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`

Feature	Descr
`project_title`	Title of the project. **Exar** • `Art Will Make You Ha` • `First Grade`
`project_grade_category`	Grade level of students for which the project is targeted. One of the fol enumerated v • `Grades Pr` • `Grades` • `Grades` • `Grades`
`project_subject_categories`	One or more (comma-separated) subject categories for the project fr following enumerated list of v • `Applied Lear` • `Care & Hu` • `Health & Sp` • `History & Ci` • `Literacy & Lang` • `Math & Sci` • `Music & The` • `Special N` • `Wa` **Exam** • `Music & The` • `Literacy & Language, Math & Sci`
`school_state`	State where school is located (Two-letter U.S. posta (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_co **Exampl**

Feature	Descr
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the p **Exam** • `Lite` • `Literature & Writing, Social Scie`
`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:** `2016-0 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]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

from plotly import plotly
```

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

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

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

Taking only 50K points as Training runs slower with many points

In [3]:

```
import random
project_data = project_data.loc[random.sample(list(project_data.index), 50000)]
```

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' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [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 [6]:

```
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 [7]:

```
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 [8]:
```
# 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_submitted_
102295	176421	p002860	be85cd8356cfe88ddafacbb06166c944	Mrs.	NY	2016-08-17
858	49400	p246167	711167ea7bfcae20127f2eb90c22fbda	Ms.	HI	2017-01-17

In [10]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [11]:
```python
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
```

```
I teach 360 creative students at a Title I school on the Lower East Side of Manhattan.  O
ur school is truly a barrier free school, serving all students who come through our door
s.  Almost 10% of our students are English language learners and 33% are students with di
sabilities.  Almost half of our students stay in after-school until after 5 PM each day w
hile their parents work. Art class is a place where students are able to express themselv
es, take part in hands on learning, and have fun during the school day.When students have
a sketchbook full of their creative ideas they are better able to grow as artists.  If my
students have a place to store all of their lesson handouts throughout the year it will h
elp them retain the skills and information from each unit, and allow them to look back on
prior techniques that they can use going forward. Doubling this resource as a sketchbook
will allow them to track their artistic growth by reflecting on their sketches as the yea
r progresses. The art room can be a messy place with many materials and classes rotating
in and out throughout the day.  The requested supplies will be a very valuable addition t
o the art room's organization and a solid resource for students.nannan
==================================================
Funding is desperately needed to combat the most pressing issue facing students: (1) sing
le parent homes, poverty and family factors that take priority over education; (2) drug a
buse and families with drug abuse; and (3) character education, care for others, and bull
ying. Often times these students lack basic school supplies. Receiving funding for educat
ion tools for these students would help to give them a fighting chance in the education w
orld. Thank you in advance for your loving support to fund the education of our future le
aders.TIME For Kids is a weekly newsmagazine that aims to engage students while presentin
g them with high-quality nonfiction writing to build reading and critical thinking skill
s.  Each issue covers a variety of themes, topics and current events.  The magazine subsc
ription comes with access to their website for printable to be used in the classroom. TIM
```

```
E For Kids is committed to helping teachers meet the Common Core State Standards. Keeping
in mind the Common Core’s particular emphasis on the reading of informational text, TIME
For Kids education editors have taken steps to help ensure that students practice and mas
ter the literacy skills highlighted in the Common Core.\r\n\r\nResource: timeforkids.comn
annan
==================================================
Students come to our library to read, work, learn, create, and relax. We want to provide
them with a great experience in their library! \r\nOur school library is an active hub of
learning for our 1700+ students! Our students attend a school where they are challenged t
o succeed in academics, arts, and athletics. Our students come from diverse backgrounds a
nd several come to our school from neighborhoods all over our city. Our school community
includes students with disabilities and ESL students. Our students are known for being hi
gh achievers, as well as for making a difference in their school and communities by being
active in many extracurricular activities.Library makerspaces provide students with the o
pportunity to explore their interests, create, innovate, and share in a stress-free conte
xt.  We are creating a brand new Makerspace in our school library.  Our students need mat
erials to start them off on their \"making\" journey!\r\n\r\nThe materials we are request
ing will provide students with hands-on STEM and art learning activities that they can en
gage in daily in their school library.  Many of our students are interested in STEM and a
rt, and we want to value, honor, and encourage our students' interests in our Makerspace.
The Lego Architecture set, the Lego blocks/baseplates, and the KNEX set will provide our
students with multi-leveled engineering experiences. We want our Makerspace to be accessi
ble to all of our students!  The Duct Tape/Washi tape project books and materials as well
as the calligraphy book and materials will provide a venue for creativity and artistry fo
r our students.  We can't wait to see all of the amazing learning, innovation, and enjoym
ent that will result from our school library Makerspace! \r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n
nannan
==================================================
I serve twenty-seven energetic, fun-loving kindergarten students. They live in the Appala
chian Mountains of Eastern Kentucky. They are eager to please and love to learn new thing
s.\r\nI want each of my students to succeed in every educational aspect of their lives.\r
\nThis is a very important step in a young child's life. They are learning so fast and so
much that it's critical to keep them focused and on task. My goal is for them to have fu
n, yet master the skills that are needed for them to progress in every way.My student's i
dea is to have a station set up to allow them to watch DVD's and dance along to get more
exercise. They told me how tired they get during the day of just sitting and not stirring
```

```
around. \"We want exercise but we want to dance to get it and Barney is our favorite way
to get us moving and shaking\" is what most have voiced to me.\r\nSo, yes, they will get
their \"moving and shaking\" project they have inspired me to write.\r\nThe television wi
ll be used with the DVD player to allow them to watch the Barney videos and get them off
their seats. They can use the locking desk for the television and DVD player to sit on as
well as to keep the videos safely stored away. They need to learn about physical fitness
and the benefits of exercise on their little bodies and I'm going to teach them all about
making these lifestyle changes. If Barney is their inspiration to get them excited about
these changes, who am I to stand in the way of progress? \r\nThey asked for these materia
ls and I'm really hoping with your generous donations it becomes a reality for my student
s. They are just so very excited and waiting to see if their idea will come to life like
Barney does.nannan
==================================================
```

In [12]:

```python
# 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 [13]:

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

```
I serve twenty-seven energetic, fun-loving kindergarten students. They live in the Appala
chian Mountains of Eastern Kentucky. They are eager to please and love to learn new thing
s.\r\nI want each of my students to succeed in every educational aspect of their lives.\r
\nThis is a very important step in a young child is life. They are learning so fast and s
o much that it is critical to keep them focused and on task. My goal is for them to have
fun, yet master the skills that are needed for them to progress in every way.My student i
s idea is to have a station set up to allow them to watch DVD is and dance along to get m
ore exercise. They told me how tired they get during the day of just sitting and not stir
ring around. \"We want exercise but we want to dance to get it and Barney is our favorite
way to get us moving and shaking\" is what most have voiced to me.\r\nSo, yes, they will
get their \"moving and shaking\" project they have inspired me to write.\r\nThe televisio
n will be used with the DVD player to allow them to watch the Barney videos and get them
off their seats. They can use the locking desk for the television and DVD player to sit o
n as well as to keep the videos safely stored away. They need to learn about physical fit
ness and the benefits of exercise on their little bodies and I am going to teach them all
about making these lifestyle changes. If Barney is their inspiration to get them excited
about these changes, who am I to stand in the way of progress? \r\nThey asked for these m
aterials and I am really hoping with your generous donations it becomes a reality for my
students. They are just so very excited and waiting to see if their idea will come to lif
e like Barney does.nannan
==================================================
```

In [14]:

```
# \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)
```

```
I serve twenty-seven energetic, fun-loving kindergarten students. They live in the Appala
chian Mountains of Eastern Kentucky. They are eager to please and love to learn new thing
s.  I want each of my students to succeed in every educational aspect of their lives.  Th
is is a very important step in a young child is life. They are learning so fast and so mu
ch that it is critical to keep them focused and on task. My goal is for them to have fun,
yet master the skills that are needed for them to progress in every way.My student is ide
a is to have a station set up to allow them to watch DVD is and dance along to get more e
xercise. They told me how tired they get during the day of just sitting and not stirring 
around.  We want exercise but we want to dance to get it and Barney is our favorite way t
o get us moving and shaking  is what most have voiced to me.  So, yes, they will get thei
r  moving and shaking  project they have inspired me to write.  The television will be us
ed with the DVD player to allow them to watch the Barney videos and get them off their se
ats. They can use the locking desk for the television and DVD player to sit on as well as
to keep the videos safely stored away. They need to learn about physical fitness and the 
benefits of exercise on their little bodies and I am going to teach them all about making
these lifestyle changes. If Barney is their inspiration to get them excited about these c
hanges, who am I to stand in the way of progress?   They asked for these materials and I 
am really hoping with your generous donations it becomes a reality for my students. They 
are just so very excited and waiting to see if their idea will come to life like Barney d
oes.nannan
```

In [15]:

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

```
I serve twenty seven energetic fun loving kindergarten students They live in the Appalach
ian Mountains of Eastern Kentucky They are eager to please and love to learn new things I
want each of my students to succeed in every educational aspect of their lives This is a
very important step in a young child is life They are learning so fast and so much that i
t is critical to keep them focused and on task My goal is for them to have fun yet master
the skills that are needed for them to progress in every way My student is idea is to hav
e a station set up to allow them to watch DVD is and dance along to get more exercise The
y told me how tired they get during the day of just sitting and not stirring around We wa
nt exercise but we want to dance to get it and Barney is our favorite way to get us movin
g and shaking is what most have voiced to me So yes they will get their moving and shakin
g project they have inspired me to write The television will be used with the DVD player
to allow them to watch the Barney videos and get them off their seats They can use the lo
cking desk for the television and DVD player to sit on as well as to keep the videos safe
ly stored away They need to learn about physical fitness and the benefits of exercise on
their little bodies and I am going to teach them all about making these lifestyle changes
If Barney is their inspiration to get them excited about these changes who am I to stand
in the way of progress They asked for these materials and I am really hoping with your ge
nerous donations it becomes a reality for my students They are just so very excited and w
aiting to see if their idea will come to life like Barney does nannan
```

In [16]:

```
# 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 [17]:

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

```
100%|████████████████████████████████████████████████| 50000/50
000 [00:42<00:00, 1171.57it/s]
```

```
In [18]: # after preprocesing
         preprocessed_essays[20000]
```

```
Out[18]: 'i serve twenty seven energetic fun loving kindergarten students they live appalachian mo
untains eastern kentucky they eager please love learn new things i want students succeed
every educational aspect lives this important step young child life they learning fast mu
ch critical keep focused task my goal fun yet master skills needed progress every way my
student idea station set allow watch dvd dance along get exercise they told tired get day
sitting not stirring around we want exercise want dance get barney favorite way get us mo
ving shaking voiced so yes get moving shaking project inspired write the television used
dvd player allow watch barney videos get seats they use locking desk television dvd playe
r sit well keep videos safely stored away they need learn physical fitness benefits exerc
ise little bodies i going teach making lifestyle changes if barney inspiration get excite
d changes i stand way progress they asked materials i really hoping generous donations be
comes reality students they excited waiting see idea come life like barney nannan'
```

1.4 Preprocessing of `project_title`

```
In [19]: # similarly you can preprocess the titles also
```

Following Code blocks provided by me.

In [20]:
```
# Code took from original code provided.
# Also function used from original code.
preprocessed_titles = []

for sent in tqdm(project_data['project_title'].values):
    sent = decontracted(sent)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = ' '.join(e.lower() for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████| 50000/500
00 [00:01<00:00, 31054.33it/s]
```

In [21]:
```
preprocessed_titles[20000]
```

Out[21]:
```
'let get moving shaking kindergarten'
```

Following Code blocks present in original notebook.

1.5 Preparing data for models

In [22]:
```
project_data.columns
```

Out[22]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

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 [23]:

```
# 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', 'SpecialNeed
s', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (50000, 9)
```

In [24]:

```
# 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', 'Civics_Gover
nment', 'Extracurricular', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hung
er', 'SocialSciences', 'CharacterEducation', 'PerformingArts', 'TeamSports', 'Other', 'Co
llege_CareerPrep', 'History_Geography', 'Music', 'EarlyDevelopment', 'Health_LifeScienc
e', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'Appli
edSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (50000, 30)
```

In [25]:

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

Following Code blocks provided by me.

In [26]:

```
# Code took from original code provided.
states = project_data['school_state'].unique()
vectorizer = CountVectorizer(vocabulary=list(states), 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)
```

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

There are some NaN's in teacher_prefix column. replacing them with 'Mrs.' as that has high occurance in that column.

In [27]:

```
print("Number of NaN's before replacement in column: ", sum(project_data['teacher_prefix'].
project_data['teacher_prefix'] = project_data['teacher_prefix'].replace(np.nan, 'Mrs.', reg
print("Number of NaN's after replacement in column: ", sum(project_data['teacher_prefix'].i

# Output may show both zeros as I re-run this several times. But there are 3 zeros in origi
```

```
Number of NaN's before replacement in column:  0
Number of NaN's after replacement in column:  0
```

In [28]:

```
# Code took from original code provided.
prefixes = project_data['teacher_prefix'].unique()
vectorizer = CountVectorizer(vocabulary=list(prefixes), lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())

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

```
['Mrs.', 'Ms.', 'Teacher', 'Mr.', 'Dr.']
Shape of matrix after one hot encoding (50000, 5)
```

In [29]:

```
grades = project_data['project_grade_category'].unique()
vectorizer = CountVectorizer(vocabulary=list(grades), lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())

project_grade_category_one_hot = vectorizer.transform(project_data['project_grade_category'
print("Shape of matrix after one hot encoding", project_grade_category_one_hot.shape)
```

```
['Grades PreK-2', 'Grades 6-8', 'Grades 3-5', 'Grades 9-12']
Shape of matrix after one hot encoding (50000, 4)
```

Following Code blocks present in original notebook.

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [30]:

```
# 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, 12234)
```

In [31]:

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

Following Code blocks provided by me.

In [32]:

```
# Code took from original code provided.
# We are considering only the words which appeared in at least 5 documents(rows or projects
# Reduced number as title has less words
vectorizer = CountVectorizer(min_df=10)
titles_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ", titles_bow.shape)
```

```
Shape of matrix after one hot encodig  (50000, 2034)
```

Following Code blocks present in original notebook.

1.5.2.2 TFIDF vectorizer

In [33]:

```
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, 12234)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [34]:

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

In [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%|██████████████████████████████████████████████████| 50000/50
000 [00:20<00:00, 2430.11it/s]

50000
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [36]:
```
# 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())) # gettin
            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%|██████████████████████████████████████████████████████████████████| 50000/5
0000 [03:19<00:00, 251.06it/s]

50000
300
```

In [38]:

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

Following Code blocks provided by me.

In [39]:
```
# Code took from original code provided.
# tfidf of project titles
vectorizer = TfidfVectorizer(min_df=10)
titles_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",titles_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (50000, 2034)
```

In [40]:
```
# Code took from original code provided.
# avg-w2v for project titles
avg_w2v_titles = []
for sentence in tqdm(preprocessed_titles):
    vector = np.zeros(300)
    cnt_words =0;
    for word in sentence.split():
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles.append(vector)

print(len(avg_w2v_titles))
print(len(avg_w2v_titles[0]))
```

```
100%|█████████████████████████████████████████████████████████████████████████| 50000/500
00 [00:00<00:00, 60567.34it/s]

50000
300
```

In [41]:

```
# Code took from original code provided.
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [42]:

```
# Code took from original code provided.
# tfidf-w2v for project titles
tfidf_w2v_titles = []
for sentence in tqdm(preprocessed_titles):
    vector = np.zeros(300)
    tf_idf_weight =0
    for word in sentence.split():
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_titles.append(vector)

print(len(tfidf_w2v_titles))
print(len(tfidf_w2v_titles[0]))
```

```
100%|████████████████████████████████████████████████████████████████████| 50000/500
00 [00:02<00:00, 23443.57it/s]

50000
300
```

Following Code blocks present in original notebook.

1.5.3 Vectorizing Numerical features

In [43]:
```
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 [44]:
```
# 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 : 298.9165988, Standard deviation : 366.4424409289348
```

In [45]:
```
price_standardized
```

Out[45]:
```
array([[-0.28238159],
       [-0.07577342],
       [ 0.37788582],
       ...,
       [-0.68656512],
       [-0.18242046],
       [ 0.81891006]])
```

Following Code blocks provided by me.

In [46]:

```
warnings.filterwarnings("ignore")
# Code took from original code provided
scalar = StandardScaler()
scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,
print(f"Mean : {scalar.mean_[0]}, Standard deviation : {np.sqrt(scalar.var_[0])}")

# Now standardize the data with above maen and variance.
previously_posted_projects_standardized = \
                scalar.transform(project_data['teacher_number_of_previously_posted_projects
print(previously_posted_projects_standardized)
```

```
Mean : 11.07926, Standard deviation : 27.729798734437292
[[-0.36348118]
 [ 0.71838747]
 [-0.39954347]
 ...
 [-0.21923203]
 [ 0.28564001]
 [-0.36348118]]
```

Following Code blocks present in original notebook.

1.5.4 Merging all the above features

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

In [47]:

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

```
(50000, 9)
(50000, 30)
(50000, 12234)
(50000, 1)
```

In [48]:

```
# 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[48]:

```
(50000, 12274)
```

In [49]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# 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 [50]:

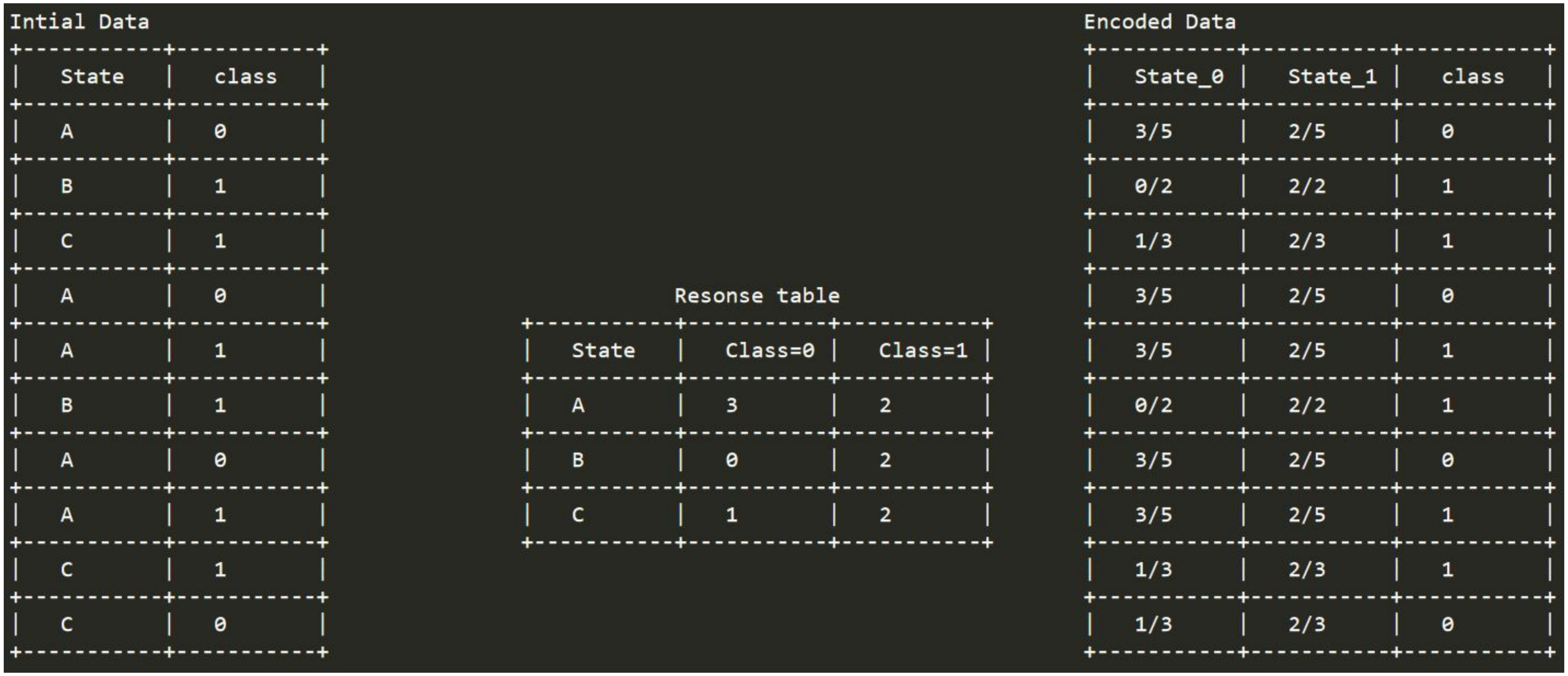
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
nltk.download('vader_lexicon')

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 o
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_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\narayana\AppData\Roaming\nltk_data...
```

```
[nltk_data]   Package vader_lexicon is already up-to-date!
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**

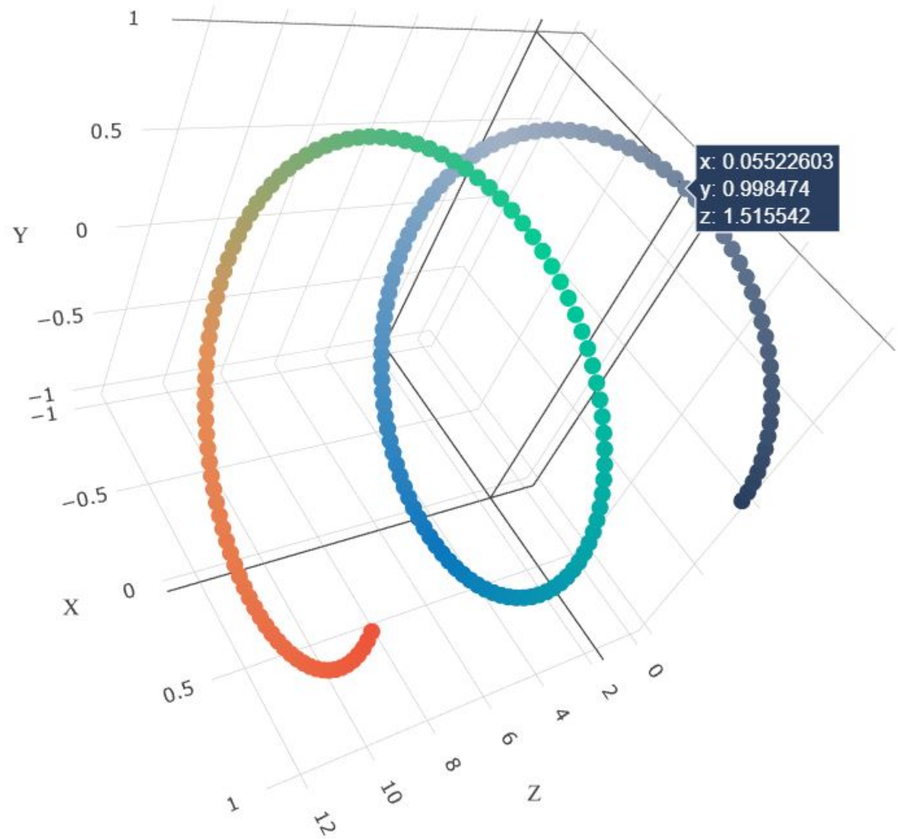
    - Set 1: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical(instead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

    - 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**

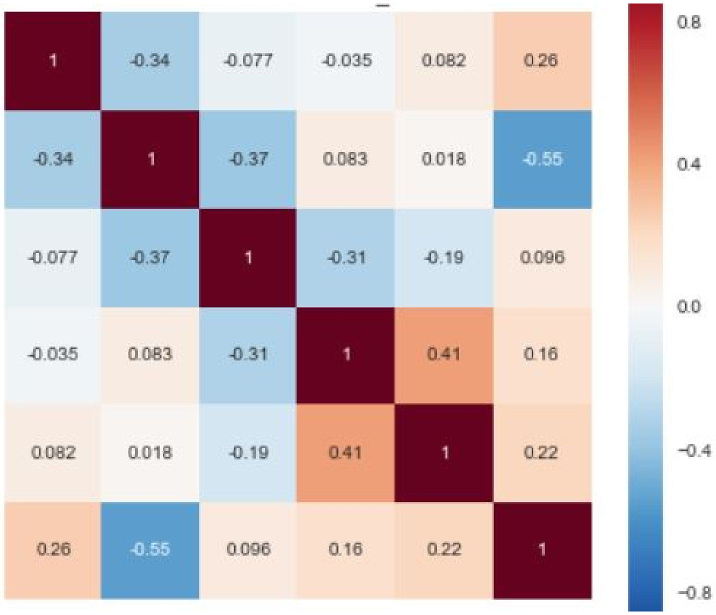
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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*

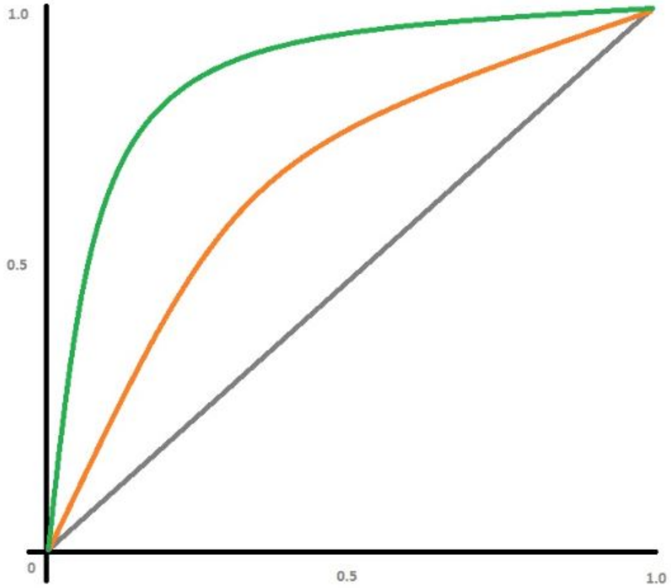
or

- 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

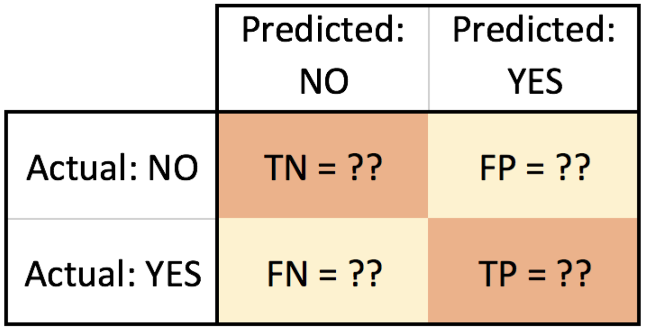


seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**

- 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.



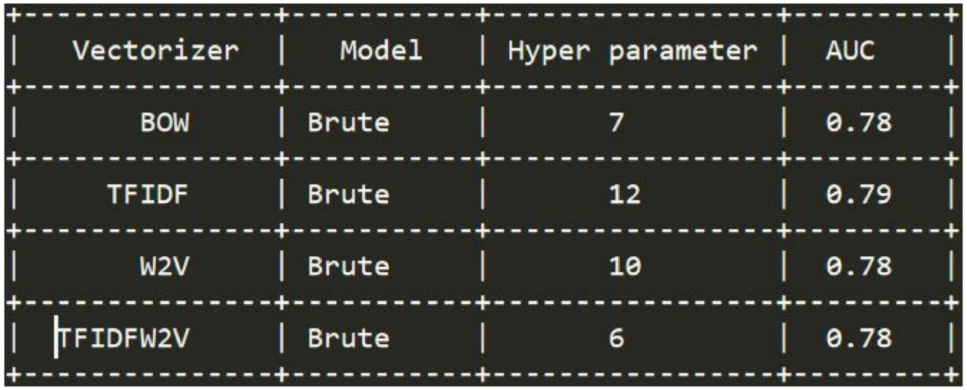
- Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points



	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

Some code blocks are taken from previous assignments. And some used the code present in original file ('9_DonorsChoose_RF_GBDT') which is mentioned in comments.

Following Code blocks provided by me.

Adding a column `summary_numeric_bool` instead of `project_resource_summary` column which tells if resource summary has a number in it

In [51]:

```
# ref: https://stackoverflow.com/questions/4138202/using-isdigit-for-floats
def nums_in_str(text):
    """
    Returns list of numbers present in the given string. Numbers := floats ints etc.
    """
    result = []
    for s in text.split():
        try:
            x = float(s)
            result.append(x)
        except:
            continue
    return result
```

In [52]:

```
print(nums_in_str('HE44LLo 56 are -89 I 820.353 in -78.39 what .293 about 00'))
```

```
[56.0, -89.0, 820.353, -78.39, 0.293, 0.0]
```

In [53]:

```
numbers_in_summary = np.array([len(nums_in_str(s)) for s in project_data['project_resource_
project_data['summary_numeric_bool'] = list(map(int, numbers_in_summary>0))
```

Taking Relevant columns as X (input data to model) and y (output class label)

In [54]:
```
project_data.columns
```

Out[54]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'summary_numeric_bool'],
      dtype='object')
```

In [55]:
```
project_data.head(2)
```

Out[55]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_dateti
0	176421	p002860	be85cd8356cfe88ddafacbb06166c944	Mrs.	NY	2016-08-17 11:48
1	49400	p246167	711167ea7bfcae20127f2eb90c22fbda	Ms.	HI	2017-01-17 00:41

2 rows × 21 columns

In [56]:

```
# Categorical and numerical columns are listed below.
X_columns = ['teacher_prefix', 'school_state', 'project_grade_category', 'summary_numeric_b
             'teacher_number_of_previously_posted_projects', 'clean_categories', 'clean_sub
             'price', 'quantity']
X = project_data[X_columns]
y = project_data['project_is_approved']
```

Adding preprocessed_essays and preprocessed_titles as columns to X before splitting

In [57]:

```
X['essay'] = preprocessed_essays
X['project_title'] = preprocessed_titles
X_columns.append('essay')
X_columns.append('project_title')
print('final columns used in input data are: ', X_columns)
```

```
final columns used in input data are:  ['teacher_prefix', 'school_state', 'project_grade_
category', 'summary_numeric_bool', 'teacher_number_of_previously_posted_projects', 'clean
_categories', 'clean_subcategories', 'price', 'quantity', 'essay', 'project_title']
```

In [58]:

```
X['essay_word_count'] = [len(es.split()) for es in X['essay']]
X['title_word_count'] = [len(title.split()) for title in X['project_title']]
```

In [59]:

```
print(X.columns)
```

```
Index(['teacher_prefix', 'school_state', 'project_grade_category',
       'summary_numeric_bool', 'teacher_number_of_previously_posted_projects',
       'clean_categories', 'clean_subcategories', 'price', 'quantity', 'essay',
       'project_title', 'essay_word_count', 'title_word_count'],
      dtype='object')
```

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

Sampling

In [60]:

```
# 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
```

Not creating CV data as I am using K-fold validation

In [61]:

```
# Code took from SAMPLE_SOLUTION notebook
# splitting into 80-20 ratio for train-test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, stratify=y)
```

In [62]:

```
print(X_train.shape)
print(X_test.shape)
print('='*30)
print(y_train.shape)
print(y_test.shape)
```

```
(40000, 13)
(10000, 13)
==============================
(40000,)
(10000,)
```

2.2 Make Data Model Ready: encoding numerical, categorical

features

In [63]:
```
# 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 label
```

numerical columns

- `teacher_number_of_previously_posted_projects`
- `price`
- `quantity`

Leaving `summary_numeric_bool` as it is because it only has 0's and 1's in it.

categorical columns

- `teacher_prefix`
- `school_state`
- `project_grade_category`
- `clean_categories`
- `clean_subcategories`

Normalizing `teacher_number_of_previously_posted_projects` column

In [64]:

```
warnings.filterwarnings("ignore")
# Code took from original Code provided.
scaler = StandardScaler()
scaler.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
print(f"Mean : {scaler.mean_[0]}, Standard deviation : {np.sqrt(scaler.var_[0])}")
```

```
Mean : 11.05455, Standard deviation : 27.583383119144393
```

In [65]:

```
warnings.filterwarnings("ignore")
X_train_tnppp_norm = scaler.transform(X_train['teacher_number_of_previously_posted_projects
X_test_tnppp_norm = scaler.transform(X_test['teacher_number_of_previously_posted_projects']
```

Normalizing price column

In [66]:

```
# Code took from original Code provided.
scaler = StandardScaler()
scaler.fit(X_train['price'].values.reshape(-1,1))
print(f"Mean : {scaler.mean_[0]}, Standard deviation : {np.sqrt(scaler.var_[0])}")
```

```
Mean : 298.93655575, Standard deviation : 365.11354870027714
```

In [67]:

```
X_train_price_norm = scaler.transform(X_train['price'].values.reshape(-1,1))
X_test_price_norm = scaler.transform(X_test['price'].values.reshape(-1,1))
```

Normalizing quantity column

In [68]:

```
warnings.filterwarnings("ignore")
# Code took from original Code provided.
scaler = StandardScaler()
scaler.fit(X_train['quantity'].values.reshape(-1,1))
print(f"Mean : {scaler.mean_[0]}, Standard deviation : {np.sqrt(scaler.var_[0])}")
```

```
Mean : 17.02945, Standard deviation : 26.22273217453704
```

In [69]:

```
warnings.filterwarnings("ignore")
X_train_quant_norm = scaler.transform(X_train['quantity'].values.reshape(-1,1))
X_test_quant_norm = scaler.transform(X_test['quantity'].values.reshape(-1,1))
```

Using a array to store column names data to use at last when interpreting the model

In [70]:

```
# when combining the input matrix the order of columns is same as cat_num_columns
cat_num_columns = ['previously_posted_projects', 'price', 'quantity', 'summary_numeric_bool
```

Function for Response Coding given input data

In [71]:

```
from collections import defaultdict

def response_coding(col_data, y_data):
    data = pd.concat([col_data, y_data], axis=1)
    data['Counter'] = 1
    cols = data.columns.values
    grp = data.groupby([cols[0], cols[1]]).count()
    # Calculating Response coding value for each index in a loop and converting to dictiona
    resp_val_dict = dict([(ind, float(grp.loc[ind].loc[1]/grp.loc[ind].sum())) if grp.loc[i
                          else (ind, 0) for ind in grp.index.levels[0]])
    # Calculating mean of reaponse values for any other unknown data
    mean_resp_val = np.mean(list(resp_val_dict.values()))
    # Making a defaultdict and returned
    final_dict = defaultdict(lambda: mean_resp_val, resp_val_dict)
    return final_dict
```

In [72]:

```
resp_code = response_coding(X_train['teacher_prefix'], y_train)
```

In [73]:

```
print(resp_code)
```

```
defaultdict(<function response_coding.<locals>.<lambda> at 0x000002670028DA60>, {'Dr.':
0.6666666666666666, 'Mr.': 0.8524507326932794, 'Mrs.': 0.8569594369679966, 'Ms.': 0.83948
35250123475, 'Teacher': 0.790167865707434})
```

In [74]:

```
resp_code['Ms.']
```

Out[74]:

```
0.8394835250123475
```

In [75]:

```
resp_code['Teacher']
```

Out[75]:

```
0.790167865707434
```

In [76]:
```
resp_code['Some_prefix'] # Given some unknown input.. it returns the mean of all values
```

Out[76]:
```
0.8011456454095448
```

Function for transforming new_data into responce variable given the Response_coding output

In [77]:
```
def resp_code_transform(resp_code, data):
    ans = [resp_code[datum] for datum in data]
    ans = pd.DataFrame(ans, index=data.index)
    return ans
```

In [78]:
```
resp_code_transform(resp_code, X_test['teacher_prefix'])[:10]
```

Out[78]:

	0
24670	0.856959
27860	0.856959
14702	0.856959
10389	0.856959
16404	0.856959
35601	0.856959
38132	0.856959
19483	0.839484
32955	0.856959
3105	0.839484

Now we can use these functions for response coding of our categorical data

Response Coding `teacher_prefix` column

In [79]:
```
resp_code = response_coding(X_train['teacher_prefix'], y_train)
X_train_prefix_resp_code = resp_code_transform(resp_code, X_train['teacher_prefix'])
X_test_prefix_resp_code = resp_code_transform(resp_code, X_test['teacher_prefix'])
```

Response Coding `school_state` column

In [80]:
```
resp_code = response_coding(X_train['school_state'], y_train)
X_train_school_resp_code = resp_code_transform(resp_code, X_train['school_state'])
X_test_school_resp_code = resp_code_transform(resp_code, X_test['school_state'])
```

Response Coding `project_grade_category` column

In [81]:
```
resp_code = response_coding(X_train['project_grade_category'], y_train)
X_train_grade_resp_code = resp_code_transform(resp_code, X_train['project_grade_category'])
X_test_grade_resp_code = resp_code_transform(resp_code, X_test['project_grade_category'])
```

Response Coding `clean_categories` column

In [82]:
```
resp_code = response_coding(X_train['clean_categories'], y_train)
X_train_categ_resp_code = resp_code_transform(resp_code, X_train['clean_categories'])
X_test_categ_resp_code = resp_code_transform(resp_code, X_test['clean_categories'])
```

Response Coding `clean_subcategories` column

In [83]:
```
resp_code = response_coding(X_train['clean_subcategories'], y_train)
X_train_subcat_resp_code = resp_code_transform(resp_code, X_train['clean_subcategories'])
X_test_subcat_resp_code = resp_code_transform(resp_code, X_test['clean_subcategories'])
```

Combining categorical and numerical data for further use.

In [84]:
```
from scipy.sparse import hstack
cat_num_train = hstack((X_train_tnppp_norm, X_train_price_norm, X_train_quant_norm,\
                        np.array(X_train['summary_numeric_bool']).reshape(-1, 1),\
                        X_train_prefix_resp_code, X_train_grade_resp_code, X_train_school_r
                        X_train_categ_resp_code, X_train_subcat_resp_code))
cat_num_test = hstack((X_test_tnppp_norm, X_test_price_norm, X_test_quant_norm,\
                       np.array(X_test['summary_numeric_bool']).reshape(-1, 1),\
                       X_test_prefix_resp_code, X_test_grade_resp_code, X_test_school_resp_
                       X_test_categ_resp_code, X_test_subcat_resp_code))
```

In [85]:
```
print(cat_num_train.shape, y_train.shape)
print(cat_num_test.shape, y_test.shape)
```

```
(40000, 9) (40000,)
(10000, 9) (10000,)
```

2.3 Make Data Model Ready: encoding essay, and project_title

In [86]:

```
# 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 label
```

Converting essay column to vector using Bag of Words (BoW).

In [87]:

```
# Code took from original Code provided.
vectorizer = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
vectorizer.fit(X_train['essay'].values)
print(len(vectorizer.get_feature_names()))
```

```
5000
```

In [88]:

```
# Code took from SAMPLE_SOLUTION notebook.
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, y_train.shape)
print(X_test_essay_bow.shape, y_test.shape)
```

```
(40000, 5000) (40000,)
(10000, 5000) (10000,)
```

In [89]:

```
essay_bow_columns = ['essay_'+i for i in vectorizer.get_feature_names()]
print(len(essay_bow_columns))
```

```
5000
```

In [90]:

```
import random
random.sample(essay_bow_columns, 10)
```

Out[90]:

```
['essay_varied',
 'essay_observe',
 'essay_notes',
 'essay_feel safe',
 'essay_smile face',
 'essay_fruits',
 'essay_school also',
 'essay_successfully',
 'essay_asking',
 'essay_listen stories']
```

Converting essay column to vector using TFIDF Vectorizer.

In [91]:

```
# Code took from original Code provided.
vectorizer = TfidfVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
vectorizer.fit(X_train['essay'].values)
print(len(vectorizer.get_feature_names()))
```

```
5000
```

In [92]:

```
# Code took from SAMPLE_SOLUTION notebook.
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, y_train.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
```

```
(40000, 5000) (40000,)
(10000, 5000) (10000,)
```

In [93]:

```
essay_tfidf_columns = ['essay_'+i for i in vectorizer.get_feature_names()]
print(len(essay_tfidf_columns))
```

```
5000
```

Converting essay column to vector using Average Word2Vec.

Creating function to return average word2vec vectors given sentences

In [94]:

```
 # Code took from original Code provided.
def avg_w2v(arr):
    """
    Returns array of vectors given array of sentences. Array of vectors are created by Aver
    words is taken from 'glove_vectors' file.
    """
    avg_w2v_vectors = []
    for sentence in tqdm(arr):
        vector = np.zeros(300)
        cnt_words = 0
        for word in sentence.split():
            if word in glove_words:
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_vectors.append(vector)
    return avg_w2v_vectors
```

In [95]:

```
X_train_essay_avgw2v = np.array(avg_w2v(X_train['essay'].values))
X_test_essay_avgw2v = np.array(avg_w2v(X_test['essay'].values))

print(X_train_essay_avgw2v.shape, y_train.shape)
print(X_test_essay_avgw2v.shape, y_test.shape)
```

```
100%|██████████████████████████████████████████████████| 40000/40
000 [00:13<00:00, 3026.43it/s]
100%|██████████████████████████████████████████████████| 10000/10
000 [00:03<00:00, 3127.76it/s]

(40000, 300) (40000,)
(10000, 300) (10000,)
```

Converting essay column to vector using TFIDF weighted Word2Vec.

Creating function to return tfidf weighted word2vec vectors given sentences and idf dictionary for words

In [96]:

```
# Code took from original Code provided.
def tfidf_w2v(arr, idf_dict):
    """
    Returns array of vectors given array of sentences and dictionary containing IDF values
    Array of vectors are created by TFIDF weighted Word2Vec method and vectors for words is
    """
    tfidf_w2v_vectors = []
    for sentence in tqdm(arr):
        vector = np.zeros(300)
        tf_idf_weight = 0;
        for word in sentence.split():
            if (word in glove_words) and (word in idf_dict):
                vec = model[word]
                tf_idf = idf_dict[word]/len(sentence.split())
                vector += (vec * tf_idf)
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors.append(vector)
    return tfidf_w2v_vectors
```

Getting idf values for the words in X_train.essay data

In [97]:

```
# Code took from original Code provided.
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_)))
```

In [98]:
```
X_train_essay_tfidfw2v = np.array(tfidf_w2v(X_train['essay'].values, dictionary))
X_test_essay_tfidfw2v = np.array(tfidf_w2v(X_test['essay'].values, dictionary))

print(X_train_essay_tfidfw2v.shape, y_train.shape)
print(X_test_essay_tfidfw2v.shape, y_test.shape)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 40000/4
0000 [01:45<00:00, 380.76it/s]
100%|██████████████████████████████████████████████████████████████████████████| 10000/1
0000 [00:26<00:00, 384.54it/s]

(40000, 300) (40000,)
(10000, 300) (10000,)
```

Converting `project_title` column to vector using Bag of Words (BoW).

In [99]:
```
# Code took from original Code provided.
vectorizer = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
vectorizer.fit(X_train['project_title'].values)
print(len(vectorizer.get_feature_names()))
```

```
2735
```

In [100]:
```
# Code took from SAMPLE_SOLUTION notebook.
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, y_train.shape)
print(X_test_title_bow.shape, y_test.shape)
```

```
(40000, 2735) (40000,)
(10000, 2735) (10000,)
```

```
In [101]: title_bow_columns = ['title_'+i for i in vectorizer.get_feature_names()]
          print(len(title_bow_columns))
```

```
2735
```

Converting `project_title` column to vector using TFIDF Vectorizer.

```
In [102]: # Code took from original Code provided.
          vectorizer = TfidfVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
          vectorizer.fit(X_train['project_title'].values)
          print(len(vectorizer.get_feature_names()))
```

```
2735
```

```
In [103]: # Code took from SAMPLE_SOLUTION notebook.
          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, y_train.shape)
          print(X_test_title_tfidf.shape, y_test.shape)
```

```
(40000, 2735) (40000,)
(10000, 2735) (10000,)
```

```
In [104]: title_tfidf_columns = ['title_'+i for i in vectorizer.get_feature_names()]
          print(len(title_tfidf_columns))
```

```
2735
```

Converting `project_title` column to vector using Average Word2Vec.

Can use avg_w2v function

In [105]:
```
X_train_title_avgw2v = np.array(avg_w2v(X_train['project_title'].values))
X_test_title_avgw2v = np.array(avg_w2v(X_test['project_title'].values))

print(X_train_title_avgw2v.shape, y_train.shape)
print(X_test_title_avgw2v.shape, y_test.shape)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 40000/400
00 [00:00<00:00, 54084.97it/s]
100%|██████████████████████████████████████████████████████████████████████████| 10000/100
00 [00:00<00:00, 42759.75it/s]

(40000, 300) (40000,)
(10000, 300) (10000,)
```

Converting `project_title` column to vector using TFIDF weighted Word2Vec.

Can use tfidf_w2v function but should calculate idf dictionary before using it

In [106]:
```
# Code took from original Code provided.
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_)))
```

In [107]:

```
X_train_title_tfidfw2v = np.array(tfidf_w2v(X_train['project_title'].values, dictionary))
X_test_title_tfidfw2v = np.array(tfidf_w2v(X_test['project_title'].values, dictionary))

print(X_train_title_tfidfw2v.shape, y_train.shape)
print(X_test_title_tfidfw2v.shape, y_test.shape)
```

```
100%|█████████████████████████████████████████████████████████████████████████| 40000/400
00 [00:01<00:00, 26261.38it/s]
100%|█████████████████████████████████████████████████████████████████████████| 10000/100
00 [00:00<00:00, 27042.47it/s]

(40000, 300) (40000,)
(10000, 300) (10000,)
```

In [108]:

```
bow_train = hstack((cat_num_train, X_train_essay_bow, X_train_title_bow)).tocsr()
bow_test = hstack((cat_num_test, X_test_essay_bow, X_test_title_bow)).tocsr()

tfidf_train = hstack((cat_num_train, X_train_essay_tfidf, X_train_title_tfidf)).tocsr()
tfidf_test = hstack((cat_num_test, X_test_essay_tfidf, X_test_title_tfidf)).tocsr()

avgw2v_train = np.hstack((cat_num_train.toarray(), X_train_essay_avgw2v, X_train_title_avgw
avgw2v_test = np.hstack((cat_num_test.toarray(), X_test_essay_avgw2v, X_test_title_avgw2v))

tfidfw2v_train = np.hstack((cat_num_train.toarray(), X_train_essay_tfidfw2v, X_train_title_
tfidfw2v_test = np.hstack((cat_num_test.toarray(), X_test_essay_tfidfw2v, X_test_title_tfid

print('='*30)
print(bow_train.shape)
print(bow_test.shape)
print('='*30)
print(tfidf_train.shape)
print(tfidf_test.shape)
print('='*30)
print(avgw2v_train.shape)
print(avgw2v_test.shape)
print('='*30)
print(tfidfw2v_train.shape)
print(tfidfw2v_test.shape)
print('='*30)
```

```
==============================
(40000, 7744)
(10000, 7744)
==============================
(40000, 7744)
(10000, 7744)
==============================
(40000, 609)
(10000, 609)
```

```
==============================
(40000, 609)
(10000, 609)
==============================
```

Writing several functions to reuse them later

Function to plot AUC values with respect to hyper-parameters given train data using K-fold validation

In [109]:

```
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
import math

# Code inside function took from SAMPLE_SOLUTION notebook
def auc_vs_K_plot(model, X_train, y_train, n_estimators, max_depth):
    """
    Plots the AUC results for different n_estimators and max_depth values on train and CV d
    Parameters:
    X_train, y_train - data which is used for K-fold validation and used to train tree base
    (RandomForestClassifier or XGBClassifier)
    max_depth - list of max_depth values on which we have to train the data and plot the re
    n_estimators - list of number of estimators on which we have to train the data and plot
    """
    parameters = {'n_estimators': n_estimators, 'max_depth': max_depth}
    clf = GridSearchCV(model, parameters, cv=3, scoring='roc_auc', return_train_score=True)
    clf.fit(X_train, y_train)

    train_auc= clf.cv_results_['mean_train_score']
    cv_auc = clf.cv_results_['mean_test_score']

    train_auc = train_auc.reshape((len(n_estimators), len(max_depth)))
    cv_auc = cv_auc.reshape((len(n_estimators), len(max_depth)))

    sns.heatmap(train_auc, vmin=0, vmax=1, annot=True, xticklabels=max_depth, yticklabels=n
    plt.xlabel("max depth")
    plt.ylabel("number of estimators")
    plt.title("Train score")
    plt.show()

    sns.heatmap(cv_auc, vmin=0, vmax=1, annot=True, xticklabels=max_depth, yticklabels=n_es
    plt.xlabel("max depth")
    plt.ylabel("number of estimators")
    plt.title("CV score")
    plt.show()
```

Function to plots ROC curves and confusion matrices for train and test data. Function returns AUC Values for train, test data

In [110]:

```
from sklearn.metrics import roc_curve, auc, precision_recall_curve
from IPython.display import Markdown, display

# Code inside function took from SAMPLE_SOLUTION notebook
def ROC_conf_mat(model, X_train, y_train, X_test, y_test, n_estimators, max_depth, plots =
    """
    Plots ROC Curve given best hyper parameter values, Train data and Test data using Tree
    And also plots confusion matrix for train data and test data taking a optimal threshold
    Returns Area Under ROC Curve for Train, Test data which can be taken as performance of
    """
    # Plotting ROC Curve code
    params = {'n_estimators': n_estimators, 'max_depth': max_depth}
    model.set_params(**params)
#     dt_model = DecisionTreeClassifier(max_depth = best_depth, min_samples_split = best_ms
    model.fit(X_train, y_train)

    y_train_pred = model.predict_proba(X_train)[:, 1]
    y_test_pred = model.predict_proba(X_test)[:, 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)

    result = {}

    result['train_auc'], result['test_auc'] = (auc(train_fpr, train_tpr), auc(test_fpr, tes
    result['model'] = model

    thr_train = tr_thresholds[np.argmax(train_tpr*(1-train_fpr))]
    thr_test = te_thresholds[np.argmax(test_tpr*(1-test_fpr))]

    train_predictions = []
    for i in y_train_pred:
        if i >= thr_train:
            train_predictions.append(1)
        else:
            train_predictions.append(0)

    test_predictions = []
    for i in y_test_pred:
        if i >= thr_test:
            test_predictions.append(1)
        else:
            test_predictions.append(0)

    # Collecting False Positive indices from the test data.
#     result['false_positive'] = [i for i in range(len(y_test)) if test_predictions[i]==1 a

    if(plots):
        display(Markdown(f"**Analysis for max_depth = {max_depth} and n_estimators = {n_est

        plt.plot(train_fpr, train_tpr, label="train AUC ="+str(np.round(result['train_auc']
        plt.plot(test_fpr, test_tpr, label="test AUC ="+str(np.round(result['test_auc'], 3)
        plt.legend()
        plt.xlabel("False Positive rate")
        plt.ylabel("True Positive rate")
        plt.title("ROC Curves for Train and Test data")
        plt.grid()
        plt.show()

        # Printing confusion matrices code
        print(f"\nConfusion matrix for Train data with {thr_train} as threshold:")

        ax = sns.heatmap(confusion_matrix(y_train, train_predictions), annot=True, fmt='g')
        ax.set_yticklabels(['Rejected', 'Accepted'])
        ax.set_xticklabels(['Rejected', 'Accepted'])
        plt.xlabel("Predicted")
        plt.ylabel("Actual")
        plt.title('Confusion matrix for Train')
        plt.show()

        print(f"\nConfusion matrix for Test data with {thr_test} as threshold:")
        ax = sns.heatmap(confusion_matrix(y_test, test_predictions), annot=True, fmt='g')
        ax.set_yticklabels(['Rejected', 'Accepted'])
        ax.set_xticklabels(['Rejected', 'Accepted'])
        plt.xlabel("Predicted")
        plt.ylabel("Actual")
        plt.title('Confusion matrix for Test')
        plt.show()

    return result
```

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

2.4.1 Applying Random Forests on BOW, SET 1

In [113]:

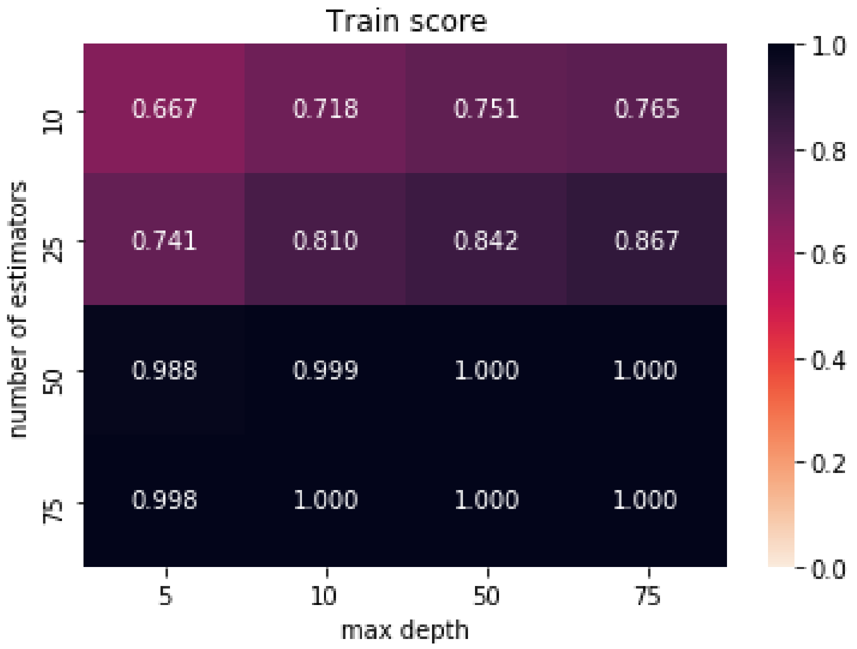
```
from sklearn.ensemble import RandomForestClassifier
```

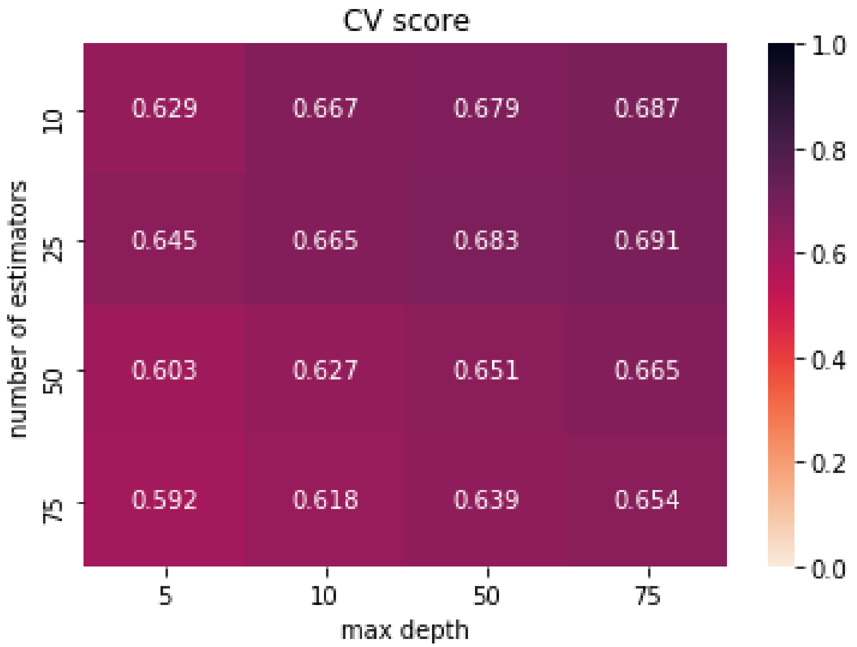
In [114]:

```
from xgboost import XGBClassifier
```

Limiting range for n_estimators and max_depth to [10, 75] and [5, 75] respectively. Because if we take more than these values, the models seems to be overfitting

In [115]:
```
# Please write all the code with proper documentation
n_est = [10, 25, 50, 75]
max_d = [5, 10, 50, 75]
auc_vs_K_plot(RandomForestClassifier(), bow_train, y_train, n_est, max_d)
```

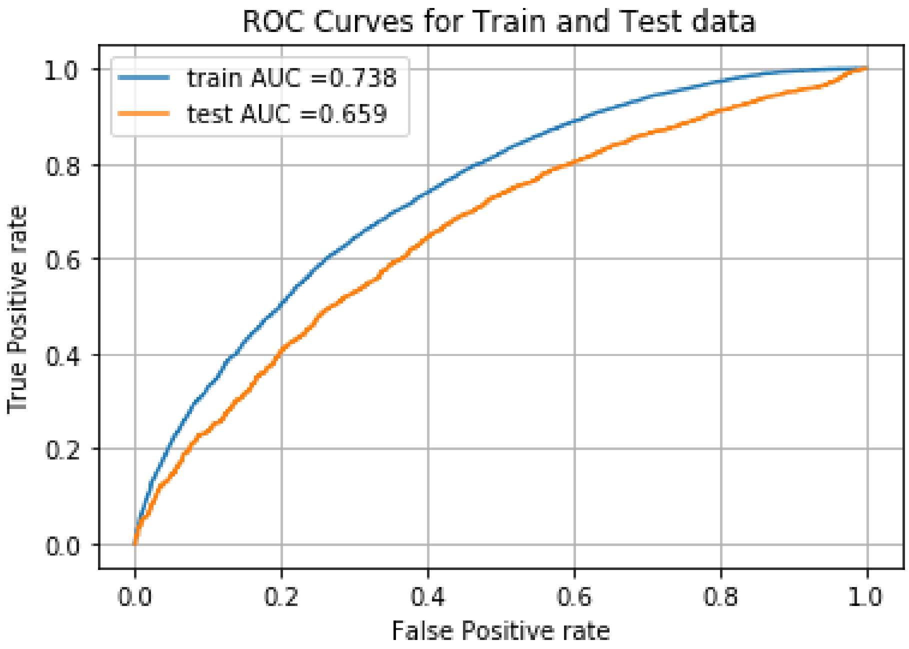




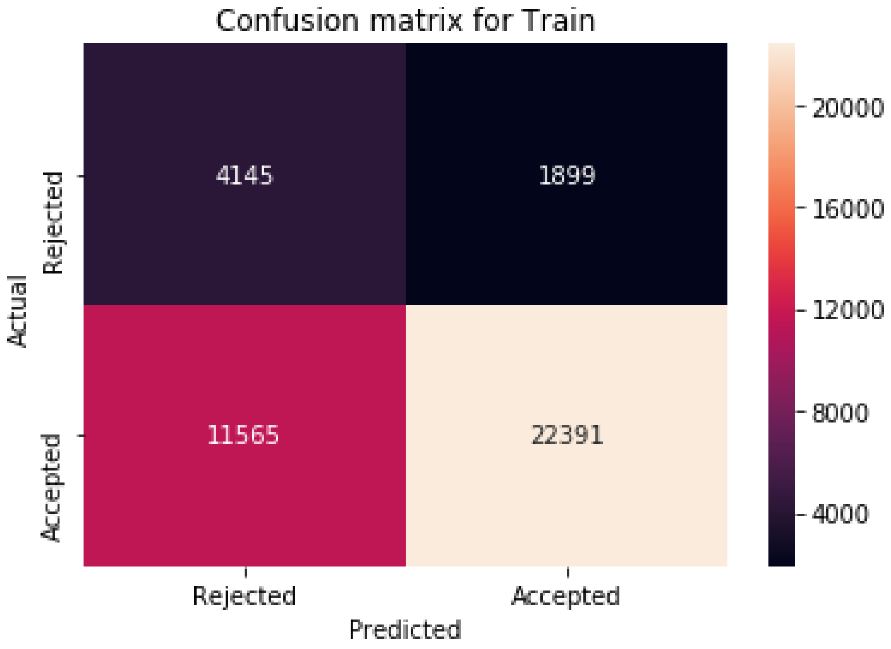
Taking (10, 10) as best n_estimators and max_depth. The best hyper-parameters obtained from above heatmaps seems to be overfitting a lot. So taking (10, 10) as best after some trails

```
In [131]: bow_result = {}
          bow_result['10,10'] = ROC_conf_mat(RandomForestClassifier(), bow_train, y_train, bow_test,
```

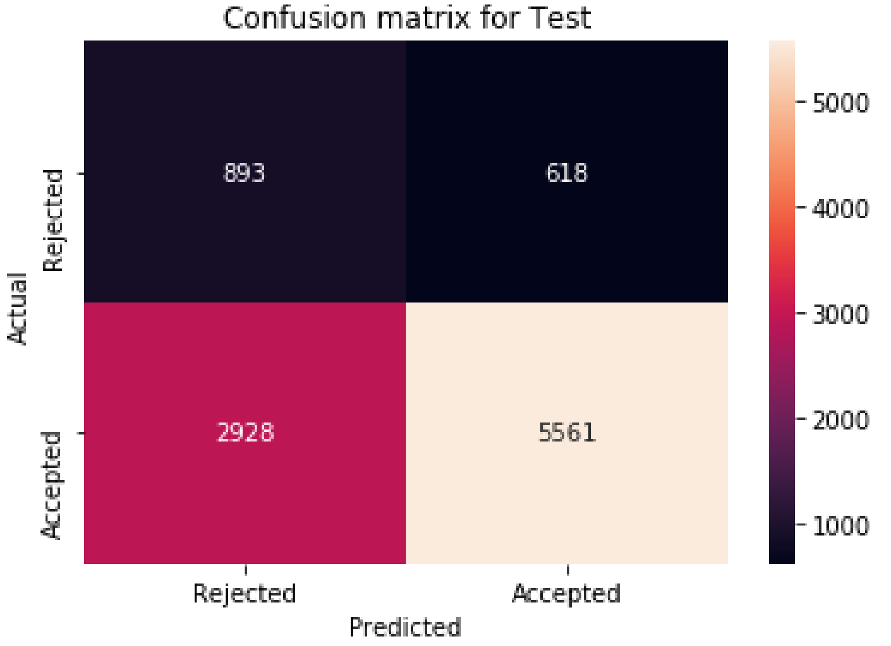
Analysis for max_depth = 10 and n_estimators = 10



```
Confusion matrix for Train data with 0.8484104749557358 as threshold:
```

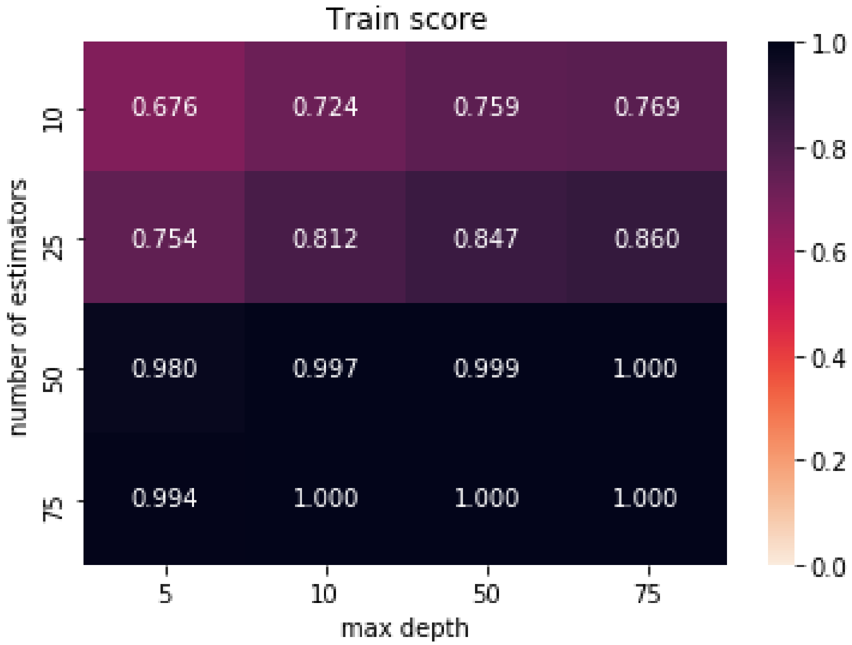


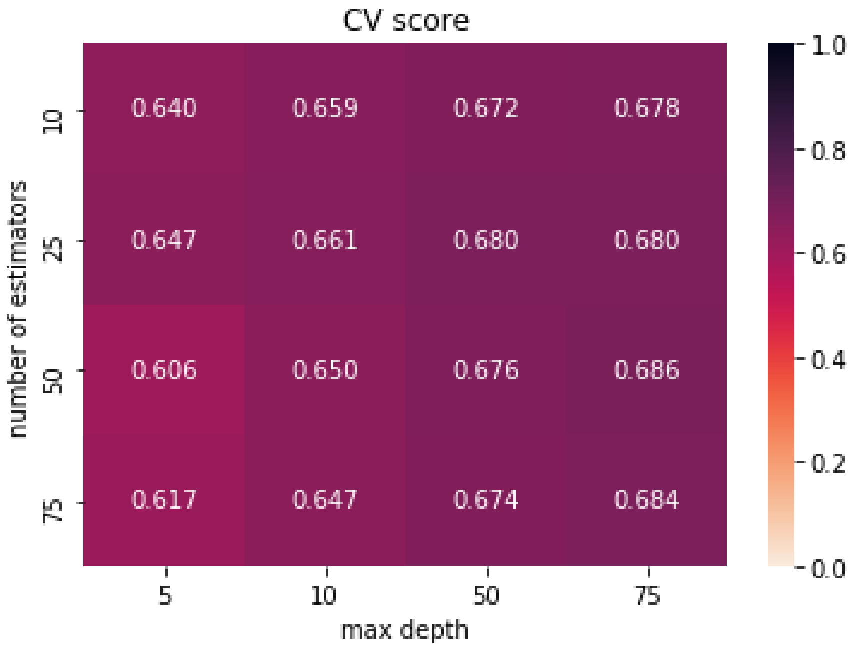
```
Confusion matrix for Test data with 0.8481002947758981 as threshold:
```



2.4.2 Applying Random Forests on TFIDF, SET 2

In [116]:
```
# Please write all the code with proper documentation
n_est = [10, 25, 50, 75]
max_d = [5, 10, 50, 75]
auc_vs_K_plot(RandomForestClassifier(), tfidf_train, y_train, n_est, max_d)
```

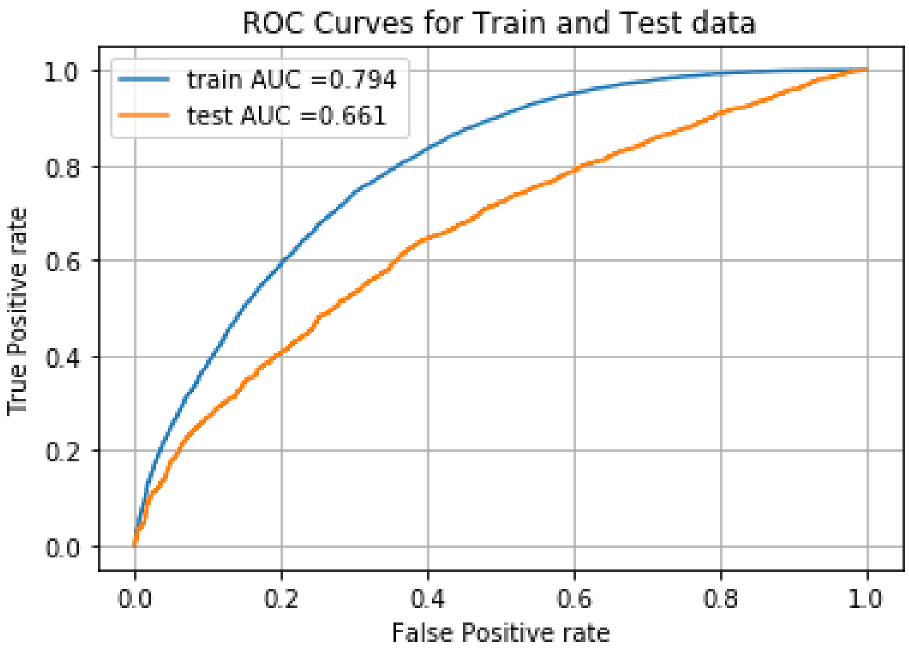




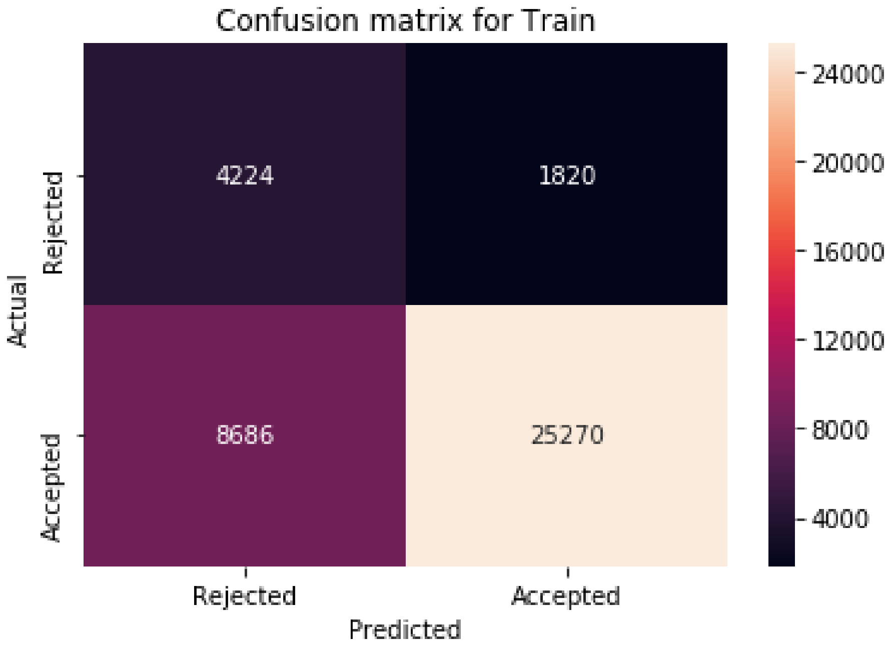
Taking (25, 10) as best n_estimators and max_depth after some trails

```
In [149]: tfidf_result = {}
          tfidf_result['25,10'] = ROC_conf_mat(RandomForestClassifier(), tfidf_train, y_train, tfidf_
```

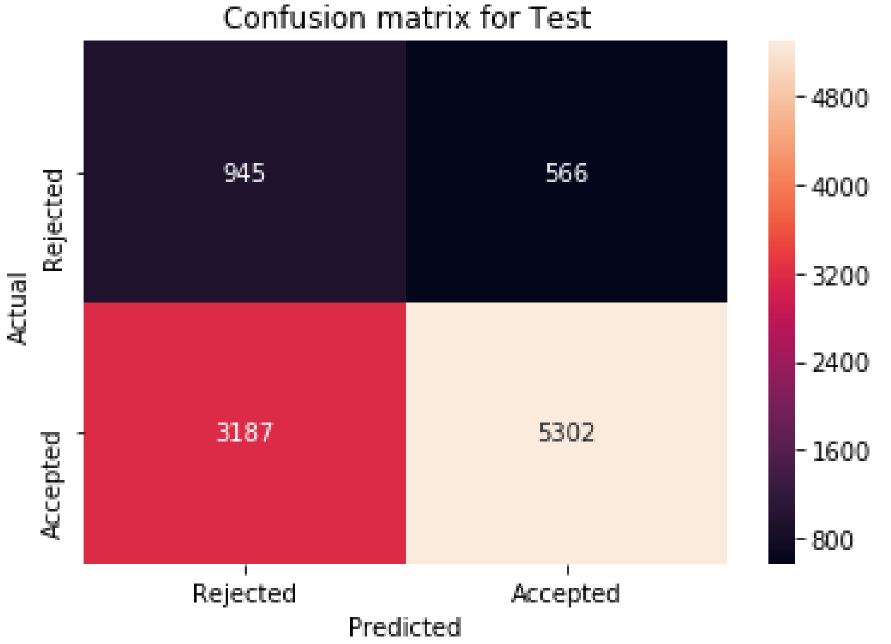
Analysis for max_depth = 10 and n_estimators = 25



```
Confusion matrix for Train data with 0.8460455260964046 as threshold:
```

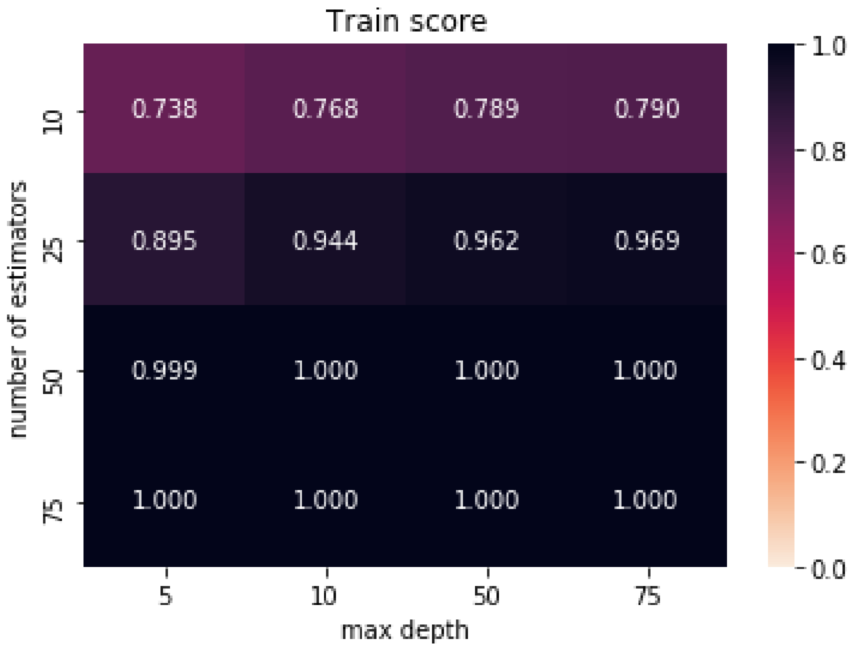


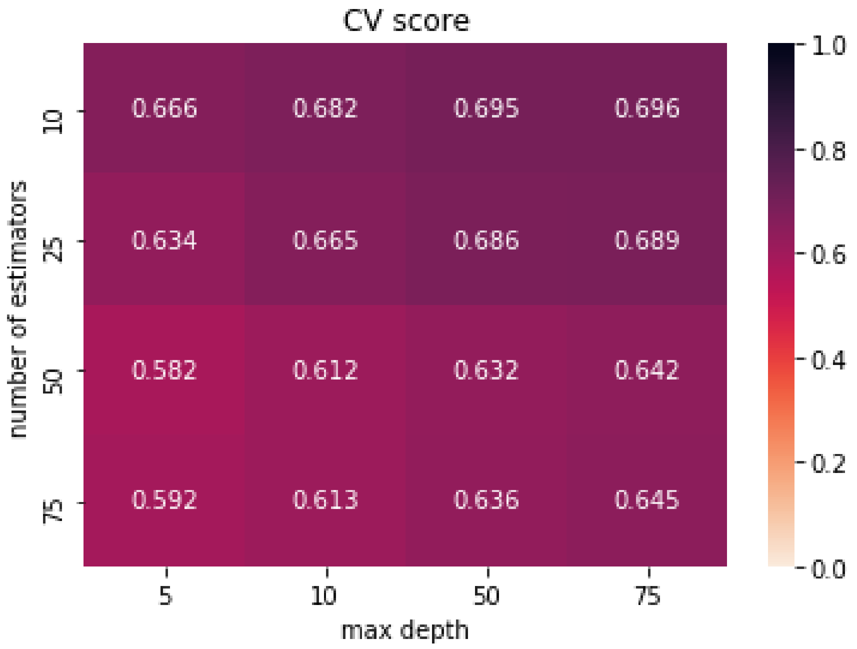
```
Confusion matrix for Test data with 0.8503214069898708 as threshold:
```



2.4.3 Applying Random Forests on AVG W2V, SET 3

In [117]:
```
# Please write all the code with proper documentation
n_est = [10, 25, 50, 75]
max_d = [5, 10, 50, 75]
auc_vs_K_plot(RandomForestClassifier(), avgw2v_train, y_train, n_est, max_d)
```

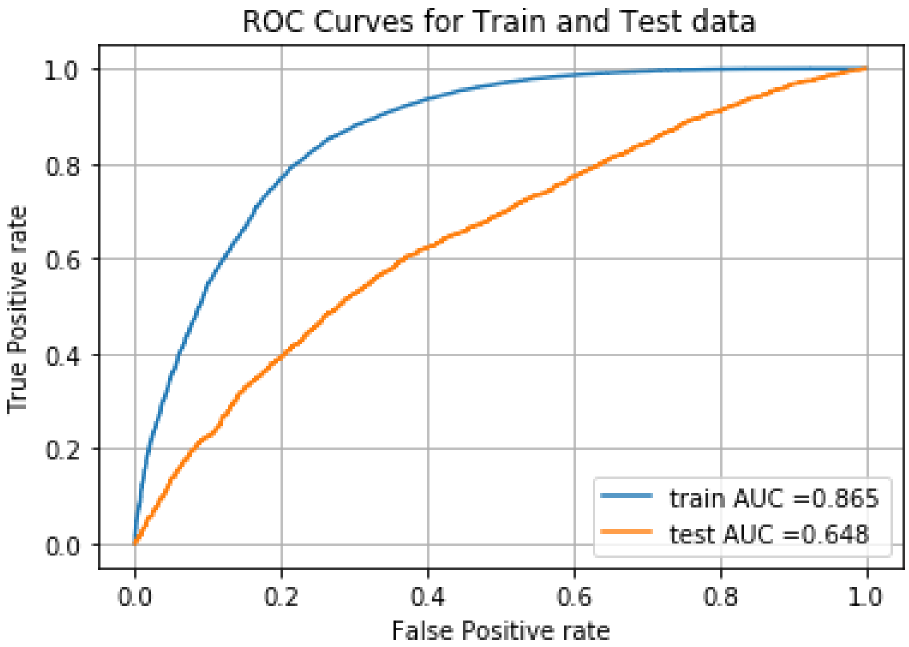




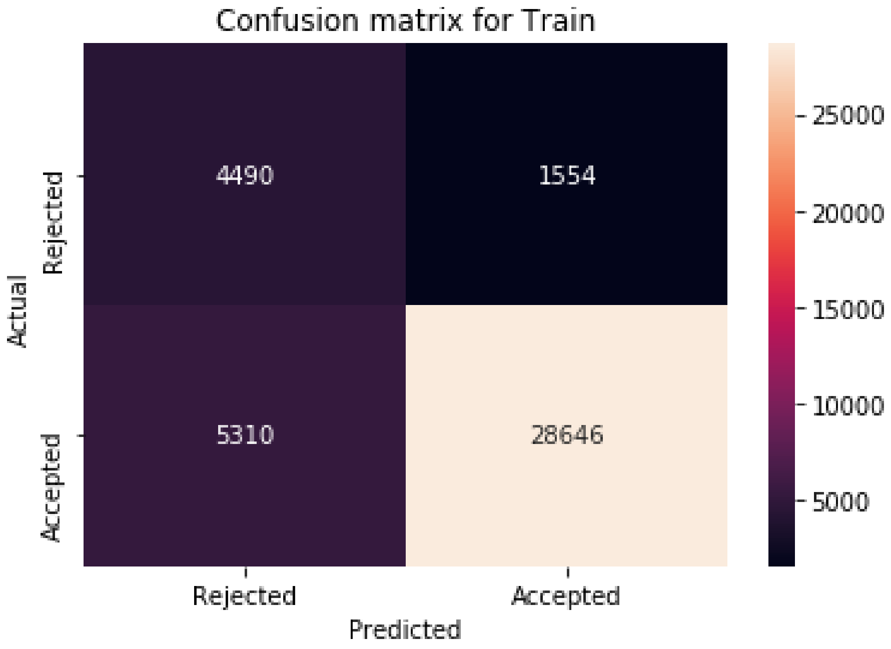
Taking (10, 10) as best n_estimators and max_depth after some trails. These models are averfitting a lot. very hard to find best hyper-parameters

In [159]:
```
avgw2v_result = {}
avgw2v_result['10,10'] = ROC_conf_mat(RandomForestClassifier(), avgw2v_train, y_train, avgw
```

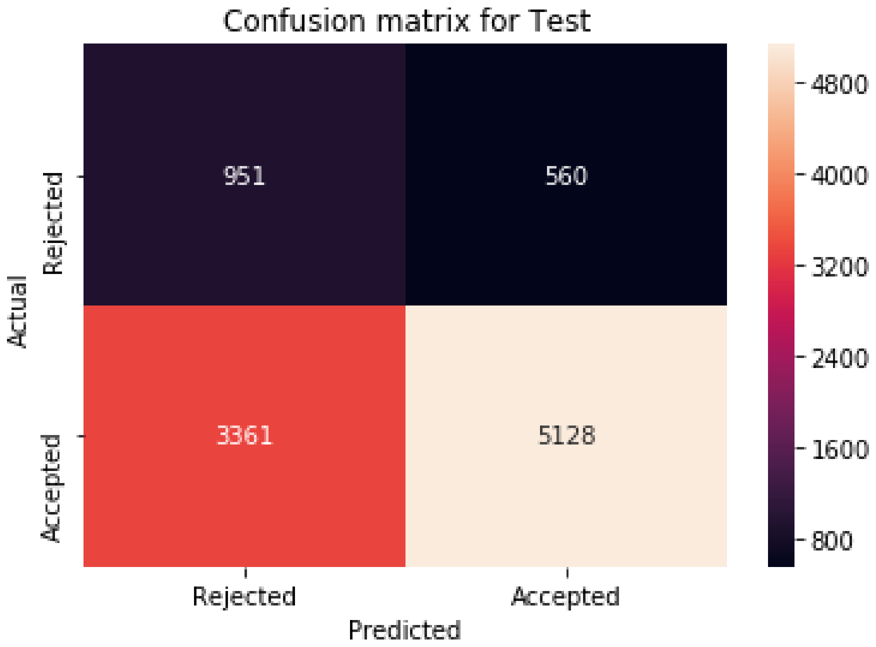
Analysis for max_depth = 10 and n_estimators = 10



```
Confusion matrix for Train data with 0.8284226863331641 as threshold:
```

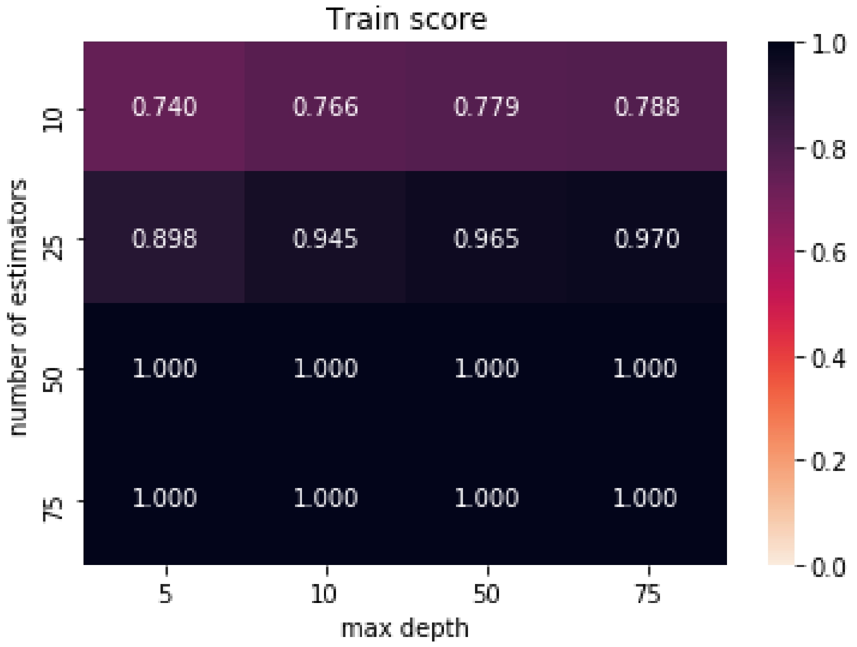


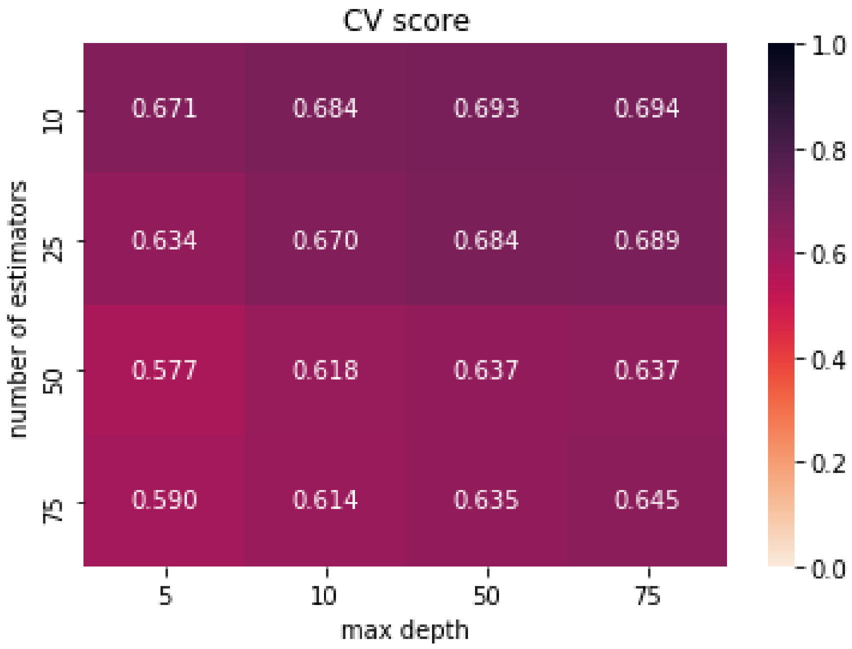
```
Confusion matrix for Test data with 0.8605741170631406 as threshold:
```



2.4.4 Applying Random Forests on TFIDF W2V, SET 4

In [118]:
```
# Please write all the code with proper documentation
n_est = [10, 25, 50, 75]
max_d = [5, 10, 50, 75]
auc_vs_K_plot(RandomForestClassifier(), tfidfw2v_train, y_train, n_est, max_d)
```

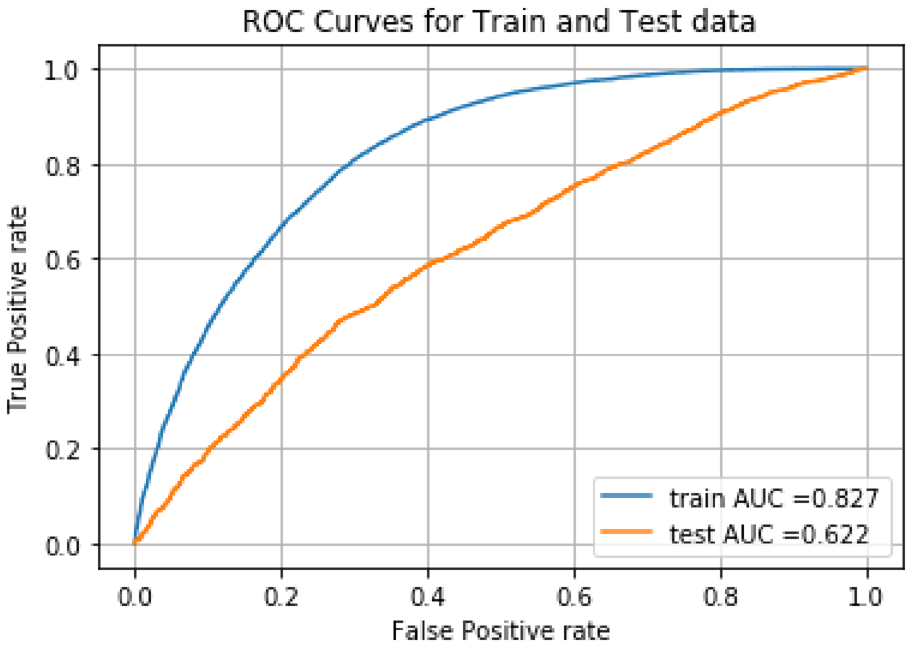




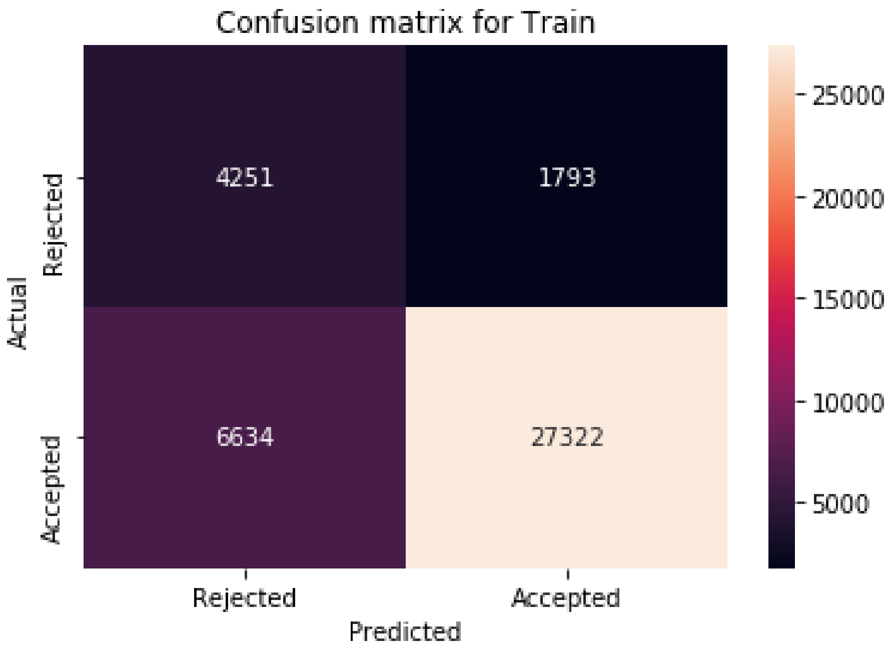
Taking (5, 10) as best n_estimators and max_depth

In [162]:
```
tfidfw2v_result = {}
tfidfw2v_result['5,10'] = ROC_conf_mat(RandomForestClassifier(), tfidfw2v_train, y_train, t
```

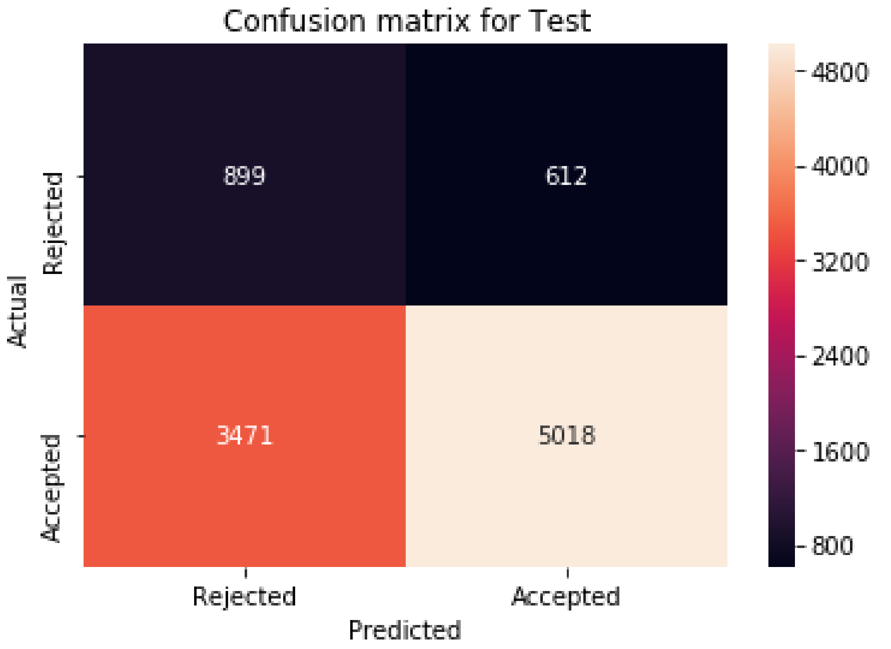
Analysis for max_depth = 10 and n_estimators = 5



```
Confusion matrix for Train data with 0.8425050127592622 as threshold:
```



```
Confusion matrix for Test data with 0.8705581559194313 as threshold:
```

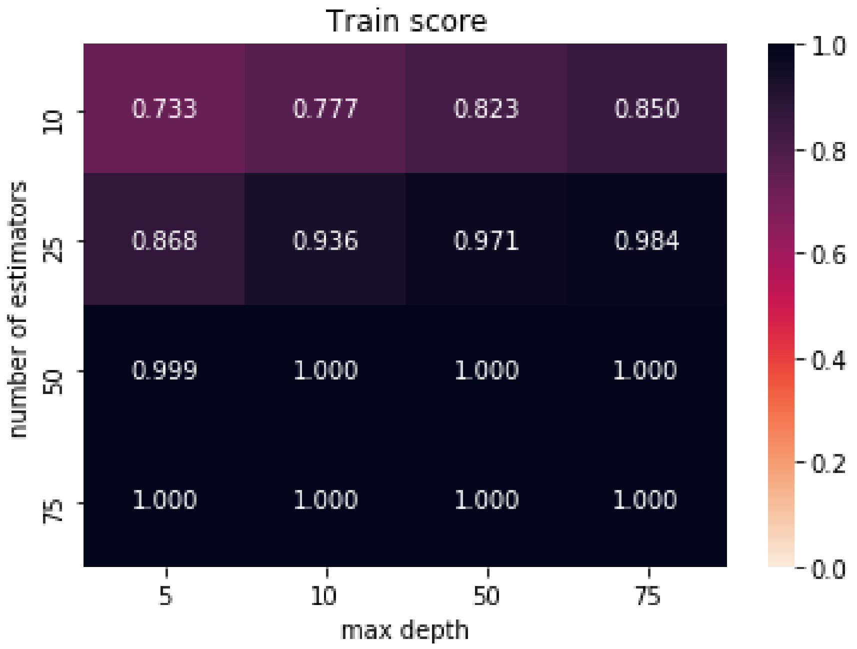


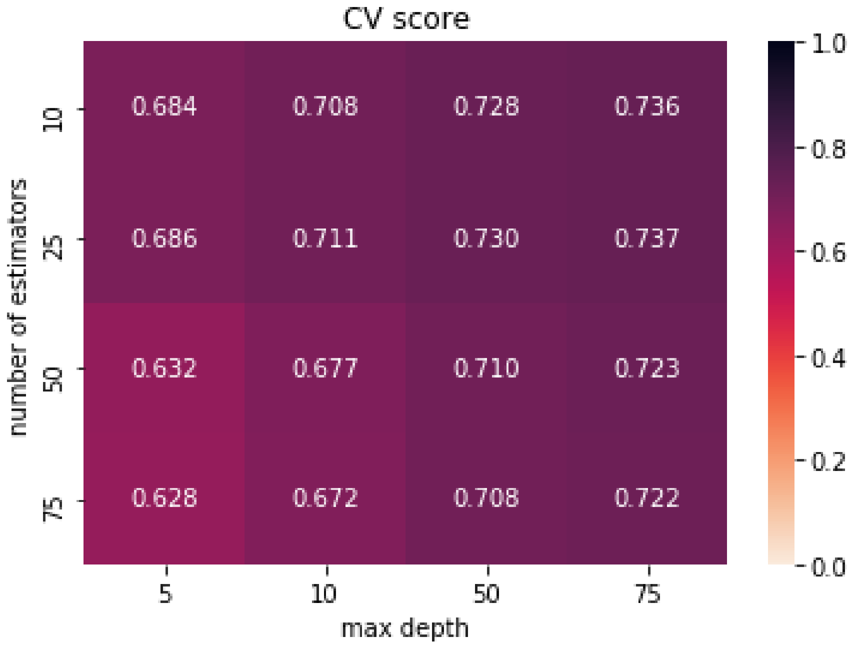
2.5 Applying GBDT

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 [119]:
```
# Please write all the code with proper documentation
n_est = [10, 25, 50, 75]
max_d = [5, 10, 50, 75]
auc_vs_K_plot(XGBClassifier(), bow_train, y_train, n_est, max_d)
```

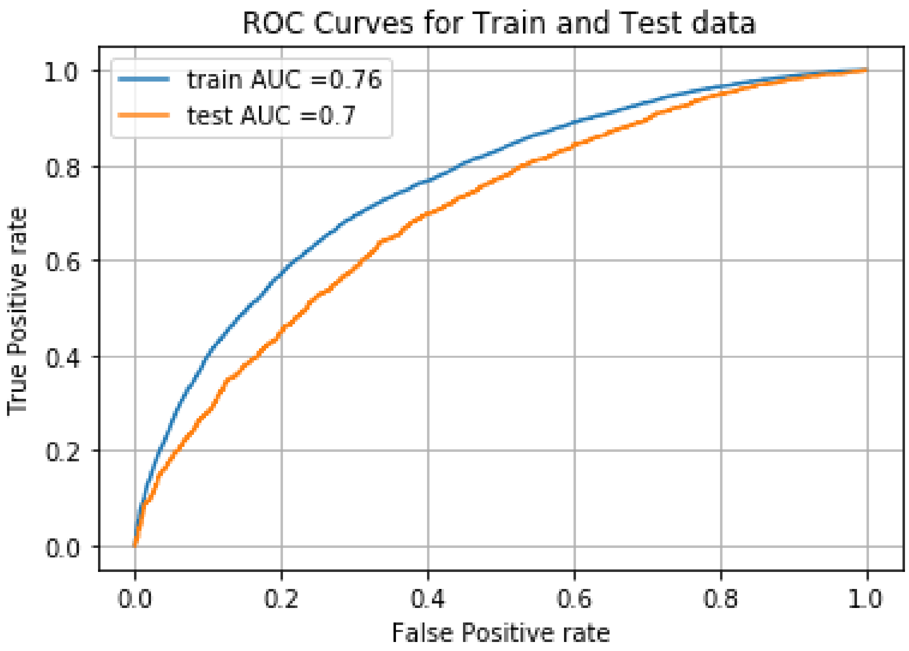




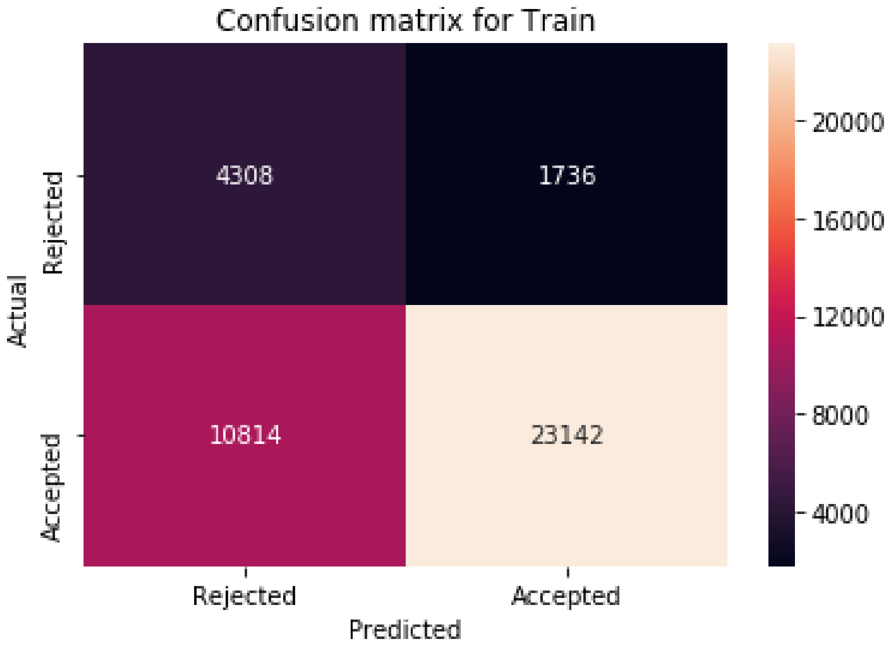
Taking (25, 5) as best n_estimators and max_depth

In [174]:
```
bow_xgb_result = {}
bow_xgb_result['25,5'] = ROC_conf_mat(XGBClassifier(), bow_train, y_train, bow_test, y_test
```

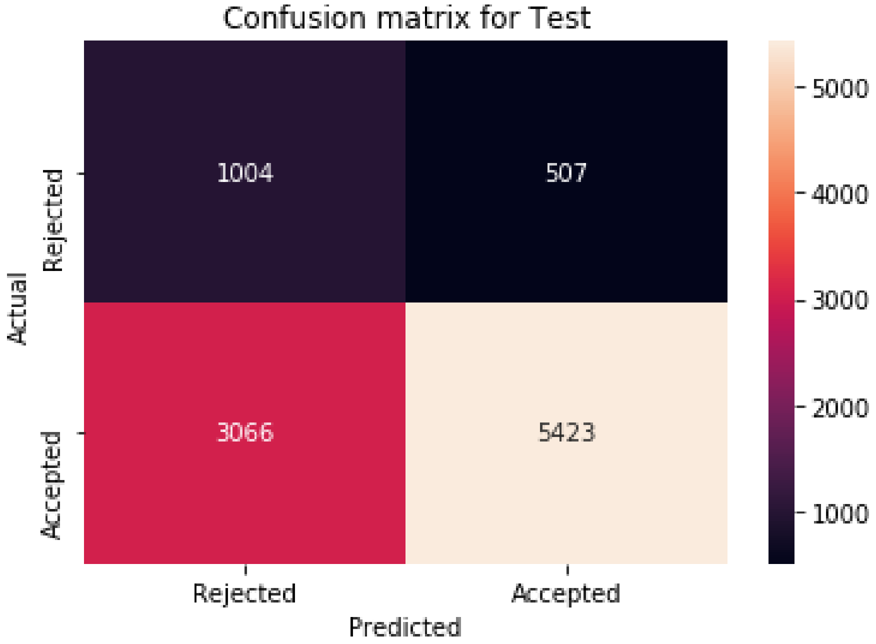
Analysis for max_depth = 5 and n_estimators = 25



```
Confusion matrix for Train data with 0.8199276924133301 as threshold:
```

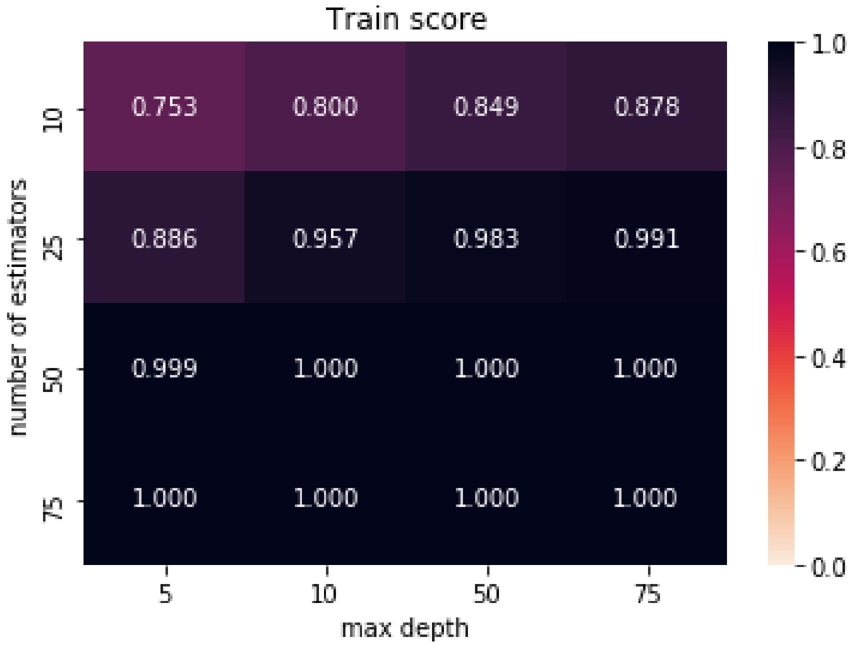


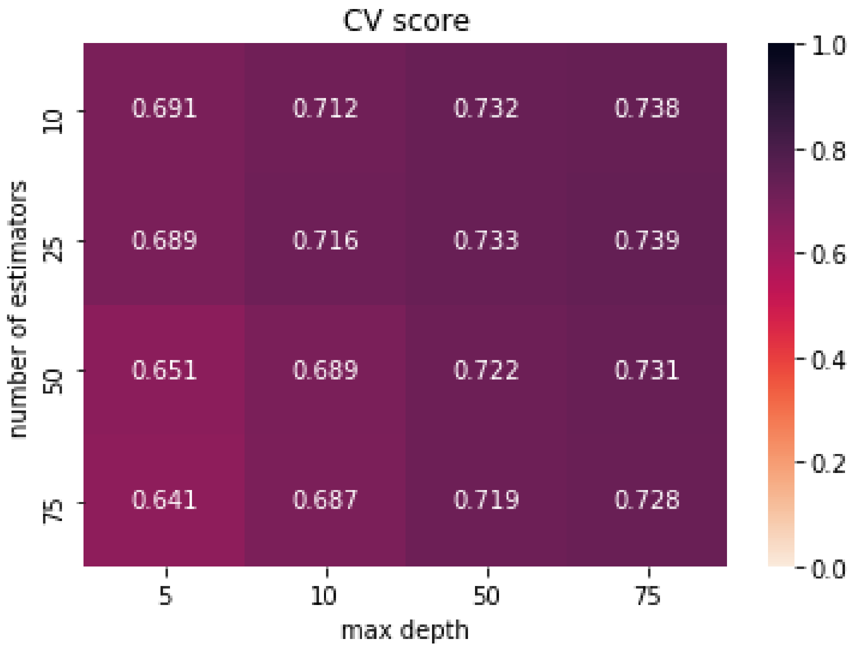
```
Confusion matrix for Test data with 0.8272839784622192 as threshold:
```



2.5.2 Applying XGBOOST on TFIDF, SET 2

In [120]:
```
# Please write all the code with proper documentation
n_est = [10, 25, 50, 75]
max_d = [5, 10, 50, 75]
auc_vs_K_plot(XGBClassifier(), tfidf_train, y_train, n_est, max_d)
```



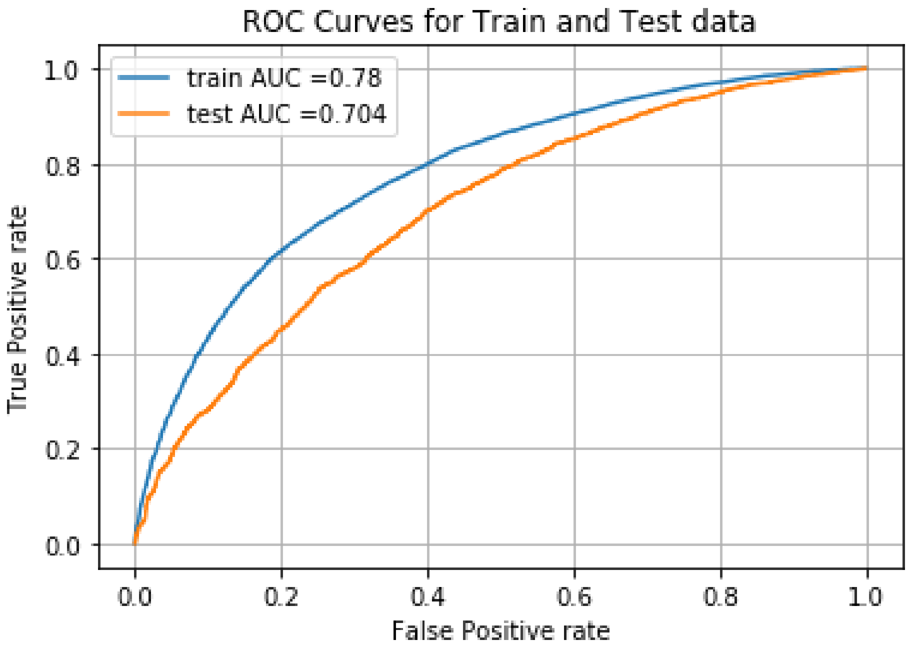


Taking (25, 5) as best n_estimators and max_depth

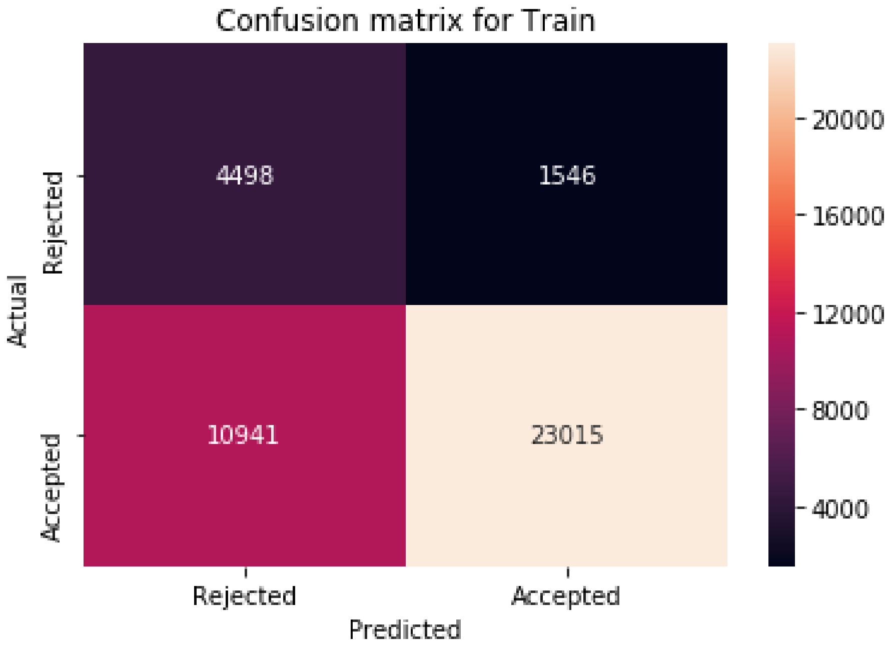
In [175]:

```
tfidf_xgb_result = {}
tfidf_xgb_result['25,5'] = ROC_conf_mat(XGBClassifier(), tfidf_train, y_train, tfidf_test,
```

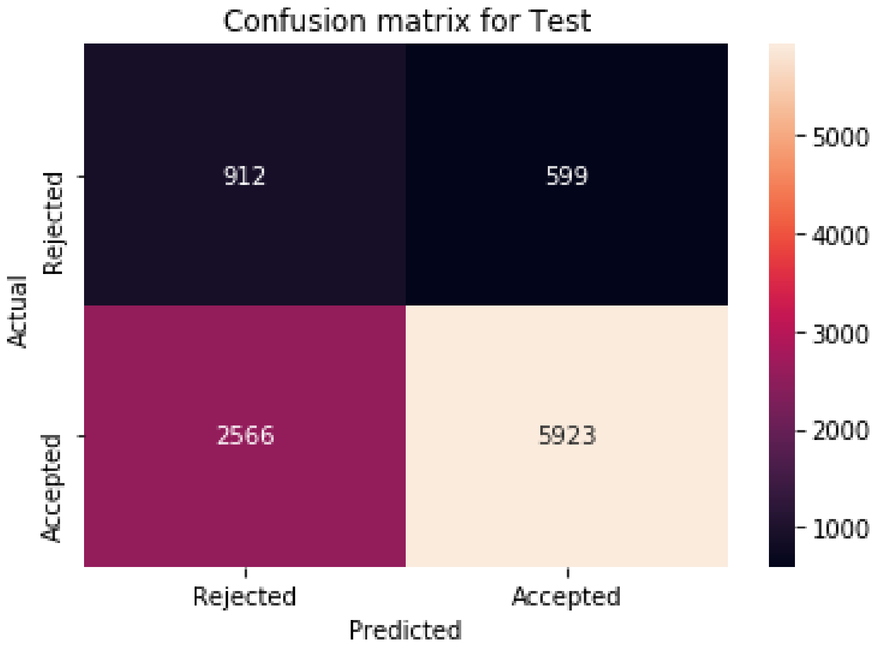
Analysis for max_depth = 5 and n_estimators = 25



```
Confusion matrix for Train data with 0.8252877593040466 as threshold:
```

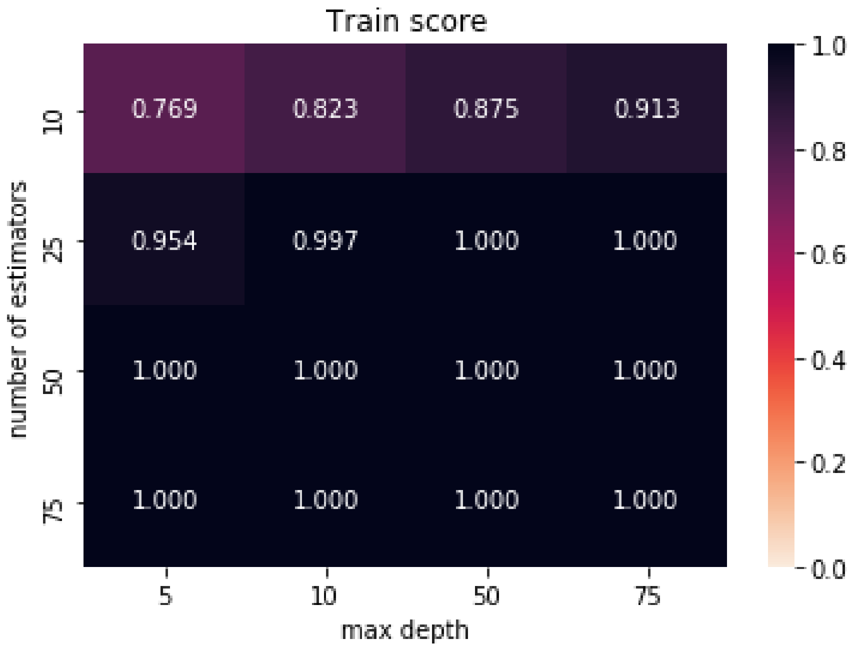


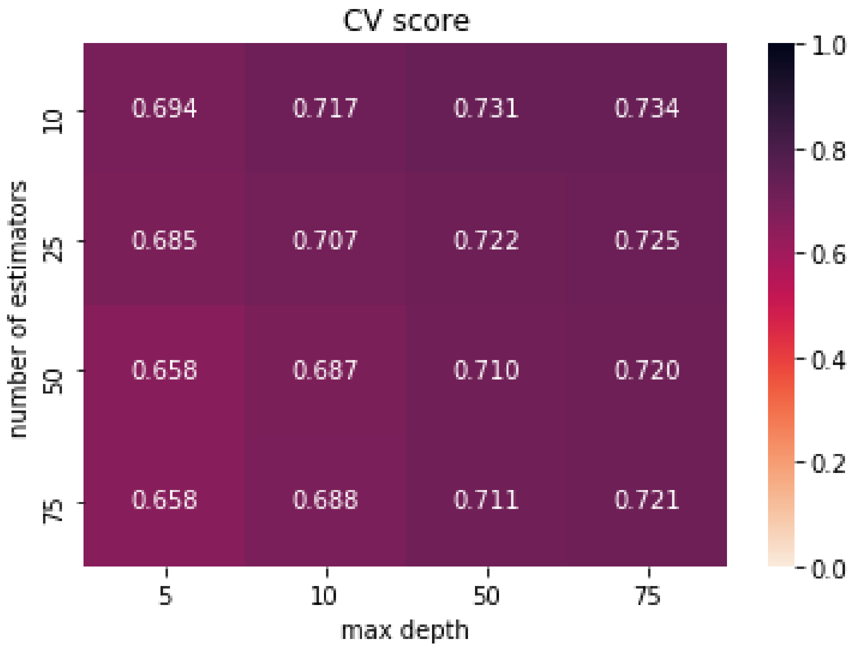
```
Confusion matrix for Test data with 0.8161970376968384 as threshold:
```



2.5.3 Applying XGBOOST on AVG W2V, SET 3

In [121]:
```
# Please write all the code with proper documentation
n_est = [10, 25, 50, 75]
max_d = [5, 10, 50, 75]
auc_vs_K_plot(XGBClassifier(), avgw2v_train, y_train, n_est, max_d)
```

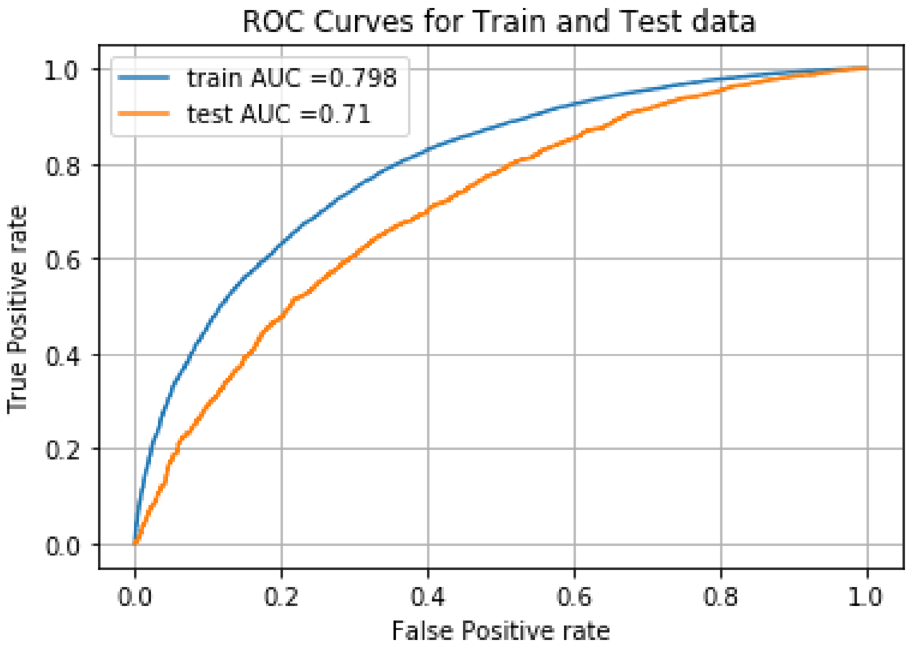




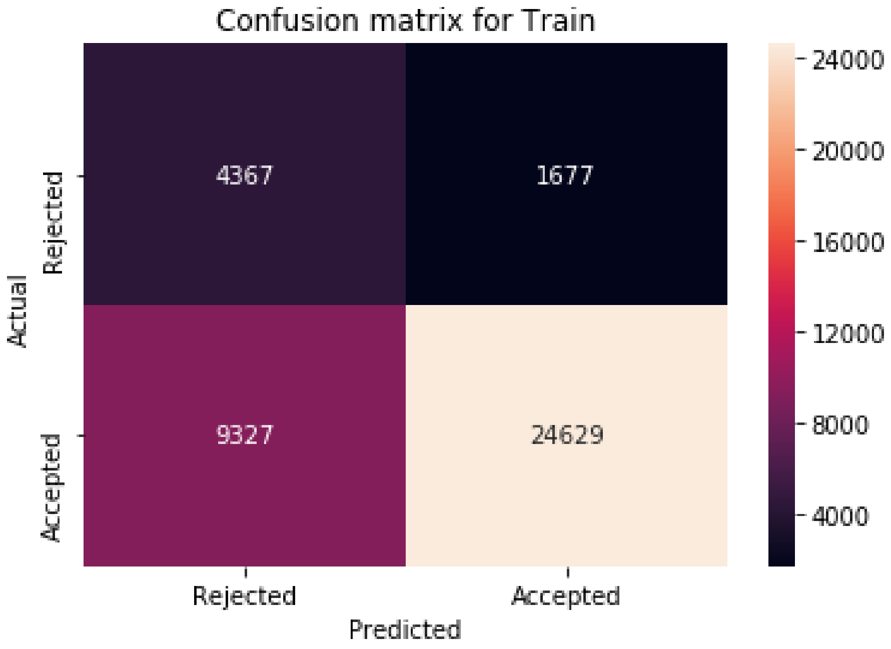
Taking (25, 5) as best n_estimators and max_depth

```
In [176]: avgw2v_xgb_result = {}
          avgw2v_xgb_result['25,5'] = ROC_conf_mat(XGBClassifier(), avgw2v_train, y_train, avgw2v_tes
```

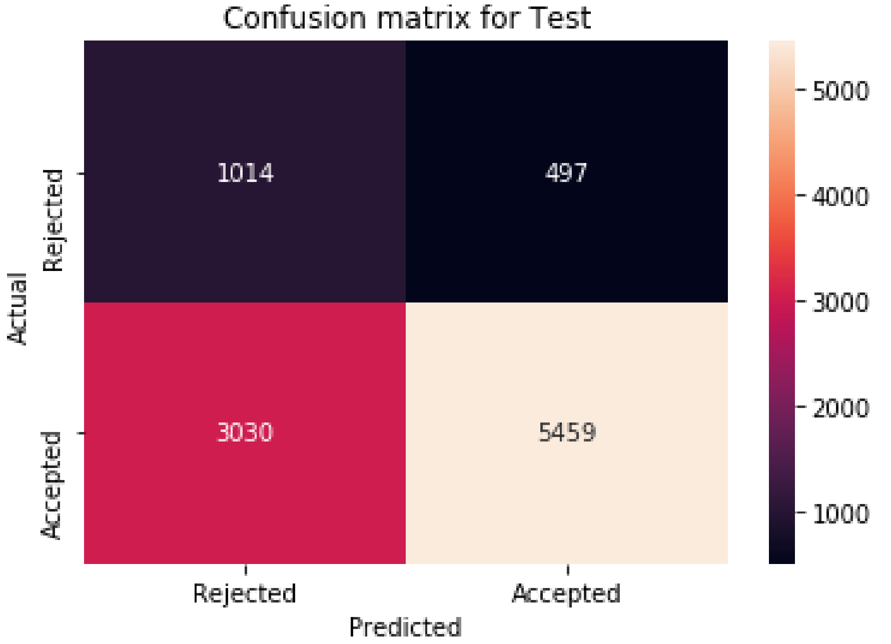
Analysis for max_depth = 5 and n_estimators = 25



```
Confusion matrix for Train data with 0.8160549998283386 as threshold:
```

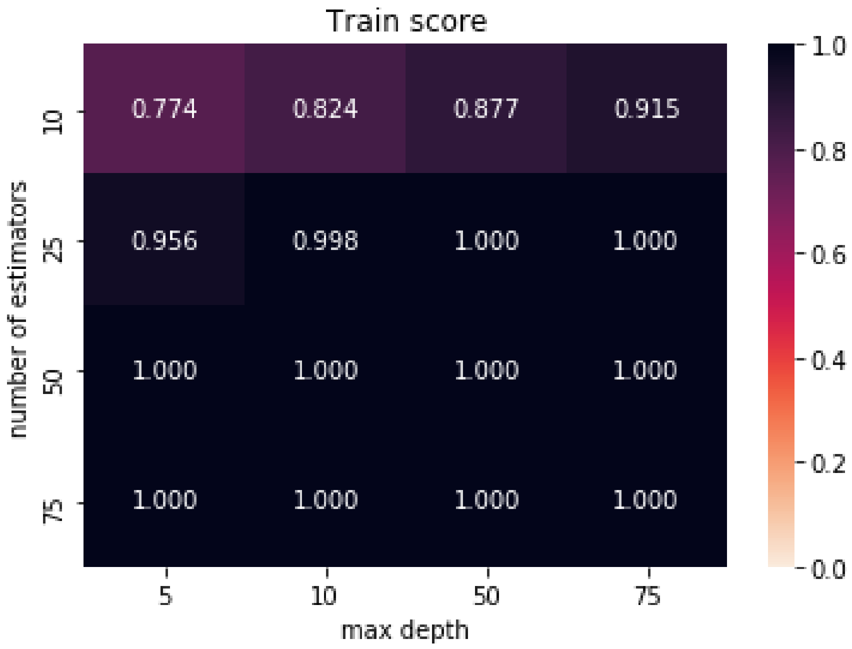


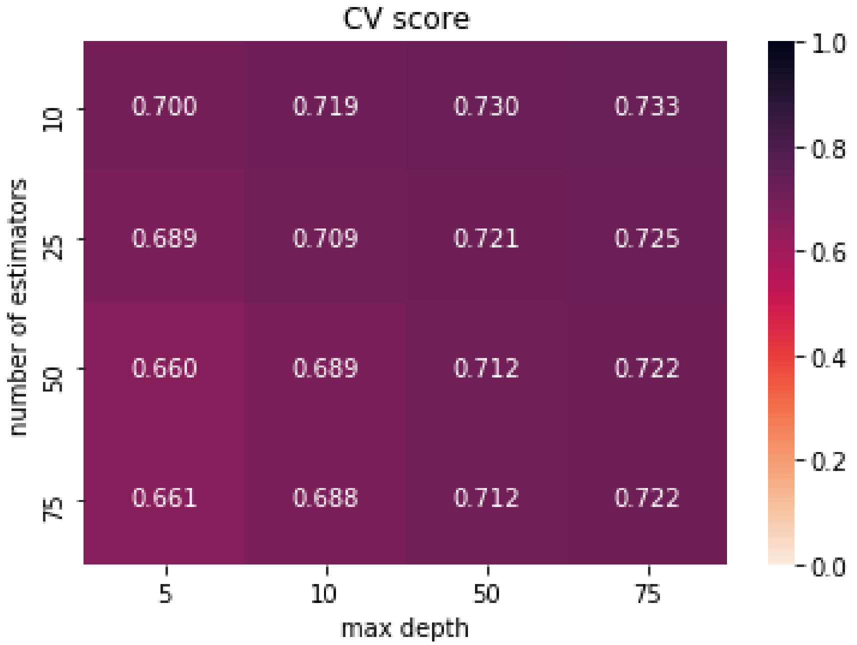
```
Confusion matrix for Test data with 0.8277551531791687 as threshold:
```



2.5.4 Applying XGBOOST on TFIDF W2V, SET 4

```
In [122]: # Please write all the code with proper documentation
          n_est = [10, 25, 50, 75]
          max_d = [5, 10, 50, 75]
          auc_vs_K_plot(XGBClassifier(), tfidfw2v_train, y_train, n_est, max_d)
```



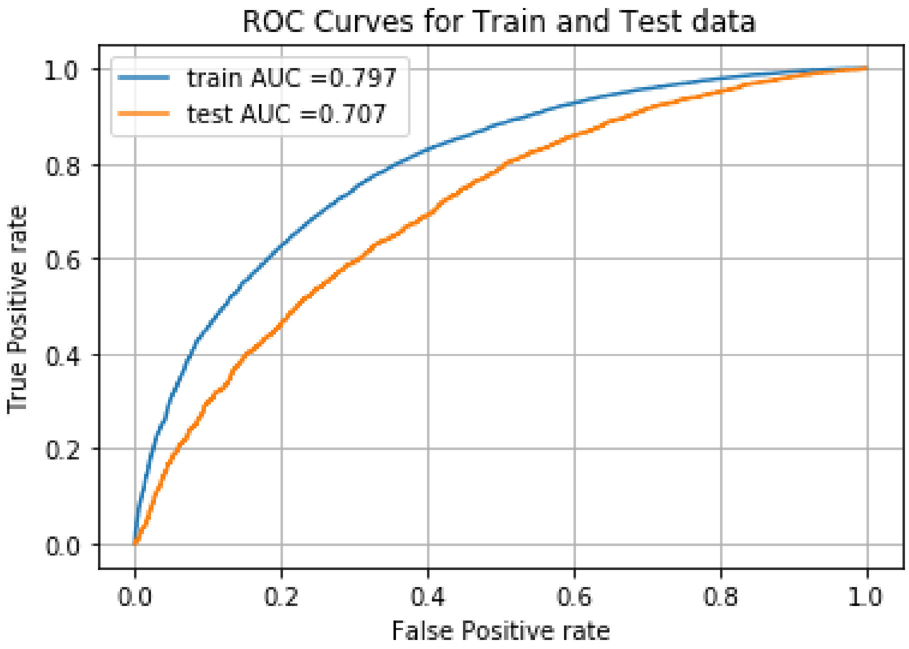


Taking (25, 5) as best n_estimators and max_depth

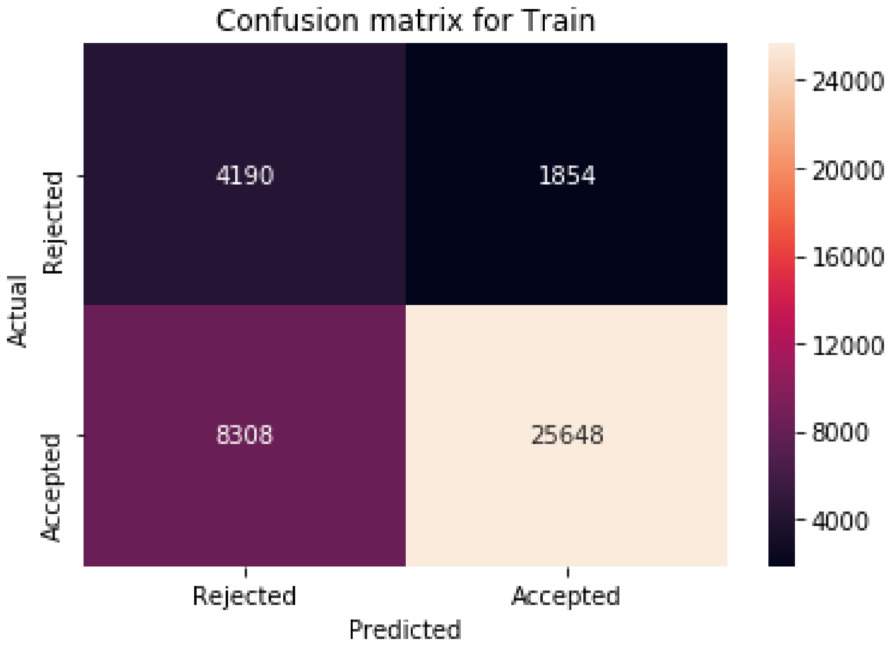
In [177]:

```
tfidfw2v_xgb_result = {}
tfidfw2v_xgb_result['25,5'] = ROC_conf_mat(XGBClassifier(), tfidfw2v_train, y_train, tfidfw
```

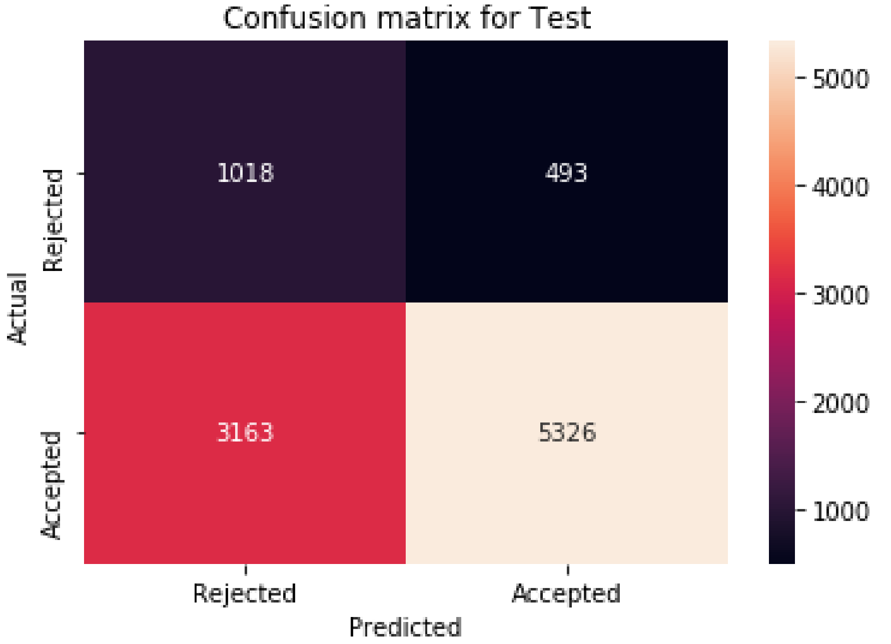
Analysis for max_depth = 5 and n_estimators = 25



```
Confusion matrix for Train data with 0.8078930974006653 as threshold:
```



```
Confusion matrix for Test data with 0.8295511603355408 as threshold:
```



3. Conclusion

```
In [190]: tfidfw2v_xgb_result
```

```
Out[190]: {'25,5': {'train_auc': 0.7972471835315513,
            'test_auc': 0.7074215793257269,
            'model': 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=5, min_child_weight=1, missing=None,
                   n_estimators=25, 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)}}
```

In [193]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable
table = PrettyTable()
table.field_names = ['Vectorizer', 'Model', 'n_estimators', 'max_depth', 'Train AUC', 'Test
table.add_row(['Bag of Words', 'Random Forest', 10, 10, np.round(bow_result['10,10']['train
               np.round(bow_result['10,10']['test_auc'], 3)])
table.add_row(['TfIdf', 'Random Forest', 25, 10, np.round(tfidf_result['25,10']['train_auc'
               np.round(tfidf_result['25,10']['test_auc'], 3)])
table.add_row(['Average Word2Vec', 'Random Forest', 10, 10, np.round(avgw2v_result['10,10']
               np.round(avgw2v_result['10,10']['test_auc'], 3)])
table.add_row(['TfIdf Word2Vec', 'Random Forest', 5, 10, np.round(tfidfw2v_result['5,10']['
               np.round(tfidfw2v_result['5,10']['test_auc'], 3)])
table.add_row(['Bag of Words', 'GBDT', 25, 5, np.round(bow_xgb_result['25,5']['train_auc'],
               np.round(bow_xgb_result['25,5']['test_auc'], 3)])
table.add_row(['TfIdf', 'GBDT', 25, 5, np.round(tfidf_xgb_result['25,5']['train_auc'], 3),\
               np.round(tfidf_xgb_result['25,5']['test_auc'], 3)])
table.add_row(['Average Word2Vec', 'GDBT', 25, 5, np.round(avgw2v_xgb_result['25,5']['train
               np.round(avgw2v_xgb_result['25,5']['test_auc'], 3)])
table.add_row(['TfIdf Word2Vec', 'GDBT', 25, 5, np.round(tfidfw2v_xgb_result['25,5']['train
               np.round(tfidfw2v_xgb_result['25,5']['test_auc'], 3)])
print(table)
```

```
+------------------+---------------+--------------+-----------+-----------+----------+
|    Vectorizer    |     Model     | n_estimators | max_depth | Train AUC | Test AUC |
+------------------+---------------+--------------+-----------+-----------+----------+
|   Bag of Words   | Random Forest |      10      |     10    |   0.738   |  0.659   |
|      TfIdf       | Random Forest |      25      |     10    |   0.794   |  0.661   |
| Average Word2Vec | Random Forest |      10      |     10    |   0.865   |  0.648   |
|  TfIdf Word2Vec  | Random Forest |      5       |     10    |   0.827   |  0.622   |
|   Bag of Words   |      GBDT     |      25      |     5     |    0.76   |   0.7    |
|      TfIdf       |      GBDT     |      25      |     5     |    0.78   |  0.704   |
| Average Word2Vec |      GDBT     |      25      |     5     |   0.798   |   0.71   |
|  TfIdf Word2Vec  |      GDBT     |      25      |     5     |   0.797   |  0.707   |
+------------------+---------------+--------------+-----------+-----------+----------+
```

Conclusion:

- **GBDT Models did well compared to Random Forest models. And these models are highly overfit to the train data.**
- **Random Forest models are highly overfit. Training time is very high for both models And training time for GBDT is very high when compared to Random Forest**
- **GDBT has high performance than other models i.e. Naive Bayes, KNN. But Still SVM has better performance than GDBT. May be High dimentionality is reason for lack of performance in Decision tree models.**