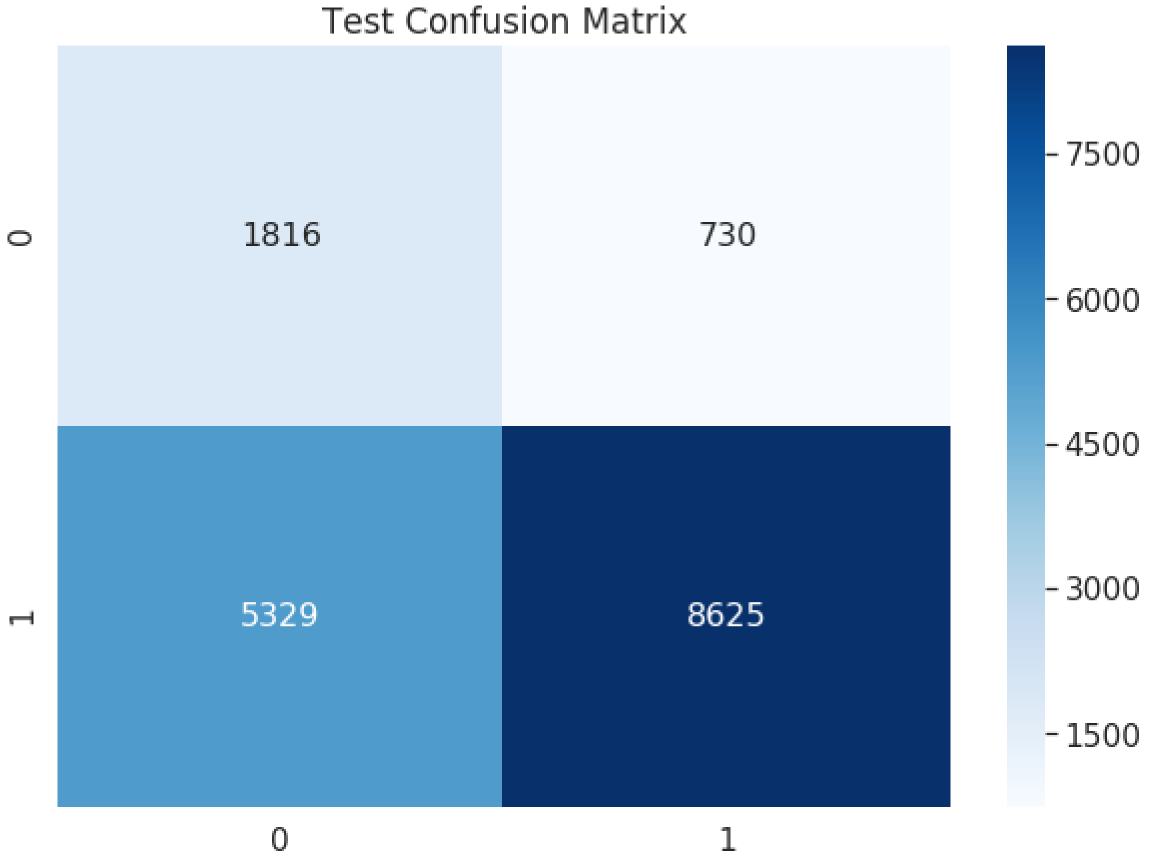


In [0]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[156]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7faeff097da0>
```



2.5.3 Applying XGBOOST on AVG W2V, SET 3

In [0]:

```
# Please write all the code with proper documentation
```

In [80]:

```
#Hyperparameter Tuning
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV

tuned_parameters = {'n_estimators':[10, 50, 100, 150, 200], 'max_depth':[2, 3, 4, 5, 6]}
xgb_model = xgb.XGBClassifier()
clf = RandomizedSearchCV(xgb_model, tuned_parameters, cv=5, scoring='roc_auc', return_train
clf.fit(x_train_avg_w2v, y_train)
```

Out[80]:

```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                           colsample_bylevel=1,
                                           colsample_bynode=1,
                                           colsample_bytree=1, gamma=0,
                                           learning_rate=0.1, max_delta_step
=0,
                                           max_depth=3, min_child_weight=1,
                                           missing=None, n_estimators=100,
                                           n_jobs=1, nthread=None,
                                           objective='binary:logistic',
                                           random_state=0, reg_alpha=0,
                                           reg_lambda=1, scale_pos_weight=1,
                                           seed=None, silent=None, subsample
=1,
                                           verbosity=1),
                   iid='warn', n_iter=10, n_jobs=None,
                   param_distributions={'max_depth': [2, 3, 4, 5, 6],
                                        'n_estimators': [10, 50, 100, 150,
                                                         200]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)
```

In [0]:

```

results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])


#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [0]:

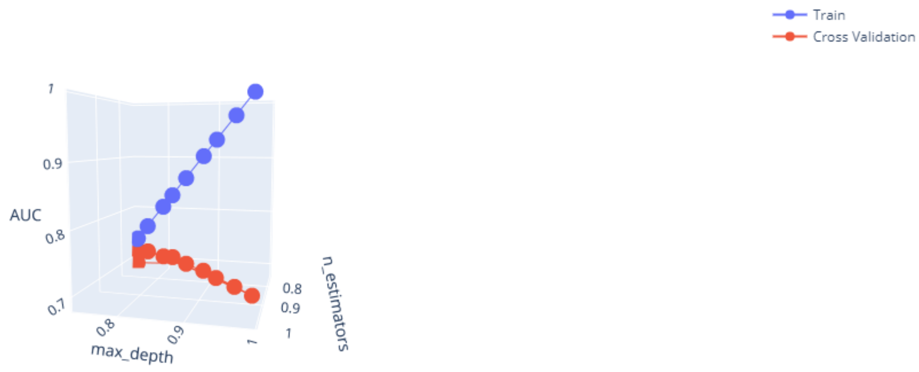
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [85]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```



In [84]:

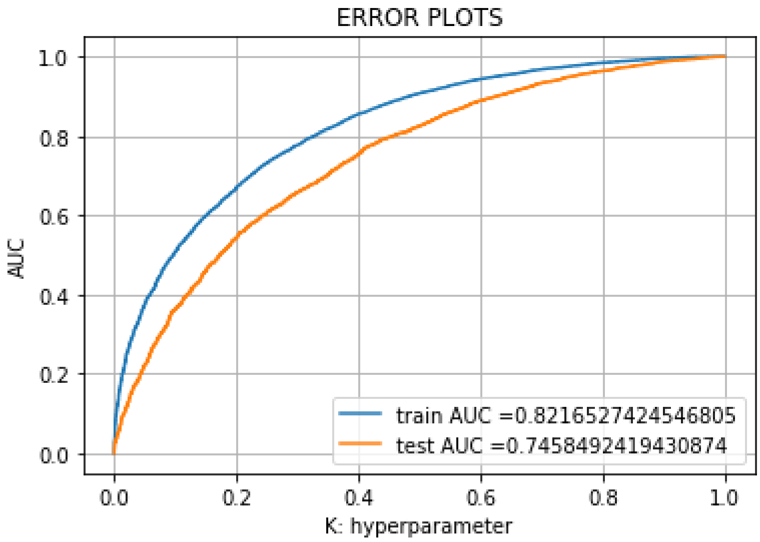
```
clf.best_params_
```

Out[84]:

```
{'max_depth': 3, 'n_estimators': 150}
```

In [87]:

```
from sklearn.metrics import roc_curve, auc

xgb_model = xgb.XGBClassifier(max_depth=3, n_estimators=150)
xgb_model.fit(x_train_avg_w2v, y_train)

y_train_pred = clf.predict_proba(x_train_avg_w2v)[:, 1]
y_test_pred = clf.predict_proba(x_test_avg_w2v)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.grid()
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("ERROR PLOTS")

plt.show()
```



In [91]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

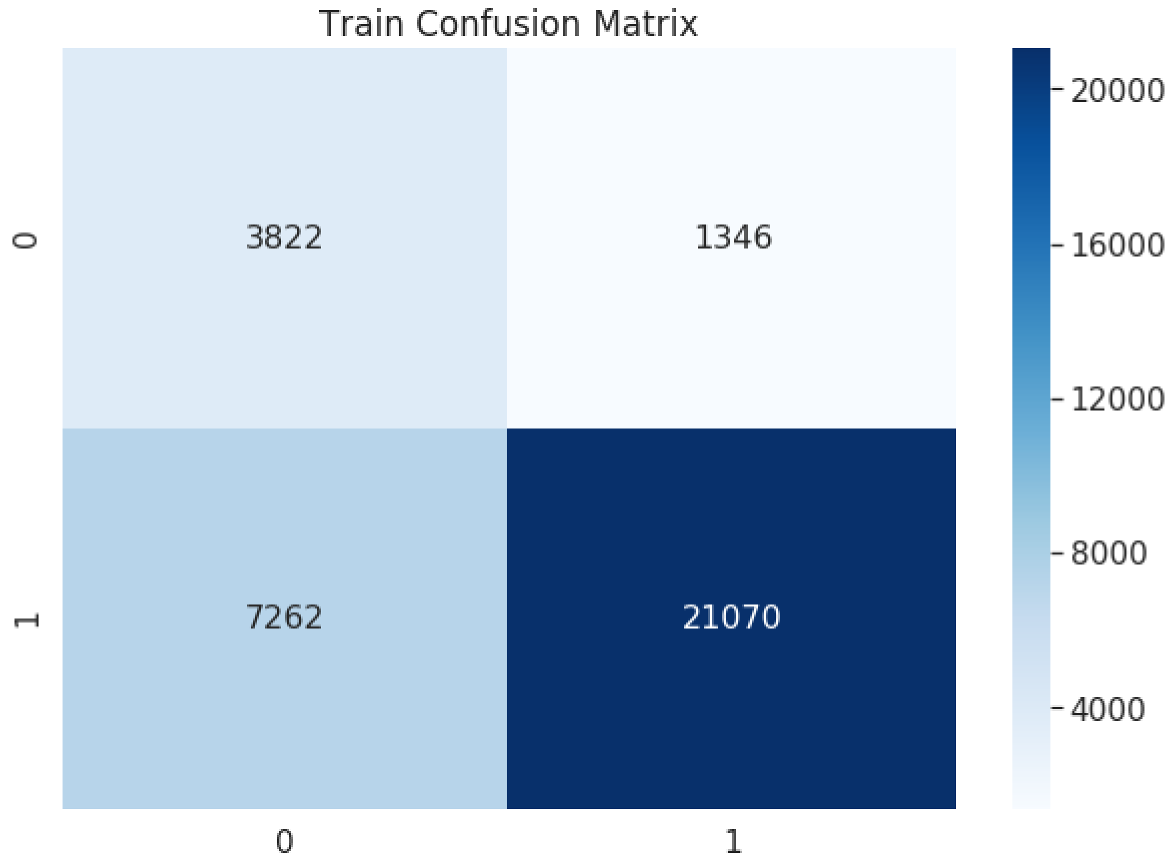
```
the maximum value of tpr*(1-fpr) 0.5499908700857458 for threshold 0.834
Train confusion matrix
[[ 3822  1346]
 [ 7262 21070]]
Test confusion matrix
[[1778  768]
 [4762 9192]]
```

In [92]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[92]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06362171d0>
```

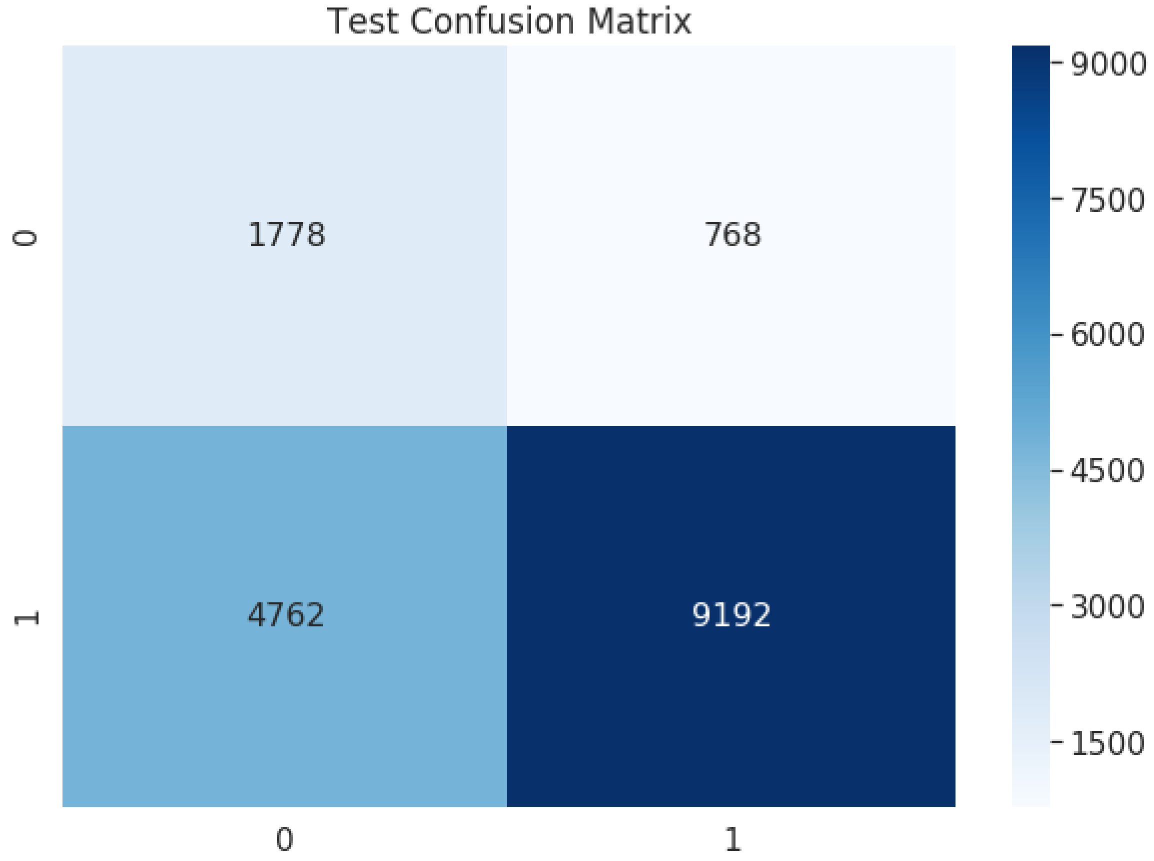


In [93]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0636211e10>
```



In [0]:

```

```

2.5.4 Applying XGBOOST on TFIDF W2V, SET 4

In [0]:

```
# Please write all the code with proper documentation
```

In [94]:

```
#Hyperparameter Tuning

tuned_parameters = {'n_estimators':[10, 50, 100, 150, 200], 'max_depth':[2, 3, 4, 5, 6]}
xgb_model = xgb.XGBClassifier()
clf = RandomizedSearchCV(xgb_model, tuned_parameters, cv=5, scoring='roc_auc', return_trair
clf.fit(x_train_tfidf_w2v, y_train)
```

Out[94]:

```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                           colsample_bylevel=1,
                                           colsample_bynode=1,
                                           colsample_bytree=1, gamma=0,
                                           learning_rate=0.1, max_delta_step
=0,
                                           max_depth=3, min_child_weight=1,
                                           missing=None, n_estimators=100,
                                           n_jobs=1, nthread=None,
                                           objective='binary:logistic',
                                           random_state=0, reg_alpha=0,
                                           reg_lambda=1, scale_pos_weight=1,
                                           seed=None, silent=None, subsample
=1,
                                           verbosity=1),
                   iid='warn', n_iter=10, n_jobs=None,
                   param_distributions={'max_depth': [2, 3, 4, 5, 6],
                                        'n_estimators': [10, 50, 100, 150,
                                                         200]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)
```

In [0]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
n_estimators_results = results.sort_values(['param_n_estimators'])
max_depth_results = results.sort_values(['param_max_depth'])


#For n_estimators
train_auc_n_estimators = results['mean_train_score']
train_auc_std_n_estimators = results['std_train_score']
cv_auc_n_estimators = results['mean_test_score']
cv_auc_std_n_estimators = results['std_test_score']
n_estimators = results['param_n_estimators']

#For max_depth
train_auc_max_depth= results['mean_train_score']
train_auc_std_max_depth= results['std_train_score']
cv_auc_max_depth = results['mean_test_score']
cv_auc_std_max_depth= results['std_test_score']
max_depth =  results['param_max_depth']
```

In [0]:

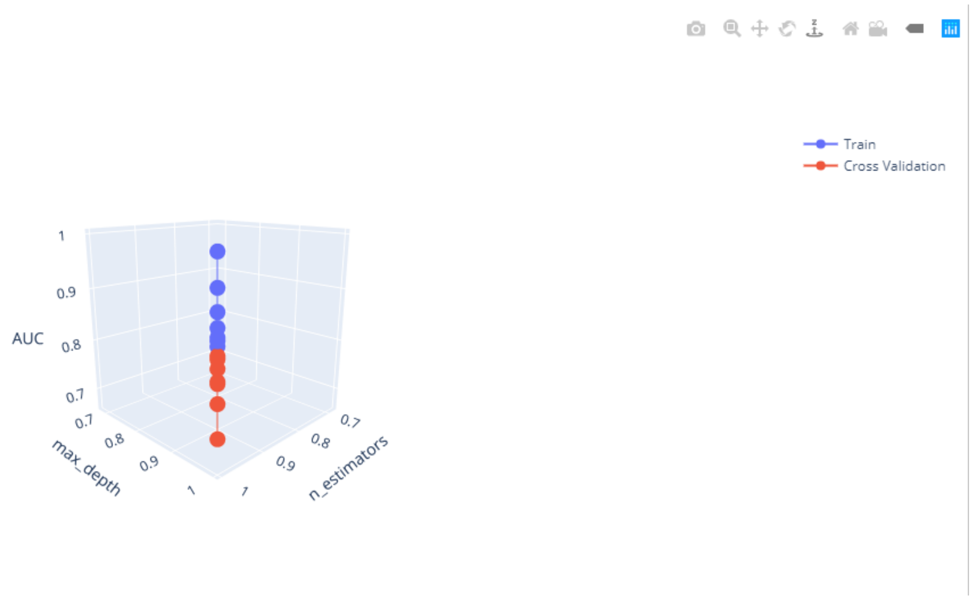
```
x1 = train_auc_n_estimators
y1 = train_auc_max_depth
z1 = results['mean_train_score']

x2 = cv_auc_n_estimators
y2 = cv_auc_max_depth
z2 = results['mean_test_score']
```

In [99]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Train')
trace2 = go.Scatter3d(x=x1, y=y1, z=z2, name='Cross Validation')

data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
#I used colab to run the notebook. The plots I got in colab are not displaying in jupyter n
#So I take screenshots from colab and attaching the screenshots here. suggested by team
```



In [98]:

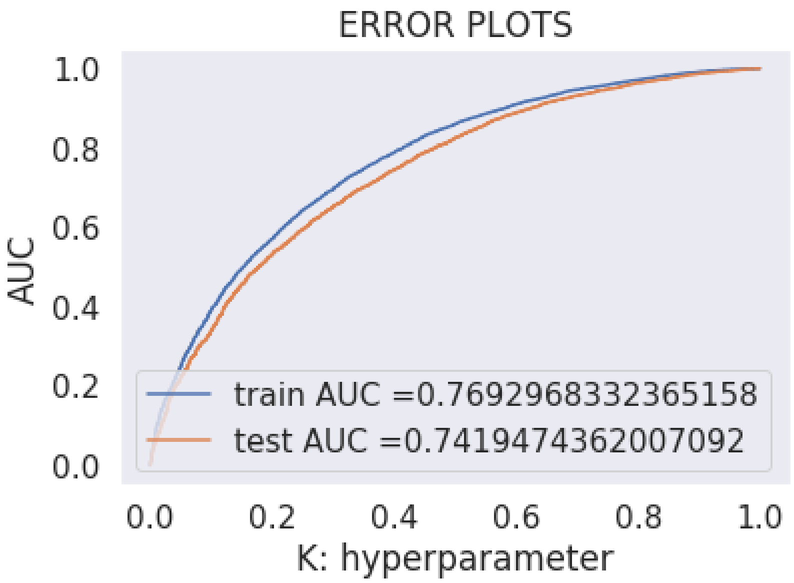
```
clf.best_params_
```

Out[98]:

```
{'max_depth': 2, 'n_estimators': 150}
```

In [100]:

```
from sklearn.metrics import roc_curve, auc

xgb_model = xgb.XGBClassifier(max_depth=2, n_estimators=150)
xgb_model.fit(x_train_tfidf_w2v, y_train)

y_train_pred = clf.predict_proba(x_train_tfidf_w2v)[:, 1]
y_test_pred = clf.predict_proba(x_test_tfidf_w2v)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.grid()
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("ERROR PLOTS")

plt.show()
```



In [101]:

```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

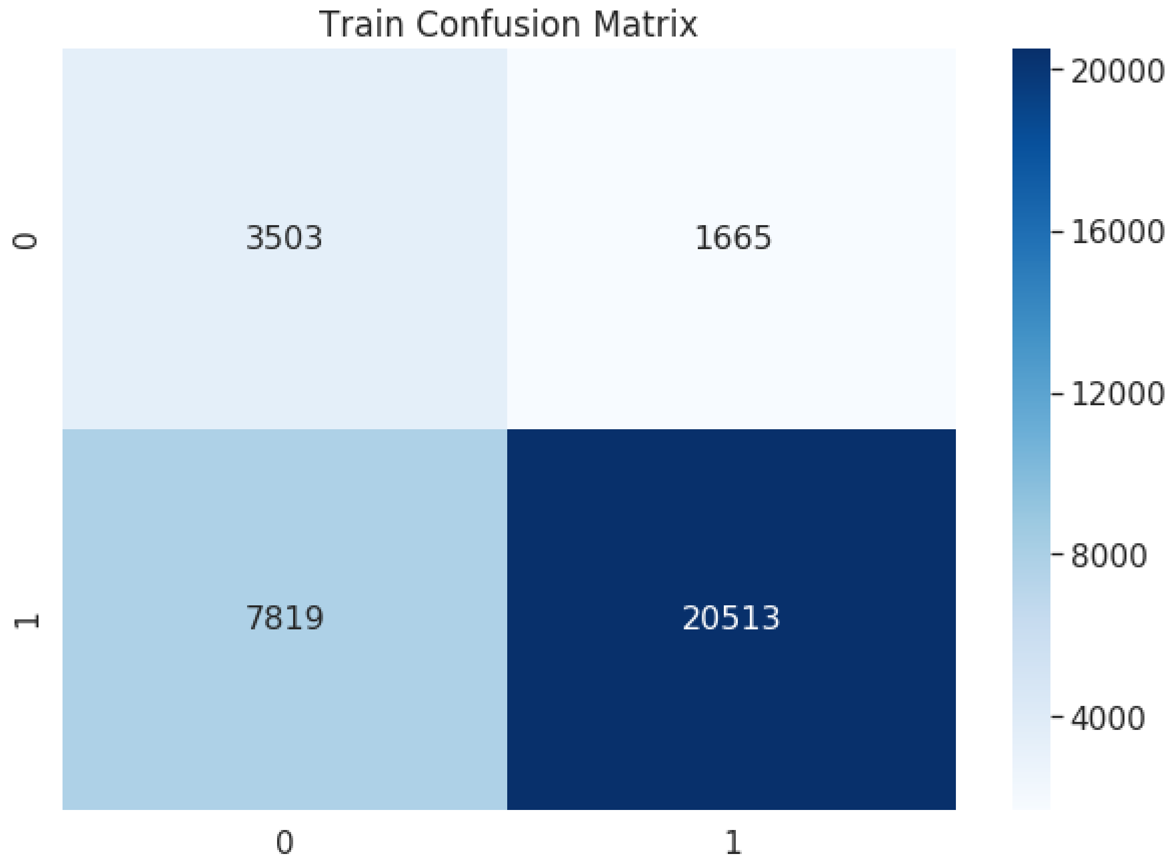
```
the maximum value of tpr*(1-fpr) 0.49076047623512276 for threshold 0.829
Train confusion matrix
[[ 3503  1665]
 [ 7819 20513]]
Test confusion matrix
[[1793  753]
 [4927 9027]]
```

In [102]:

```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.unique(y_trair

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06362479b0>
```

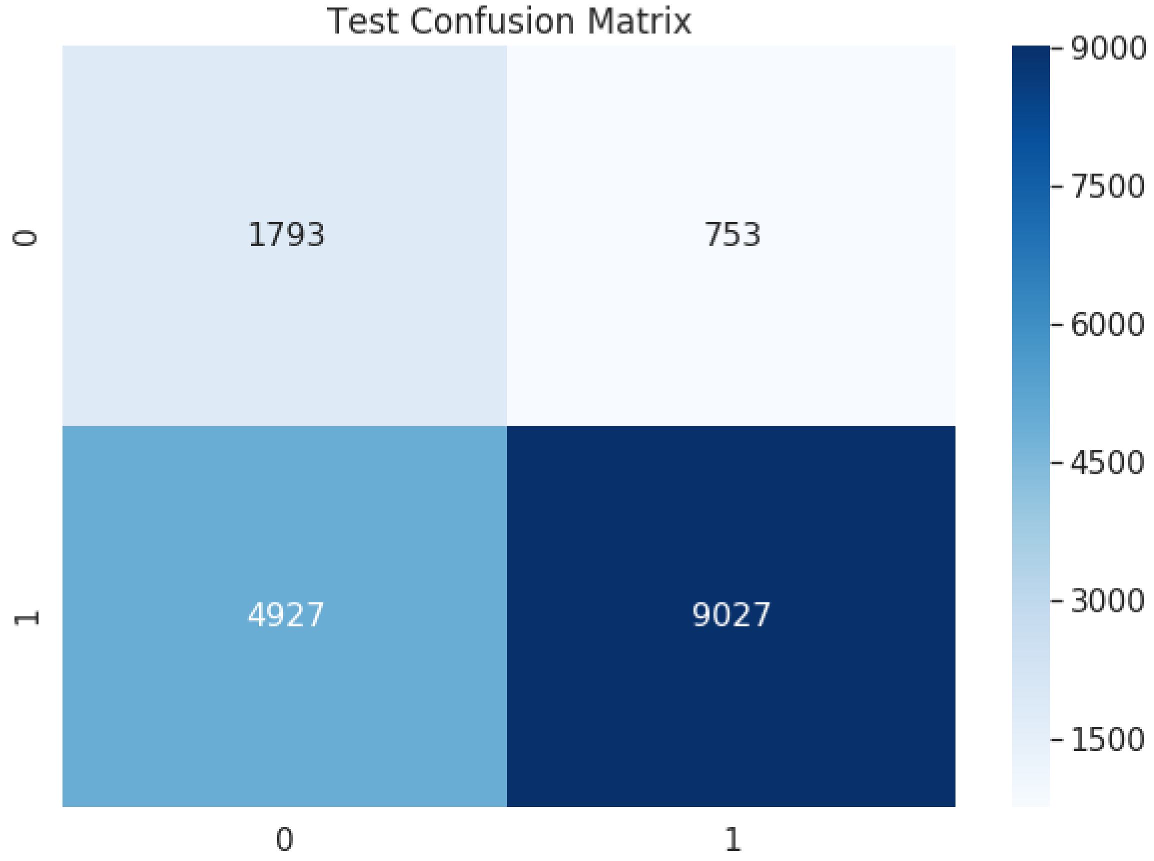


In [103]:

```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.unique(y_test),

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[103]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f063667d780>
```



In [0]:

```
```

3. Conclusion

In [0]:

```
# Please compare all your models using Prettytable library
```

In [104]:

```
from prettytable import PrettyTable

x = PrettyTable()

x. field_names = ['Vectorizer', 'Model', 'max_depth', 'n_estimators', 'Train AUC', 'Test AU

x.add_row(['BOW', 'Random Forest', '8', '500', '0.77', '0.69'])
x.add_row(['TFIDF', 'Random Forest', '8', '500','0.77', '0.69'])
x.add_row(['AVG W2V', 'Random Forest', '8', '500', '0.92', '0.71'])
x.add_row(['TFIDF W2V', 'Random Forest', '8', '500','0.90', '0.70'])
x.add_row([' ', ' ', ' ', ' ', ' ', ' '])

x.add_row(['BOW', 'xgboost', '2', '500', '0.78', '0.73'])
x.add_row(['TFIDF', 'xgboost', '2', '500', '0.80', '0.72'])
x.add_row(['AVG W2V', 'xgboost', '3', '150', '0.82', '0.74'])
x.add_row(['TFIDF W2V', 'xgboost', '2', '150', '0.76', '0.74'])

print(x)
```

```
+------------+---------------+-----------+--------------+-----------+-------
---+
| Vectorizer |     Model     | max_depth | n_estimators | Train AUC | Test A
UC |
+------------+---------------+-----------+--------------+-----------+-------
---+
|    BOW     | Random Forest |     8     |     500      |    0.77   |   0.69
|
|   TFIDF    | Random Forest |     8     |     500      |    0.77   |   0.69
|
|  AVG W2V   | Random Forest |     8     |     500      |    0.92   |   0.71
|
| TFIDF W2V  | Random Forest |     8     |     500      |    0.90   |   0.70
|
|            |               |           |              |           |
|
|    BOW     |    xgboost    |     2     |     500      |    0.78   |   0.73
|
|   TFIDF    |    xgboost    |     2     |     500      |    0.80   |   0.72
|
|  AVG W2V   |    xgboost    |     3     |     150      |    0.82   |   0.74
|
| TFIDF W2V  |    xgboost    |     2     |     150      |    0.76   |   0.74
|
+------------+---------------+-----------+--------------+-----------+-------
---+
```