DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

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

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

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

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

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

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

In [5]:

```
## import all the modules
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import scipy
import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
from scipy import sparse
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.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
from sklearn.preprocessing import Normalizer
from scipy.sparse import hstack
from sklearn.tree import DecisionTreeClassifier
import re
from sklearn.ensemble import RandomForestClassifier
# 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
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
from tqdm import tqdm
import os
from sklearn.tree import export_graphviz
from os import system
import graphviz
from six import StringIO
from chart_studio import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

In [4]:

```
!pip install six
```

```
Requirement already satisfied: six in c:\users\sundararaman\anaconda3\lib\site-packages (1.15.0)
```

1. Reading the Data

In [6]:

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

In [7]:

```python
## Check the shape and attributes of the project data
print("Number of data points in project train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

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

In [8]:

```python
## Check the shape and attributes of the resource data
print("Number of data points in resource train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in resource train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[8]:

	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

1.1 Preprocessing Categorical Features: project_grade_category

In [9]:

```python
print("Project grade" ,project_data['project_grade_category'].value_counts(dropna=False))
## visulaize how project grade looks like
print('-'*50)
print(project_data['project_grade_category'].values[1000])
print(project_data['project_grade_category'].values[1500])
```

```
Project grade Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
--------------------------------------------------
Grades 3-5
Grades PreK-2
```

In [10]:

```python
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-on-other-column-value
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('Grades ','')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[10]:

```
prek_2    44225
3_5       37137
6_8       16923
9_12      10963
Name: project_grade_category, dtype: int64
```

1.2 Preprocessing Categorical Features: project_subject_category

In [11]:

```python
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039
# reference from course material : reference_EDA.ipynb
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)


### maintain a dict that stores count of values
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 Categorical Features: project_subject_subcategory

In [12]:

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

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

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

In [13]:

```python
project_data['school_state'].value_counts()
## Convert it to lower
project_data['school_state'] = project_data['school_state'].str.lower()
#project_data['school_state'].value_counts(dropna=False)
```

1.4 Preprocessing Categorical Features: Teacher_prefix

In [14]:

```python
print(project_data['teacher_prefix'].value_counts(dropna=False))
# try to remove the dots from the teacher prefix and replace nan with mrs.
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
project_data['teacher_prefix']=project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix']=project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix']=project_data['teacher_prefix'].str.strip()
```

```
Mrs.       57269
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
NaN            3
Name: teacher_prefix, dtype: int64
```

1.5 Combining all the essays

In [15]:

```python
# 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)
```

1.6 Number of Words in the Essay and Title

In [16]:

```
source:'''https://www.geeksforgeeks.org/python-program-to-count-words-in-a-sentence/'''
words_counter=[]
for string in project_data['essay']:
    res = len(re.findall(r'\w+', string))
    words_counter.append(res)

project_data["words_in_essay"] = words_counter

words_counter=[]

for string in project_data['project_title']:
    res = len(re.findall(r'\w+', string))
    words_counter.append(res)
project_data["words_in_title"] = words_counter
```

1.7. Preprocessing Numerical Values: price

In [17]:

```
## calculate the overall count of resources and the total price for each project id
price_data=resource_data.groupby('id',as_index=False).agg({'price':'sum','quantity':'sum' })

```

In [18]:

```
project_data = pd.merge(project_data,price_data,on='id',how='left')
```

1.8 Preprocessing Text Features: project_title

In [19]:

```
# https://stackoverflow.com/a/47091490/408403
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 [20]:

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

In [21]:

```
print("printing some random reviews")
print(9, project_data['project_title'].values[9])
print(34, project_data['project_title'].values[34])
print(147, project_data['project_title'].values[147])
```

```
printing some random reviews
9 Just For the Love of Reading--\r\nPure Pleasure
34 \"Have A Ball!!!\"
147 Who needs a Chromebook?\r\nWE DO!!
```

In [22]:

```
# Combining all the above stundents

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

In [23]:

```
preprocessed_titles = preprocess_text(project_data['project_title'].values)
```

```
100%|██████████| 109248/109248 [00:23<00:00, 4647.98it/s]
```

In [24]:

```
print("printing some random reviews")
print(9, preprocessed_titles[9])
print(34, preprocessed_titles[34])
print(147, preprocessed_titles[147])
```

```
printing some random reviews
9 love reading pure pleasure
34 ball
147 needs chromebook
```

1.9 Preprocessing Text Features: essay

In [25]:

```
print("printing some random essay")
print(9, project_data['essay'].values[9])
print('-'*50)
print(34, project_data['essay'].values[34])
print('-'*50)
print(147, project_data['essay'].values[147])
```

```
printing some random essay
9 Over 95% of my students are on free or reduced lunch.  I have a few who are homeless, but despite that, they come to school with an eagerness to learn.  My students are inquisitive eager learners who  embrace the challenge of not having great books
and other resources  every day.  Many of them are not afforded the opportunity to engage with these big colorful pages of a book on a regular basis at home and they don't travel to the public library.  \r\nIt is my duty as a teacher to do all I can t
o provide each student an opportunity to succeed in every aspect of life. \r\nReading is Fundamental! My students will read these books over and over again while boosting their comprehension skills. These books will be used for read alouds, partner r
eading and for Independent reading. \r\nThey will engage in reading to build their \"Love for Reading\" by reading for pure enjoyment. They will be introduced to some new authors as well as some old favorites. I want my students to be ready for the 2
1st Century and know the pleasure of holding a good hard back book in hand. There's nothing like a good book to read!  \r\nMy students will soar in Reading, and more because of your consideration and generous funding contribution. This will help buil
d stamina and prepare for 3rd grade. Thank you so much for reading our proposal!nannan
--------------------------------------------------
34 My students mainly come from extremely low-income families, and the majority of them come from homes where both parents work full time. Most of my students are at school from 7:30 am to 6:00 pm (2:30 to 6:00 pm in the after-school program), and th
ey all receive free and reduced meals for breakfast and lunch. \r\n\r\n\r\nI want my students to feel  as comfortable in my classroom as they do at home. Many of my students take on multiple roles both at home as well as in school. They are sometimes
the caretakers of younger siblings, cooks, babysitters, academics, friends, and most of all, they are developing who they are going to become as adults.  I consider it an essential part of my job to model helping others gain knowledge in a positive m
anner. As a result, I have a community of students who love helping each other in and outside of the classroom. They consistently look for opportunities to support each other's learning in a kind and helpful way.I am excited to be experimenting with
alternative seating in my classroom this school year. Studies have shown that giving students the option of where they sit in a classroom increases focus as well as motivation.  \r\n\r\nBy allowing students choice in the classroom, they are able to e
xplore and create in a welcoming environment. Alternative classroom seating has been experimented with more frequently in recent years. I believe (along with many others), that every child learns differently. This does not only apply to how multiplic
ation is memorized, or a paper is written, but applies to the space in which they are asked to work. I have had students in the past ask \"Can I work in the library? Can I work on the carpet?\" My answer was always, \"As long as you're learning, you
can work wherever you want!\" \r\n\r\nWith the yoga balls and the lap-desks, I will be able to increase the options for seating in my classroom and expand its imaginable space.nannan
--------------------------------------------------
147 My students are eager to learn and make their mark on the world.\r\n\r\nThey come from a Title 1 school and need extra love.\r\n\r\nMy fourth grade students are in a high poverty area and still come to school every day to get their education. I a
m trying to make it fun and educational for them so they can get the most out of their schooling. I created a caring environment for the students to bloom! They deserve the best.\r\nThank you!\r\nI am requesting 1 Chromebook to access online interven
tions, differentiate instruction, and get extra practice. The Chromebook will be used to supplement ELA and math instruction. Students will play ELA and math games that are engaging and fun, as well as participate in assignments online. This in turn
will help my students improve their skills. Having a Chromebook in the classroom would not only allow students to use the programs at their own pace, but would ensure more students are getting adequate time to use the programs. The online programs ha
ve been especially beneficial to my students with special needs. They are able to work at their level as well as be challenged with some different materials. This is making these students more confident in their abilities.\r\n\r\nThe Chromebook would
allow my students to have daily access to computers and increase their computing skills.\r\nThis will change their lives for the better as they become more successful in school. Having access to technology in the classroom would help bridge the achie
vement gap.nannan
```

In [26]:

```
preprocessed_essays = preprocess_text(project_data['essay'].values)
```

```
100%|██████████| 109248/109248 [08:15<00:00, 220.50it/s]
```

In [27]:

```
print("printing some random essay")
print(9, preprocessed_essays[9])
print('-'*50)
print(34, preprocessed_essays[34])
print('-'*50)
print(147, preprocessed_essays[147])

#merge the column in the project_data
project_data['processed_essay']=preprocessed_essays
```

```
printing some random essay
9 95 students free reduced lunch homeless despite come school eagerness learn students inquisitive eager learners embrace challenge not great books resources every day many not afforded opportunity engage big colorful pages book regular basis home no
t travel public library duty teacher provide student opportunity succeed every aspect life reading fundamental students read books boosting comprehension skills books used read alouds partner reading independent reading engage reading build love read
ing reading pure enjoyment introduced new authors well old favorites want students ready 21st century know pleasure holding good hard back book hand nothing like good book read students soar reading consideration generous funding contribution help bu
ild stamina prepare 3rd grade thank much reading proposal nannan
--------------------------------------------------
34 students mainly come extremely low income families majority come homes parents work full time students school 7 30 6 00 pm 2 30 6 00 pm school program receive free reduced meals breakfast lunch want students feel comfortable classroom home many st
udents take multiple roles home well school sometimes caretakers younger siblings cooks babysitters academics friends developing going become adults consider essential part job model helping others gain knowledge positive manner result community stud
ents love helping outside classroom consistently look opportunities support learning kind helpful way excited experimenting alternative seating classroom school year studies shown giving students option sit classroom increases focus well motivation a
llowing students choice classroom able explore create welcoming environment alternative classroom seating experimented frequently recent years believe along many others every child learns differently not apply multiplication memorized paper written a
pplies space asked work students past ask work library work carpet answer always long learning work wherever want yoga balls lap desks able increase options seating classroom expand imaginable space nannan
--------------------------------------------------
147 students eager learn make mark world come title 1 school need extra love fourth grade students high poverty area still come school every day get education trying make fun educational get schooling created caring environment students bloom deserve
best thank requesting 1 chromebook access online interventions differentiate instruction get extra practice chromebook used supplement ela math instruction students play ela math games engaging fun well participate assignments online turn help studen
ts improve skills chromebook classroom would not allow students use programs pace would ensure students getting adequate time use programs online programs especially beneficial students special needs able work level well challenged different material
s making students confident abilities chromebook would allow students daily access computers increase computing skills change lives better become successful school access technology classroom would help bridge achievement gap nannan
```

1.10 Preprocessing Text Features: Project Title

In [28]:

```
print("printing some random project_titles")
print(9, project_data['project_title'].values[9])
print('-'*50)
print(34, project_data['project_title'].values[34])
print('-'*50)
print(147, project_data['project_title'].values[147])
```

```
printing some random project_titles
9 Just For the Love of Reading--\r\nPure Pleasure
--------------------------------------------------
34 \"Have A Ball!!!\"
--------------------------------------------------
147 Who needs a Chromebook?\r\nWE DO!!
```

In [29]:

```
processed_title = preprocess_text(project_data['project_title'].values)
```

```
100%|██████████| 109248/109248 [00:16<00:00, 6788.75it/s]
```

In [30]:

```
print("printing some random title")
print(9, preprocessed_titles[9])
print('-'*50)
print(34, preprocessed_titles[34])
print('-'*50)
print(147, preprocessed_titles[147])

#merge the column in the project_data
project_data['processed_title']=processed_title
```

```
printing some random title
9 love reading pure pleasure
--------------------------------------------------
34 ball
--------------------------------------------------
147 needs chromebook
```

1.11 Creating sentiment columns

In [32]:

```
## craete the sentiment columns using  essay
neg=[]
pos=[]
neu=[]
compound=[]
sentiment_model=SentimentIntensityAnalyzer()
for text in project_data['processed_essay']:
    pol_scores = sentiment_model.polarity_scores(text)
    neg.append(pol_scores['neg'])
    pos.append(pol_scores['pos'])
    neu.append(pol_scores['neu'])
    compound.append(pol_scores['compound'])

project_data['pos']=pos
project_data['neg']=neg
project_data['neu']=neu
project_data['compound']=compound
```

1.12 Dropping redundant columns before splitting

In [33]:

```
project_data.drop(columns=['Unnamed: 0','project_essay_1', 'project_essay_2', 'project_essay_3','project_essay_4','project_title','essay'],inplace=True)
```

2 Train,Test,CV Split

In [52]:

```
# train test split using sklearn.model selection
## Considering only 50k points
project_data=project_data.sample(50000,random_state=1)
X_train, X_test, y_train, y_test = train_test_split(project_data, project_data['project_is_approved'], test_size=0.33, stratify = project_data['project_is_approved'],random_state=0)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train,random_state=0)
```

In [53]:

```
X_train.head(2)
```

Out[53]:

	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_resource_summary	teacher_number_of_previously_posted_projects	project_is_approved	clean_categories	...	words_in_essay	words_in_title	price	quantity	processed_e
38452	p126775	8544a1fcc7f73eddd7de295c66b47738	mrs	ma	2016-08-25 08:24:00	9_12	My students need items to create a coffee cart...	0	0	SpecialNeeds	...	210	2	315.01	9	learning give community s other
38286	p248274	2ea294df350d76c01bc4c8aa019fd2bc	mrs	mt	2016-07-13 13:15:16	3_5	My students need a tinker / maker station!\r\n...	0	0	Math_Science Literacy_Language	...	233	6	609.19	6	teach awe school prou staff thr

2 rows × 21 columns

3. Vectorization on categorical and text features

3.1 Response Encoding on categorical features :

1. Categorical features are : School state,categories,sub_categories,project_grade,teacher_prefix

In [58]:

```
## craete a response table for the categoical data :
def response_dict(col_name):
    col_cat=X_train[col_name].unique()
    res_no,res_yes={},{}
    for cat in col_cat:
        # total number of non approvals (y=0)
        no=len(X_train[(X_train[col_name] == cat) & (X_train['project_is_approved'] == 0)])
        # total number of non approvals (y=1)
        yes=len(X_train[(X_train[col_name] == cat) & (X_train['project_is_approved'] == 1)])
        #total
        total=len(X_train[X_train[col_name] == cat])
        #store the probability value for the category for both class labels
        res_no.update({cat :no/total })
        res_yes.update({cat :yes/total })
    return res_no,res_yes

def response_encoding(data,col_name):
    #store class 0 and class 1 probabilty value for each
    resp_no,resp_yes=response_dict(col_name)
    data['class_0']=0.5
    data['class_1']=0.5
    data['class_0']=data[col_name].map(resp_no)
    data['class_1']=data[col_name].map(resp_yes)
    return data['class_0'].values,data['class_1'].values
```

In [59]:

```
## vectorized categorical variables for our train data
teacher_prefix_tr_0,teacher_prefix_tr_1=response_encoding(X_train,'teacher_prefix')
proj_grade_tr_0,proj_grade_tr_1=response_encoding(X_train,'project_grade_category')
cat_tr_0,cat_tr_1=response_encoding(X_train,'clean_categories')
subcat_tr_0,subcat_tr_1=response_encoding(X_train,'clean_subcategories')
sklst_tr_0,sklst_tr_1=response_encoding(X_train,'school_state')

## merge into 2d
#https://stackoverflow.com/questions/8372399/zip-with-list-output-instead-of-tuple
vectorize_tr_trpr=np.array(list(zip(teacher_prefix_tr_0,teacher_prefix_tr_1)))
vectorize_tr_proj=np.array(list(zip(proj_grade_tr_0,proj_grade_tr_1)))
vectorize_tr_cat=np.array(list(zip(cat_tr_0,cat_tr_1)))
vectorize_tr_subcat=np.array(list(zip(subcat_tr_0,subcat_tr_1)))
vectorize_tr_sklst=np.array(list(zip(sklst_tr_0,sklst_tr_1)))

## print shapes
print(vectorize_tr_trpr.shape)
print(vectorize_tr_proj.shape)
print(vectorize_tr_cat.shape)
print(vectorize_tr_subcat.shape)
print(vectorize_tr_sklst.shape)
```

```
(22445, 2)
(22445, 2)
(22445, 2)
(22445, 2)
(22445, 2)
```

In [60]:

```
## vectorized categorical variables for our test data
teacher_prefix_te_0,teacher_prefix_te_1=response_encoding(X_test,'teacher_prefix')
proj_grade_te_0,proj_grade_te_1=response_encoding(X_test,'project_grade_category')
cat_te_0,cat_te_1=response_encoding(X_test,'clean_categories')
subcat_te_0,subcat_te_1=response_encoding(X_test,'clean_subcategories')
sklst_te_0,sklst_te_1=response_encoding(X_test,'school_state')

## merge into 2d
#https://stackoverflow.com/questions/8372399/zip-with-list-output-instead-of-tuple
vectorize_te_trpr=np.array(list(zip(teacher_prefix_te_0,teacher_prefix_te_1)))
vectorize_te_proj=np.array(list(zip(proj_grade_te_0,proj_grade_te_1)))
vectorize_te_cat=np.array(list(zip(cat_te_0,cat_te_1)))
vectorize_te_subcat=np.array(list(zip(subcat_te_0,subcat_te_1)))
vectorize_te_sklst=np.array(list(zip(sklst_te_0,sklst_te_1)))

## print shapes
print(vectorize_te_trpr.shape)
print(vectorize_te_proj.shape)
print(vectorize_te_cat.shape)
print(vectorize_te_subcat.shape)
print(vectorize_te_sklst.shape)
```

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

In [61]:

```
## vectorized categorical variables for our cv data
teacher_prefix_cv_0,teacher_prefix_cv_1=response_encoding(X_cv,'teacher_prefix')
proj_grade_cv_0,proj_grade_cv_1=response_encoding(X_cv,'project_grade_category')
cat_cv_0,cat_cv_1=response_encoding(X_cv,'clean_categories')
subcat_cv_0,subcat_cv_1=response_encoding(X_cv,'clean_subcategories')
sklst_cv_0,sklst_cv_1=response_encoding(X_cv,'school_state')

## merge into 2d
#https://stackoverflow.com/questions/8372399/zip-with-list-output-instead-of-tuple
vectorize_cv_trpr=np.array(list(zip(teacher_prefix_cv_0,teacher_prefix_cv_1)))
vectorize_cv_proj=np.array(list(zip(proj_grade_cv_0,proj_grade_cv_1)))
vectorize_cv_cat=np.array(list(zip(cat_cv_0,cat_cv_1)))
vectorize_cv_subcat=np.array(list(zip(subcat_cv_0,subcat_cv_1)))
vectorize_cv_sklst=np.array(list(zip(sklst_cv_0,sklst_cv_1)))

## print shapes
print(vectorize_cv_trpr.shape)
print(vectorize_cv_proj.shape)
print(vectorize_cv_cat.shape)
print(vectorize_cv_subcat.shape)
print(vectorize_cv_sklst.shape)
```

```
(11055, 2)
(11055, 2)
(11055, 2)
(11055, 2)
(11055, 2)
```

In [62]:

```
## drop the y labels from splits
X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

3.2 Vectorizing Text data

3.2.1 BOW on Essay data

In [63]:

```
 ##Considering the words that appeared in atleast 10 documents

bow_essay = CountVectorizer(min_df=10,max_features=5000)
bow_essay.fit(X_train['processed_essay'])

bow_essay_train = bow_essay.transform(X_train['processed_essay'])

print("Shape of matrix after one hot encoding ",bow_essay_train.shape)

## tranform Test data

bow_essay_test = bow_essay.transform(X_test['processed_essay'])

print("Shape of matrix after one hot encoding ",bow_essay_test.shape)


## Teansform cv data

bow_essay_cv = bow_essay.transform(X_cv['processed_essay'])
print("Shape of matrix after one hot encoding ",bow_essay_cv.shape)
```

```
Shape of matrix after one hot encoding  (22445, 5000)
Shape of matrix after one hot encoding  (16500, 5000)
Shape of matrix after one hot encoding  (11055, 5000)
```

3.2.2 BOW on Title data

In [64]:

```
 ##Considering the words that appeared in atleast 10 documents

bow_title = CountVectorizer(min_df=10,max_features=5000)
bow_title.fit(X_train['processed_title'])

bow_title_train = bow_title.transform(X_train['processed_title'])

print("Shape of matrix after one hot encoding ",bow_title_train.shape)

## tranform Test data

bow_title_test = bow_title.transform(X_test['processed_title'])

print("Shape of matrix after one hot encoding ",bow_title_test.shape)


## Teansform cv data

bow_title_cv = bow_title.transform(X_cv['processed_title'])
print("Shape of matrix after one hot encoding ",bow_title_cv.shape)
```

```
Shape of matrix after one hot encoding  (22445, 1146)
Shape of matrix after one hot encoding  (16500, 1146)
Shape of matrix after one hot encoding  (11055, 1146)
```

3.2.3 TFIDF on Essay data

In [65]:

```
 ##Considering the words that appeared in atleast 10 documents

tfidf_essay = TfidfVectorizer(min_df=10,max_features=5000)
tfidf_essay.fit(X_train['processed_essay'])

tfidf_essay_train = tfidf_essay.transform(X_train['processed_essay'])

print("Shape of matrix after one hot encoding ",tfidf_essay_train.shape)

## tranform Test data

tfidf_essay_test = tfidf_essay.transform(X_test['processed_essay'])

print("Shape of matrix after one hot encoding ",tfidf_essay_test.shape)


## Teansform cv data

tfidf_essay_cv = tfidf_essay.transform(X_cv['processed_essay'])
print("Shape of matrix after one hot encoding ",tfidf_essay_cv.shape)
```

```
Shape of matrix after one hot encoding  (22445, 5000)
Shape of matrix after one hot encoding  (16500, 5000)
Shape of matrix after one hot encoding  (11055, 5000)
```

3.2.4 TFIDF on Title data

In [66]:

```
 ##Considering the words that appeared in atleast 10 documents

tfidf_title = TfidfVectorizer(min_df=10,max_features=5000)
tfidf_title.fit(X_train['processed_title'])

tfidf_title_train = tfidf_title.transform(X_train['processed_title'])

print("Shape of matrix after one hot encoding ",tfidf_title_train.shape)

## tranform Test data

tfidf_title_test = tfidf_title.transform(X_test['processed_title'])

print("Shape of matrix after one hot encoding ",tfidf_title_test.shape)


## Teansform cv data

tfidf_title_cv = tfidf_title.transform(X_cv['processed_title'])
print("Shape of matrix after one hot encoding ",tfidf_title_cv.shape)
```

```
Shape of matrix after one hot encoding  (22445, 1146)
Shape of matrix after one hot encoding  (16500, 1146)
Shape of matrix after one hot encoding  (11055, 1146)
```

3.2.5 Weighted tfidf on Essay data using Pretrained Models

In [67]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
## Glove vectors are global vectors for words which has vector every word in 300d .
## for read more :https://nlp.stanford.edu/projects/glove/
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [68]:

```
# average Word2Vec on train
# compute average word2vec for each review.
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]

tfidf_model_essay = TfidfVectorizer()
tfidf_model_essay.fit(X_train["processed_essay"])

# we are converting a essay_dictionary with word as a key, and the idf as a value
essay_dictionary = dict(zip(tfidf_model_essay.get_feature_names(), list(tfidf_model_essay.idf_)))
tfidf_words_essay = set(tfidf_model_essay.get_feature_names())



def tfidf_w2v_vectors(data,glove_words,essay_dictionary,tfidf_words_essay):
    tf_vector=[]
    for sentence in data: # 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_essay):
                vec = model[word] # getting the vector for each word
                # here we are multiplying idf value(essay_dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
                tf_idf = essay_dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tf_vector.append(vector)
    print(len(tf_vector))
    print(len(tf_vector[0]))
    return tf_vector

tfidf_w2v_vectors_train=tfidf_w2v_vectors(X_train['processed_essay'],glove_words,essay_dictionary,tfidf_words_essay)
tfidf_w2v_vectors_cv=tfidf_w2v_vectors(X_cv['processed_essay'],glove_words,essay_dictionary,tfidf_words_essay)
tfidf_w2v_vectors_test=tfidf_w2v_vectors(X_test['processed_essay'],glove_words,essay_dictionary,tfidf_words_essay)
```

```
22445
300
11055
300
16500
300
```

3.2.6 Weighted tfidf on Title data using Pretrained Models

In [69]:

```
# average Word2Vec on train
# compute average word2vec for each review.
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]

tfidf_model_title = TfidfVectorizer()
tfidf_model_title.fit(X_train["processed_title"])
# we are converting a title_dictionary with word as a key, and the idf as a value
title_dictionary = dict(zip(tfidf_model_title.get_feature_names(), list(tfidf_model_title.idf_)))
tfidf_words_title = set(tfidf_model_title.get_feature_names())

tfidf_w2v_vectors_title_train=tfidf_w2v_vectors(X_train['processed_title'],glove_words,title_dictionary,tfidf_words_title)
tfidf_w2v_vectors_title_cv=tfidf_w2v_vectors(X_cv['processed_title'],glove_words,title_dictionary,tfidf_words_title)
tfidf_w2v_vectors_title_test=tfidf_w2v_vectors(X_test['processed_title'],glove_words,title_dictionary,tfidf_words_title)
```

```
22445
300
11055
300
16500
300
```

3.2.7 Avg W2V on Essay data using Pretrained Models

In [70]:

```python
# average Word2Vec on train
# compute average word2vec for each review.


def avg_w2v_vector(data,glove_words):
    tf_vector=[]
    for sentence in tqdm(data): # 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
        tf_vector.append(vector)

    print(len(tf_vector))
    print(len(tf_vector[0]))
    return tf_vector

# average Word2Vec on CV
# compute average word2vec for each review.
avg_w2v_vectors_train = avg_w2v_vector(X_train['processed_essay'],glove_words)
avg_w2v_vectors_cv = avg_w2v_vector(X_cv['processed_essay'],glove_words)
avg_w2v_vectors_test = avg_w2v_vector(X_test['processed_essay'],glove_words)

```

```
100%|██████████| 22445/22445 [00:11<00:00, 1922.51it/s]
  2%|▏         | 258/11055 [00:00<00:04, 2561.30it/s]

22445
300

100%|██████████| 11055/11055 [00:07<00:00, 1432.30it/s]
  2%|▏         | 364/16500 [00:00<00:09, 1692.43it/s]

11055
300

100%|██████████| 16500/16500 [00:07<00:00, 2249.37it/s]

16500
300
```

3.2.8 Avg W2V on Title data using Pretrained Models

In [71]:

```python
# average Word2Vec on CV
# compute average word2vec for each review.
avg_w2v_vectors_title_train = avg_w2v_vector(X_train['processed_title'],glove_words)
avg_w2v_vectors_title_cv = avg_w2v_vector(X_cv['processed_title'],glove_words)
avg_w2v_vectors_title_test = avg_w2v_vector(X_test['processed_title'],glove_words)

```

```
100%|██████████| 22445/22445 [00:01<00:00, 17903.75it/s]
 20%|██        | 2157/11055 [00:00<00:00, 21413.49it/s]

22445
300

100%|██████████| 11055/11055 [00:00<00:00, 16922.96it/s]
  0%|          | 0/16500 [00:00<?, ?it/s]

11055
300

100%|██████████| 16500/16500 [00:00<00:00, 19486.67it/s]

16500
300
```

4. Vectorizing Numerical Features

4.1 Price

In [72]:

```python
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(1,-1))

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(1,-1))
X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(1,-1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)

## reshaping
X_train_price_norm=X_train_price_norm.reshape(-1,1)
X_cv_price_norm=X_cv_price_norm.reshape(-1,1)
X_test_price_norm=X_test_price_norm.reshape(-1,1)
```

```
After vectorizations
(1, 22445) (22445,)
(1, 11055) (11055,)
(1, 16500) (16500,)
====================================================================================================
```

4.2 Quantity

In [73]:

```python
normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer.fit(X_train['quantity'].values.reshape(1,-1))

quantity_train_norm = normalizer.transform(X_train['quantity'].values.reshape(1,-1))
quantity_cv_norm = normalizer.transform(X_cv['quantity'].values.reshape(1,-1))
quantity_test_norm = normalizer.transform(X_test['quantity'].values.reshape(1,-1))

print("After vectorizations")
print(quantity_train_norm.shape, y_train.shape)
print(quantity_cv_norm.shape, y_cv.shape)
print(quantity_test_norm.shape, y_test.shape)
print("="*100)

## reshaping
quantity_train_norm=quantity_train_norm.reshape(-1,1)
quantity_cv_norm=quantity_cv_norm.reshape(-1,1)
quantity_test_norm=quantity_test_norm.reshape(-1,1)
```

```
After vectorizations
(1, 22445) (22445,)
(1, 11055) (11055,)
(1, 16500) (16500,)
====================================================================================================
```

4.3 Number of Previously posted projects

In [74]:

```python
normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

prev_projects_train_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
prev_projects_cv_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
prev_projects_test_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

print("After vectorizations")
print(prev_projects_train_norm.shape, y_train.shape)
print(prev_projects_cv_norm.shape, y_cv.shape)
print(prev_projects_test_norm.shape, y_test.shape)
print("="*100)

## reshaping
prev_projects_train_norm=prev_projects_train_norm.reshape(-1,1)
prev_projects_cv_norm=prev_projects_cv_norm.reshape(-1,1)
prev_projects_test_norm=prev_projects_test_norm.reshape(-1,1)
```

```
After vectorizations
(1, 22445) (22445,)
(1, 11055) (11055,)
(1, 16500) (16500,)
====================================================================================================
```

4.4 Title Word counts

In [75]:

```python
normalizer = Normalizer()

normalizer.fit(X_train['words_in_title'].values.reshape(1,-1))

title_word_count_train_norm = normalizer.transform(X_train['words_in_title'].values.reshape(1,-1))
title_word_count_cv_norm = normalizer.transform(X_cv['words_in_title'].values.reshape(1,-1))
title_word_count_test_norm = normalizer.transform(X_test['words_in_title'].values.reshape(1,-1))

print("After vectorizations")
print(title_word_count_train_norm.shape, y_train.shape)
print(title_word_count_cv_norm.shape, y_cv.shape)
print(title_word_count_test_norm.shape, y_test.shape)
print("="*100)

## reshaping
title_word_count_train_norm=title_word_count_train_norm.reshape(-1,1)
title_word_count_cv_norm=title_word_count_cv_norm.reshape(-1,1)
title_word_count_test_norm=title_word_count_test_norm.reshape(-1,1)
```

```
After vectorizations
(1, 22445) (22445,)
(1, 11055) (11055,)
(1, 16500) (16500,)
====================================================================================================
```

4.5 Essay Words Counts

In [76]:

```python
normalizer = Normalizer()

normalizer.fit(X_train['words_in_essay'].values.reshape(1,-1))

essay_word_count_train_norm = normalizer.transform(X_train['words_in_essay'].values.reshape(1,-1))
essay_word_count_cv_norm = normalizer.transform(X_cv['words_in_essay'].values.reshape(1,-1))
essay_word_count_test_norm = normalizer.transform(X_test['words_in_essay'].values.reshape(1,-1))

print("After vectorizations")
print(essay_word_count_train_norm.shape, y_train.shape)
print(essay_word_count_cv_norm.shape, y_cv.shape)
print(essay_word_count_test_norm.shape, y_test.shape)

## reshaping
essay_word_count_train_norm=essay_word_count_train_norm.reshape(-1,1)
essay_word_count_cv_norm=essay_word_count_cv_norm.reshape(-1,1)
essay_word_count_test_norm=essay_word_count_test_norm.reshape(-1,1)
```

```
After vectorizations
(1, 22445) (22445,)
(1, 11055) (11055,)
(1, 16500) (16500,)
```

4.6 Vectorizing sentiment Columns

In [81]:

```python
### vectorize pos
Normalize_pos=Normalizer()
Normalize_pos.fit(X_train['pos'].values.reshape(1,-1))
sentiment_pos_train_norm=Normalize_pos.transform(X_train['pos'].values.reshape(1,-1))
sentiment_pos_test_norm=Normalize_pos.transform(X_test['pos'].values.reshape(1,-1))
sentiment_pos_cv_norm=Normalize_pos.transform(X_cv['pos'].values.reshape(1,-1))
sentiment_pos_train_norm=sentiment_pos_train_norm.reshape(-1,1)
sentiment_pos_test_norm=sentiment_pos_test_norm.reshape(-1,1)
sentiment_pos_cv_norm=sentiment_pos_cv_norm.reshape(-1,1)
```

In [82]:

```python
### vectorize neg
Normalize_neg=Normalizer()
Normalize_neg.fit(X_train['neg'].values.reshape(1,-1))
sentiment_neg_train_norm=Normalize_neg.transform(X_train['neg'].values.reshape(1,-1))
sentiment_neg_test_norm=Normalize_neg.transform(X_test['neg'].values.reshape(1,-1))
sentiment_neg_cv_norm=Normalize_neg.transform(X_cv['neg'].values.reshape(1,-1))
sentiment_neg_train_norm=sentiment_neg_train_norm.reshape(-1,1)
sentiment_neg_test_norm=sentiment_neg_test_norm.reshape(-1,1)
sentiment_neg_cv_norm=sentiment_neg_cv_norm.reshape(-1,1)
```

In [83]:

```python
### vectorize compound
Normalize_co=Normalizer()
Normalize_co.fit(X_train['compound'].values.reshape(1,-1))
sentiment_compound_train_norm=Normalize_co.transform(X_train['compound'].values.reshape(1,-1))
sentiment_compound_test_norm=Normalize_co.transform(X_test['compound'].values.reshape(1,-1))
sentiment_compound_cv_norm=Normalize_co.transform(X_cv['compound'].values.reshape(1,-1))
sentiment_compound_train_norm=sentiment_compound_train_norm.reshape(-1,1)
sentiment_compound_test_norm=sentiment_compound_test_norm.reshape(-1,1)
sentiment_compound_cv_norm=sentiment_compound_cv_norm.reshape(-1,1)
```

In [84]:

```python
### vectorize neu
Normalize_neu=Normalizer()
Normalize_neu.fit(X_train['neu'].values.reshape(1,-1))
sentiment_neu_train_norm=Normalize_neu.transform(X_train['neu'].values.reshape(1,-1))
sentiment_neu_test_norm=Normalize_neu.transform(X_test['neu'].values.reshape(1,-1))
sentiment_neu_cv_norm=Normalize_neu.transform(X_cv['neu'].values.reshape(1,-1))
sentiment_neu_train_norm=sentiment_neu_train_norm.reshape(-1,1)
sentiment_neu_test_norm=sentiment_neu_test_norm.reshape(-1,1)
sentiment_neu_cv_norm=sentiment_neu_cv_norm.reshape(-1,1)
```

Assignment 9: RF

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

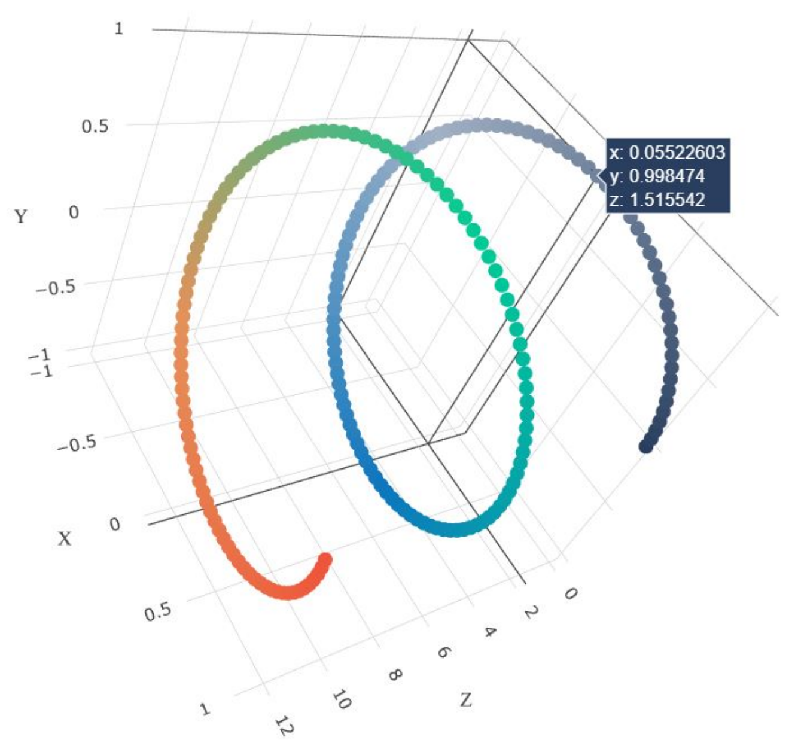
    - Set 1: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical(instead 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). Here for this set take **20K** datapoints only.
    - 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). Here for this set take **20K** datapoints only.

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

    - Consider the following range for hyperparameters **n_estimators** = [10, 50, 100, 150, 200, 300, 500, 1000], **max_depth** = [2, 3, 4, 5, 6, 7, 8, 9, 10]
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using simple cross validation data
    - You can write your own for loops to do this task

3. **Representation of results**

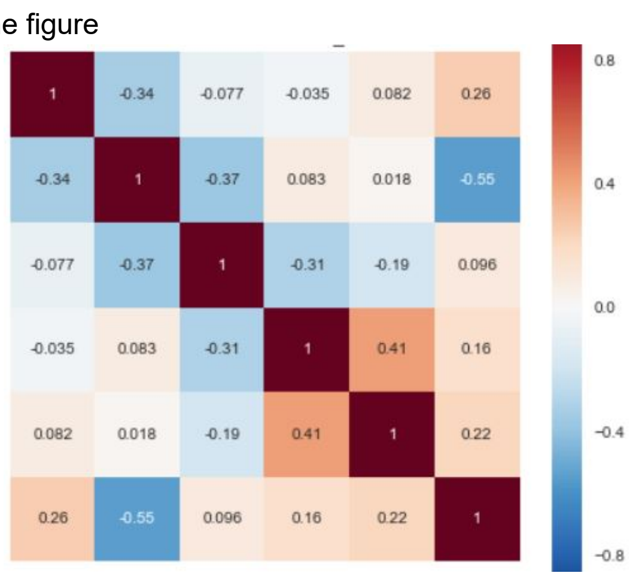
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



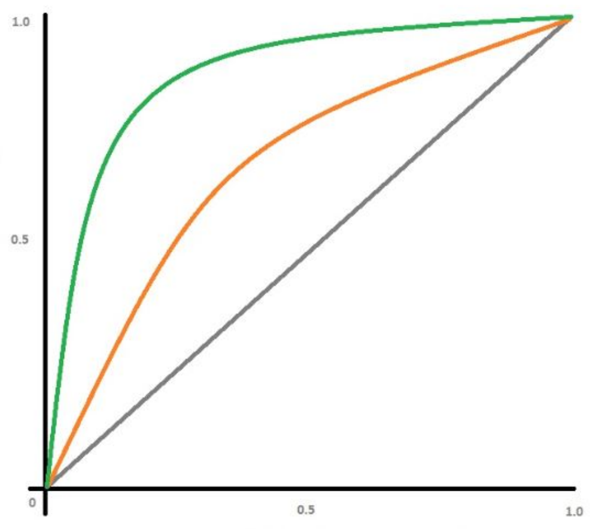
with X-axis as **n_estimators**, Y-axis as **max_depth**, and Z-axis as **AUC Score** , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

or

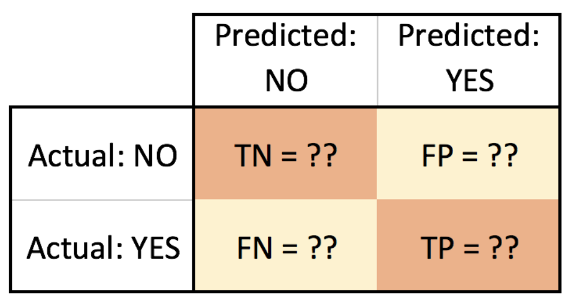
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



  [seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**
- You can choose either of the plotting techniques: 3d plot or heat map
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.



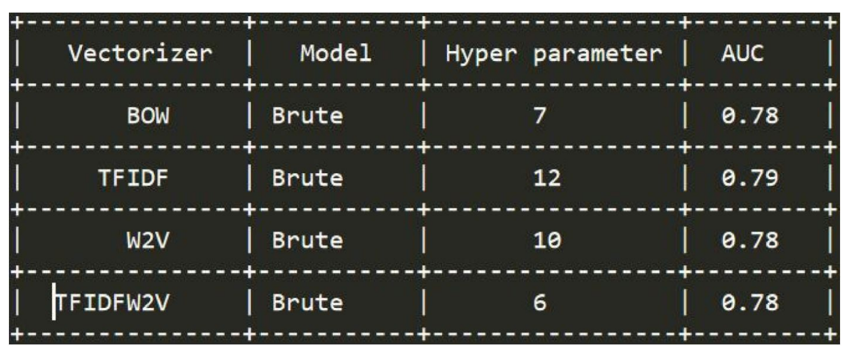
- Along with plotting ROC curve, you need to print the [confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted 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/)](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 |
+------------+--------+-----------------+------+
```

5. 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

5.1 Applying Random Forests on BOW, SET 1

In [85]:

```python
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

X_tr = hstack((vectorize_tr_trpr, vectorize_tr_proj,vectorize_tr_cat,vectorize_tr_subcat,
            vectorize_tr_sklst,bow_essay_train,bow_title_train,sentiment_pos_train_norm,
            sentiment_neg_train_norm,sentiment_compound_train_norm,sentiment_neu_train_norm,
            X_train_price_norm,quantity_train_norm,prev_projects_train_norm,title_word_count_train_norm,
            essay_word_count_train_norm)).tocsr()

X_te = hstack((vectorize_te_trpr, vectorize_te_proj,vectorize_te_cat,vectorize_te_subcat,
            vectorize_te_sklst,bow_essay_test,bow_title_test,sentiment_pos_test_norm,
            sentiment_neg_test_norm,sentiment_compound_test_norm,sentiment_neu_test_norm,
            X_test_price_norm,quantity_test_norm,prev_projects_test_norm,title_word_count_test_norm,
            essay_word_count_test_norm)).tocsr()

X_cr = hstack((vectorize_cv_trpr, vectorize_cv_proj,vectorize_cv_cat,vectorize_cv_subcat,
            vectorize_cv_sklst,bow_essay_cv,bow_title_cv,sentiment_pos_cv_norm,
            sentiment_neg_cv_norm,sentiment_compound_cv_norm,sentiment_neu_cv_norm,
            X_cv_price_norm,quantity_cv_norm,prev_projects_cv_norm,title_word_count_cv_norm,
            essay_word_count_cv_norm)).tocsr()


print(X_tr.shape)
print(X_te.shape)
print(X_cr.shape)
```

```
(22445, 6165)
(16500, 6165)
(11055, 6165)
```

In [86]:

```python
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 6165) (22445,)
(11055, 6165) (11055,)
(16500, 6165) (16500,)
====================================================================================================
```

5.1.1 Write loop to find best hyperparameters and do simple cross validation

In [87]:

```python
## Iterative looop to find best parameter
n_estimators = [10, 50, 100, 150, 200, 300, 500, 1000]
max_depth = [2, 3, 4, 5, 6, 7, 8, 9, 10,100,500]

def hyperparam_auc(max_depth,n_estimators,X_tr,X_cr,y_train,y_cv):
    train_auc=[]
    cv_auc=[]
    es,de=[],[]
    for estimators in tqdm(n_estimators):
        for depth in max_depth:
            RF=RandomForestClassifier(n_estimators=estimators,max_depth=depth,
                                      class_weight='balanced',
                                      random_state=4)
            RF.fit(X_tr,y_train)
            #predict train and cv
            train_predicts=RF.predict_proba(X_tr)[:,1]
            cv_predicts=RF.predict_proba(X_cr)[:,1]
            #Store train and cv auc score in dict
            train_auc.append(roc_auc_score(y_train,train_predicts))
            cv_auc.append(roc_auc_score(y_cv,cv_predicts))
            es.append(estimators)
            de.append(depth)
    return train_auc,cv_auc,es,de

train_auc,cv_auc,es,de=hyperparam_auc(max_depth,n_estimators,X_tr,X_cr,y_train,y_cv)
```

```
100%|██████████| 8/8 [1:47:17<00:00, 804.64s/it]
```

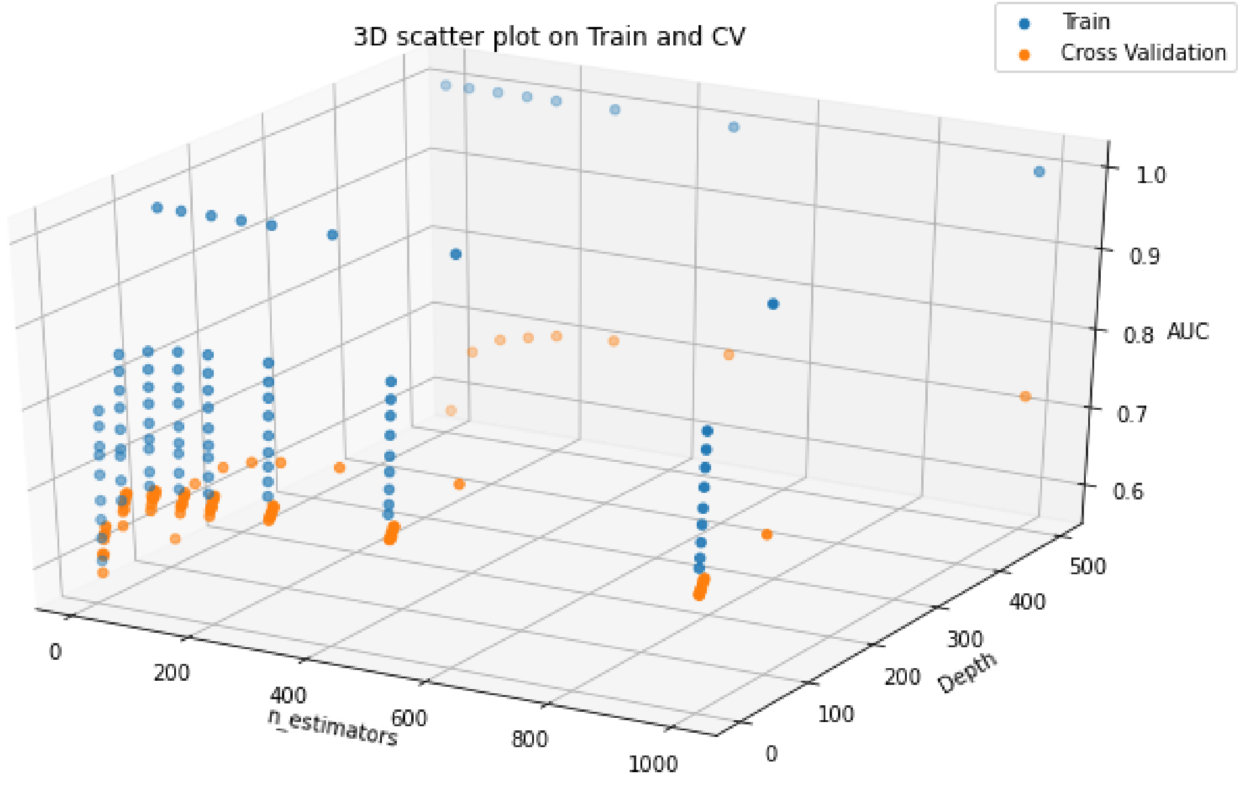
5.1.2 Representation using 3D scatter plot

In [88]:

```python
from mpl_toolkits.mplot3d import Axes3D
plt.figure(figsize=(12,7))
ax = plt.axes(projection='3d')
zipped_=list(map(list,zip(es,de)))
x1=[i[0] for i in zipped_]
y1=[i[1] for i in zipped_]
# Data for three-dimensional scattered points
# reference : https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html
ax.scatter3D(x1, y1, train_auc,label='Train')
ax.scatter3D(x1, y1, cv_auc,label='Cross Validation')
ax.set_xlabel('n_estimators')
ax.set_ylabel('Depth')
ax.set_zlabel('AUC')
ax.set_title('3D scatter plot on Train and CV')
ax.legend()
```

Out[88]:

```
<matplotlib.legend.Legend at 0x227d7cbb0b8>
```



5.1.3 Finding best parameter for training the model

In [89]:

```python
## finding the best pair of hyperparameter with max AUC using a function

def find_best_hyperparam(x1,y1,cv_auc):
    zipped_=list(zip(x1,y1,cv_auc))
    max_auc=max(cv_auc)
    print('Max_auc is',max_auc)
    for i in zipped_:
        if i[2] == max_auc:
            best_params={'n_estimators':i[0],'depth':i[1]}
        else:
            pass
    return best_params
```

In [93]:

```python
best_params=find_best_hyperparam(x1,y1,cv_auc)
best_params
```

```
Max_auc is 0.7247584111051009
```

Out[93]:

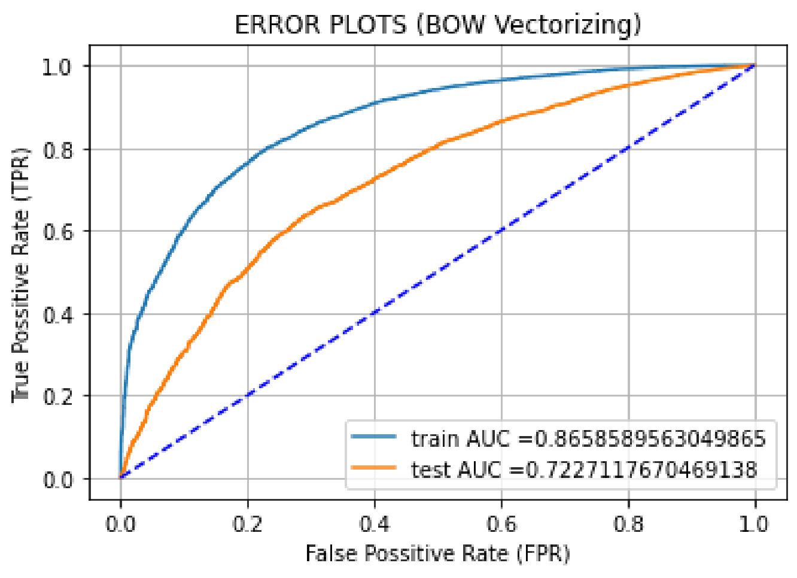
```
{'n_estimators': 1000, 'depth': 100}
```

In [99]:

```python
RF=RandomForestClassifier(class_weight='balanced',n_estimators=best_params['n_estimators'],
                          max_depth=best_params['depth'],min_samples_split=1000)
RF.fit(X_tr,y_train)

## Predict the test
train_predicts=RF.predict_proba(X_tr)[:,1]
test_predicts=RF.predict_proba(X_te)[:,1]

## Store fpr and tpr rates

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, train_predicts)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, test_predicts)
```

In [100]:

```python
#plot
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Possitive Rate (FPR)")
plt.ylabel("True Possitive Rate (TPR)")
plt.title("ERROR PLOTS (BOW Vectorizing)")
plt.plot([0, 1], [0, 1],'b--')
plt.grid()
plt.show()


## Store auc results in a dictionary
results_dict={'BOW' : {'trainauc':str(auc(train_fpr, train_tpr)),
                       'testauc': str(auc(test_fpr, test_tpr)),
                        'max_depth' : best_params['depth'],
                         'n_estimators' :best_params['n_estimators']}}
```



5.1.4 Confusion Matrix on train and test data

In [102]:

```python
## Finding best threshold for predictions
def best_threshold(thresholds,fpr,tpr):
    t=thresholds[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

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

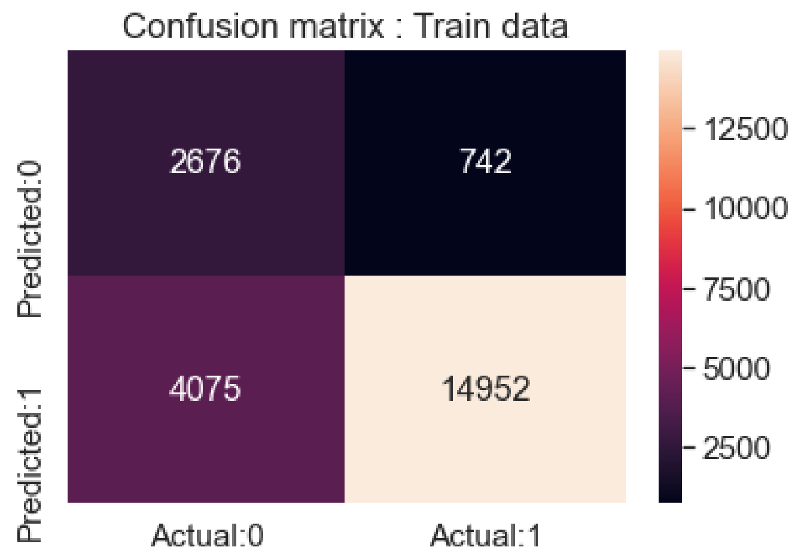
In [104]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t=best_threshold(tr_thresholds,train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(train_predicts, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(test_predicts, best_t)))
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.6152378147120736 for threshold 0.513
Train confusion matrix
[[ 2676   742]
 [ 4075 14952]]
Test confusion matrix
[[ 1321  1191]
 [ 3010 10978]]
```

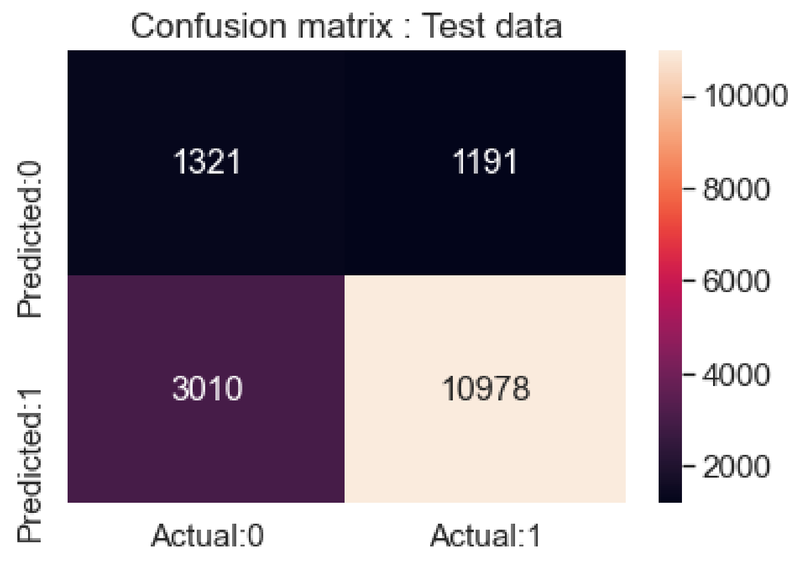
In [105]:

```
### PLOT the matrix for Train
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(train_predicts, best_t))
                     , range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',
            xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Train data')
plt.show()
```



In [106]:

```
### PLOT the matrix for Train
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(test_predicts, best_t)), range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Test data')
plt.show()
```



5.2 Applying Random Forests on TFIDF, SET 2

In [107]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

X_tr = hstack((vectorize_tr_trpr, vectorize_tr_proj,vectorize_tr_cat,vectorize_tr_subcat,
            vectorize_tr_sklst,tfidf_essay_train,tfidf_title_train,sentiment_pos_train_norm,
            sentiment_neg_train_norm,sentiment_compound_train_norm,sentiment_neu_train_norm,
            X_train_price_norm,quantity_train_norm,prev_projects_train_norm,title_word_count_train_norm,
            essay_word_count_train_norm)).tocsr()

X_te = hstack((vectorize_te_trpr, vectorize_te_proj,vectorize_te_cat,vectorize_te_subcat,
            vectorize_te_sklst,tfidf_essay_test,tfidf_title_test,sentiment_pos_test_norm,
            sentiment_neg_test_norm,sentiment_compound_test_norm,sentiment_neu_test_norm,
            X_test_price_norm,quantity_test_norm,prev_projects_test_norm,title_word_count_test_norm,
            essay_word_count_test_norm)).tocsr()

X_cr = hstack((vectorize_cv_trpr, vectorize_cv_proj,vectorize_cv_cat,vectorize_cv_subcat,
            vectorize_cv_sklst,tfidf_essay_cv,tfidf_title_cv,sentiment_pos_cv_norm,
            sentiment_neg_cv_norm,sentiment_compound_cv_norm,sentiment_neu_cv_norm,
            X_cv_price_norm,quantity_cv_norm,prev_projects_cv_norm,title_word_count_cv_norm,
            essay_word_count_cv_norm)).tocsr()


print(X_tr.shape)
print(X_te.shape)
print(X_cr.shape)
```

```
(22445, 6165)
(16500, 6165)
(11055, 6165)
```

In [108]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 6165) (22445,)
(11055, 6165) (11055,)
(16500, 6165) (16500,)
====================================================================================================
```

5.2.1 Write loop to find best hyperparameters and do simple cross validation

In [109]:

```
## Iterative looop to find best parameter
n_estimators = [10, 50, 100, 150, 200, 300, 500, 1000]
max_depth = [2, 3, 4, 5, 6, 7, 8, 9, 10,100,500]


train_auc,cv_auc,es,de=hyperparam_auc(max_depth,n_estimators,X_tr,X_cr,y_train,y_cv)
```

```
100%|██████████| 8/8 [2:15:51<00:00, 1018.92s/it]
```

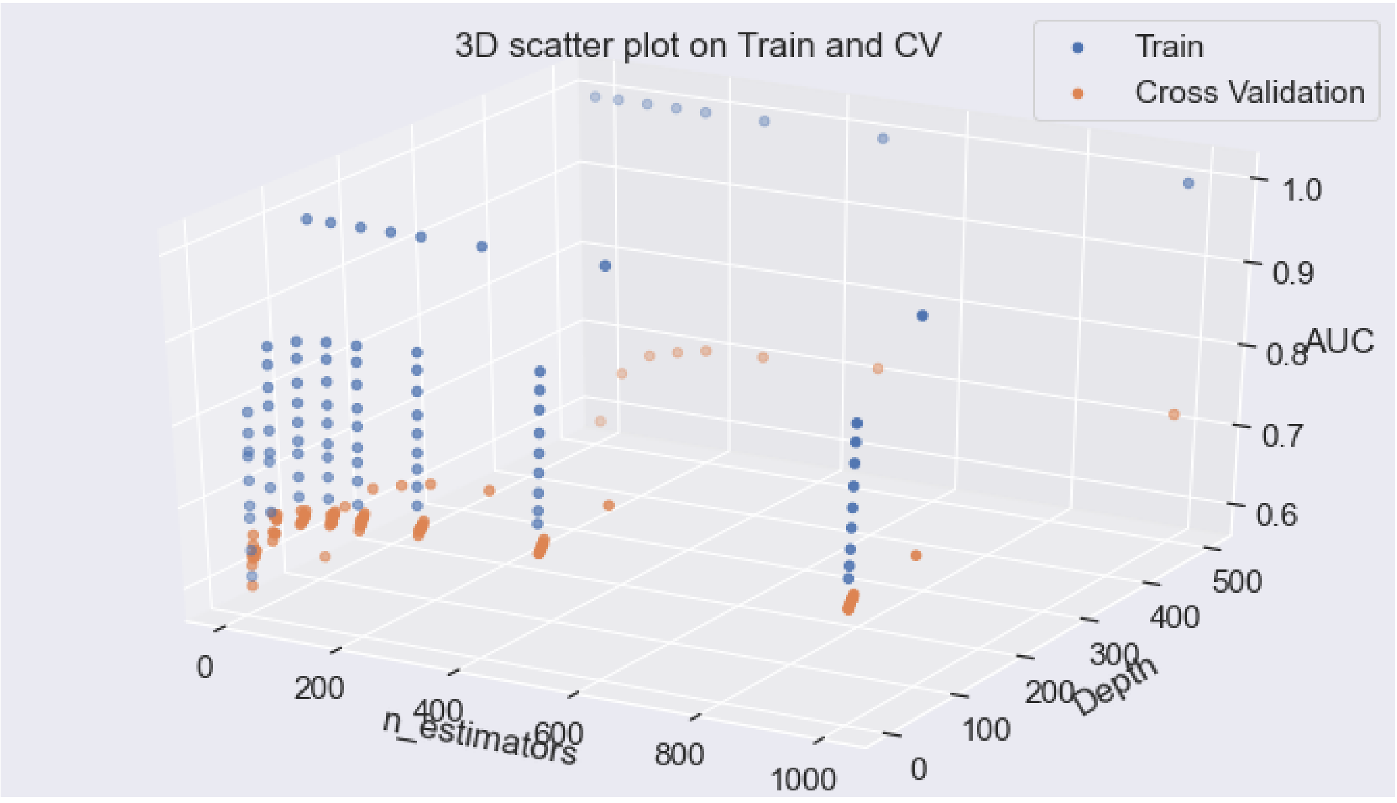
5.2.2 Representation using 3D scatter plot

In [115]:

```
#from mpl_toolkits.mplot3d import Axes3D
plt.figure(figsize=(12,7))
ax = plt.axes(projection='3d')
zipped_=list(map(list,zip(es,de)))
x1=[i[0] for i in zipped_]
y1=[i[1] for i in zipped_]
# Data for three-dimensional scattered points
# reference : https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html
ax.scatter3D(x1, y1, train_auc,label='Train')
ax.scatter3D(x1, y1, cv_auc,label='Cross Validation')
ax.set_xlabel('n_estimators')
ax.set_ylabel('Depth')
ax.set_zlabel('AUC')
ax.set_title('3D scatter plot on Train and CV')
ax.legend()
```

Out[115]:

```
<matplotlib.legend.Legend at 0x227d7c3a3c8>
```



5.2.3 Finding best parameter for training the model

In [116]:

```
best_params=find_best_hyperparam(x1,y1,cv_auc)
best_params
```

```
Max_auc is 0.7205517807686973
```

Out[116]:

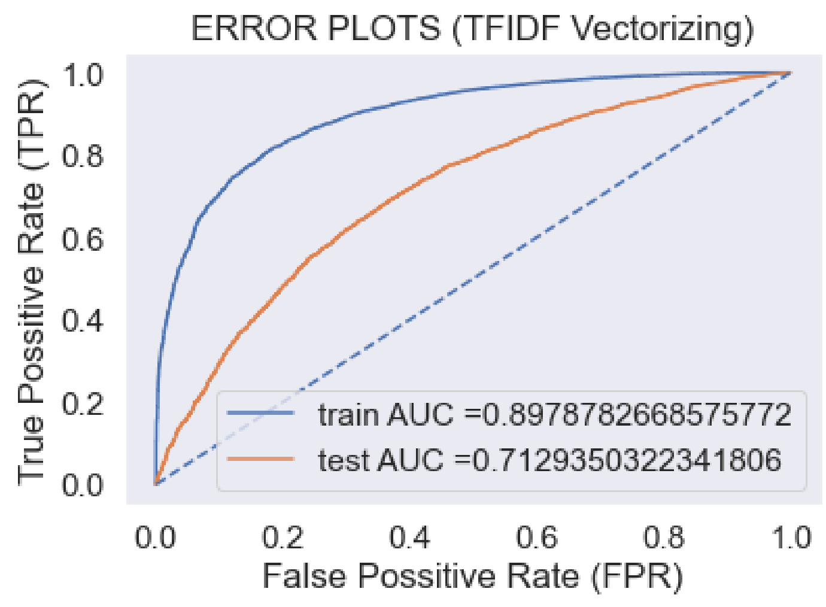
```
{'n_estimators': 1000, 'depth': 100}
```

In [117]:

```
RF=RandomForestClassifier(class_weight='balanced',n_estimators=best_params['n_estimators'],
                          max_depth=best_params['depth'],random_state=4,min_samples_split=1000)
RF.fit(X_tr,y_train)

## Predict the test
train_predicts=RF.predict_proba(X_tr)[:,1]
test_predicts=RF.predict_proba(X_te)[:,1]

## Store fpr and tpr rates

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, train_predicts)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, test_predicts)
```

In [118]:

```
#plot
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Possitive Rate (FPR)")
plt.ylabel("True Possitive Rate (TPR)")
plt.title("ERROR PLOTS (TFIDF Vectorizing)")
plt.plot([0, 1], [0, 1],'b--')
plt.grid()
plt.show()


## Store auc results in a dictionary
results_dict.update({'TFIDF' : {'trainauc':str(auc(train_fpr, train_tpr)),
                       'testauc': str(auc(test_fpr, test_tpr)),
                        'max_depth' : best_params['depth'],
                         'n_estimators' :best_params['n_estimators']} })
```



5.2.4 Confusion Matrix on train and test data

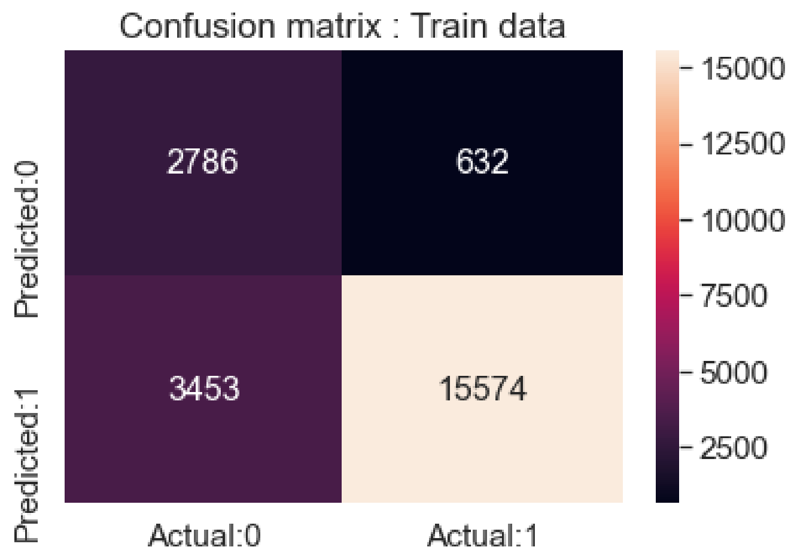
In [119]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t=best_threshold(tr_thresholds,train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(train_predicts, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(test_predicts, best_t)))
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.6671736812794408 for threshold 0.513
Train confusion matrix
[[ 2786   632]
 [ 3453 15574]]
Test confusion matrix
[[ 1272  1240]
 [ 2928 11060]]
```

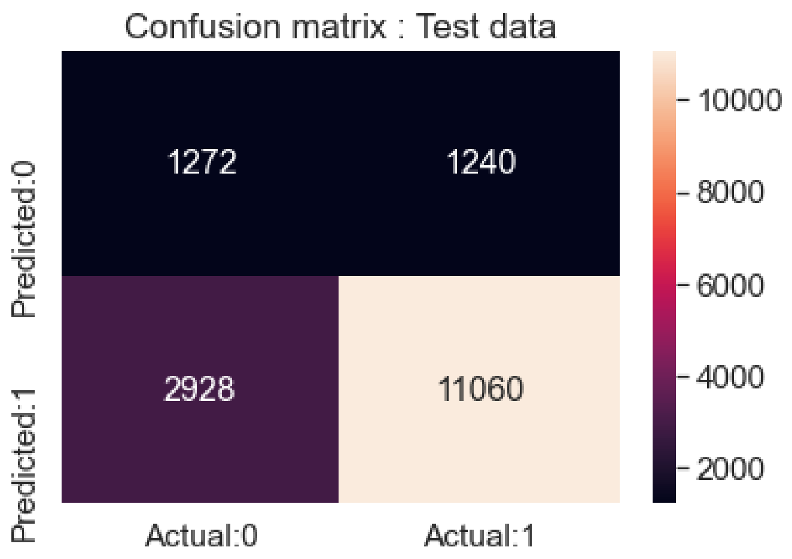
In [120]:

```
### PLOT the matrix for Train
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(train_predicts, best_t))
                     , range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',
            xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Train data')
plt.show()
```



In [121]:

```
### PLOT the matrix for Train
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(test_predicts, best_t)), range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',xticklabels=['Actual:0','Actual:1']
            ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Test data')
plt.show()
```



5.3 Applying Random Forests on AVG W2V, SET 3

In [122]:

```
## converting to csr_matrix to avoid error : "could not broadcast input array from shape
## reference : https://stackoverflow.com/questions/24924940/convert-list-and-list-of-lists-to-scipy-sparse-arrays
avg_w2v_vectors_train=scipy.sparse.csr_matrix(avg_w2v_vectors_train)
avg_w2v_vectors_title_train=scipy.sparse.csr_matrix(avg_w2v_vectors_title_train)
avg_w2v_vectors_test=scipy.sparse.csr_matrix(avg_w2v_vectors_test)
avg_w2v_vectors_title_test=scipy.sparse.csr_matrix(avg_w2v_vectors_title_test)
avg_w2v_vectors_cv=scipy.sparse.csr_matrix(avg_w2v_vectors_cv)
avg_w2v_vectors_title_cv=scipy.sparse.csr_matrix(avg_w2v_vectors_title_cv)

```

In [123]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039b
X_tr = hstack((vectorize_tr_trpr, vectorize_tr_proj,vectorize_tr_cat,vectorize_tr_subcat,
            vectorize_tr_sklst,avg_w2v_vectors_train,avg_w2v_vectors_title_train,sentiment_pos_train_norm,
            sentiment_neg_train_norm,sentiment_compound_train_norm,sentiment_neu_train_norm,
            X_train_price_norm,quantity_train_norm,prev_projects_train_norm,title_word_count_train_norm,
            essay_word_count_train_norm)).tocsr()


X_te = hstack((vectorize_te_trpr, vectorize_te_proj,vectorize_te_cat,vectorize_te_subcat,
            vectorize_te_sklst,avg_w2v_vectors_test,avg_w2v_vectors_title_test,sentiment_pos_test_norm,
            sentiment_neg_test_norm,sentiment_compound_test_norm,sentiment_neu_test_norm,
            X_test_price_norm,quantity_test_norm,prev_projects_test_norm,title_word_count_test_norm,
            essay_word_count_test_norm)).tocsr()

X_cr = hstack((vectorize_cv_trpr, vectorize_cv_proj,vectorize_cv_cat,vectorize_cv_subcat,
            vectorize_cv_sklst,avg_w2v_vectors_cv,avg_w2v_vectors_title_cv,sentiment_pos_cv_norm,
            sentiment_neg_cv_norm,sentiment_compound_cv_norm,sentiment_neu_cv_norm,
            X_cv_price_norm,quantity_cv_norm,prev_projects_cv_norm,title_word_count_cv_norm,
            essay_word_count_cv_norm)).tocsr()


print(X_tr.shape)
print(X_te.shape)
print(X_cr.shape)
```

```
(22445, 619)
(16500, 619)
(11055, 619)
```

In [382]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 619) (22445,)
(11055, 619) (11055,)
(16500, 619) (16500,)
====================================================================================================
```

In [384]:

```
### taking only 20k points for this set from the randomly sampled 50k dataset.
X_tr=X_tr[:10000]
X_cr=X_cr[:5000]
X_te=X_te[:5000]
y_tr=y_train[:10000]
y_cr=y_cv[:5000]
y_te=y_test[:5000]
```

In [385]:

```
print("Final Data matrix")
print(X_tr.shape, y_tr.shape)
print(X_cr.shape, y_cr.shape)
print(X_te.shape, y_te.shape)
print("="*100)
```

```
Final Data matrix
(10000, 619) (10000,)
(5000, 619) (5000,)
(5000, 619) (5000,)
====================================================================================================
```

5.3.1 Write loop to find best hyperparameters and do simple cross validation

In [386]:

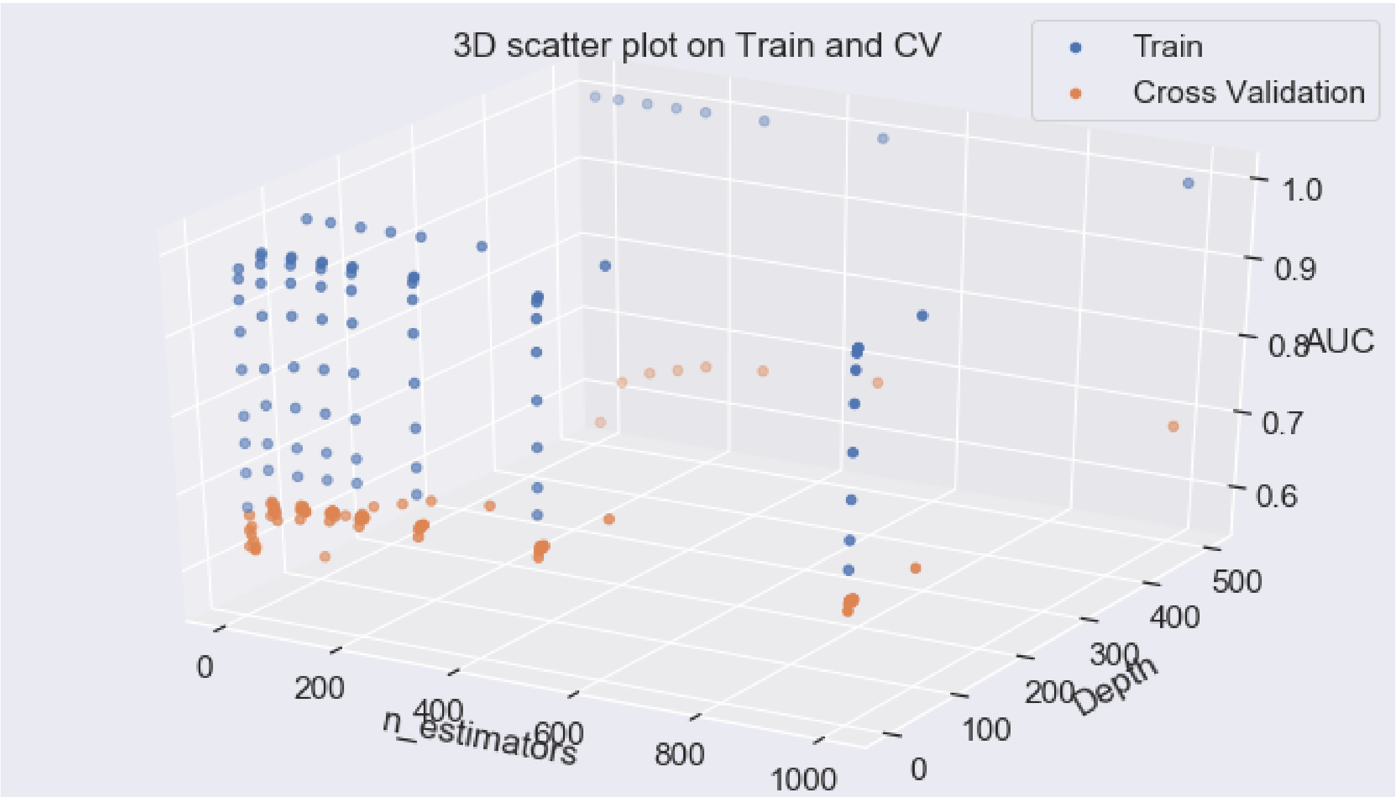
```
## Iterative looop to find best parameter
n_estimators = [10, 50, 100, 150, 200, 300, 500, 1000]
max_depth = [2, 3, 4, 5, 6, 7, 8, 9, 10,100,500]

train_auc,cv_auc,es,de=hyperparam_auc(max_depth,n_estimators,X_tr,X_cr,y_tr,y_cr)
```

```
  0%|                                                                                           | 0/8 [00:00<?, ?it/s]

 12%|██████████                                                                         | 1/8 [00:27<03:10, 27.19s/it]

 25%|█████████████████████                                                              | 2/8 [02:35<05:44, 57.43s/it]

 38%|███████████████████████████████▌                                                   | 3/8 [06:48<09:41, 116.32s/it]

 50%|██████████████████████████████████████████                                        | 4/8 [16:32<17:06, 256.58s/it]

 62%|███████████████████████████████████████████████████▉                               | 5/8 [33:24<24:09, 483.16s/it]

 75%|██████████████████████████████████████████████████████████████▎                    | 6/8 [58:29<26:19, 789.70s/it]

 88%|██████████████████████████████████████████████████████████████████████          | 7/8 [1:42:55<22:32, 1352.65s/it]

100%|████████████████████████████████████████████████████████████████████████████████| 8/8 [2:33:07<00:00, 1148.43s/it]
```

5.3.2 Representation using 3D scatter plot

In [387]:
```
#from mpl_toolkits.mplot3d import Axes3D
plt.figure(figsize=(12,7))
ax = plt.axes(projection='3d')
zipped_=list(map(list,zip(es,de)))
x1=[i[0] for i in zipped_]
y1=[i[1] for i in zipped_]
# Data for three-dimensional scattered pointsf
# reference : https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html
ax.scatter3D(x1, y1, train_auc,label='Train')
ax.scatter3D(x1, y1, cv_auc,label='Cross Validation')
ax.set_xlabel('n_estimators')
ax.set_ylabel('Depth')
ax.set_zlabel('AUC')
ax.set_title('3D scatter plot on Train and CV')
ax.legend()
```

Out[387]:
```
<matplotlib.legend.Legend at 0x1922aa88b70>
```



5.3.3 Finding best parameter for training the model

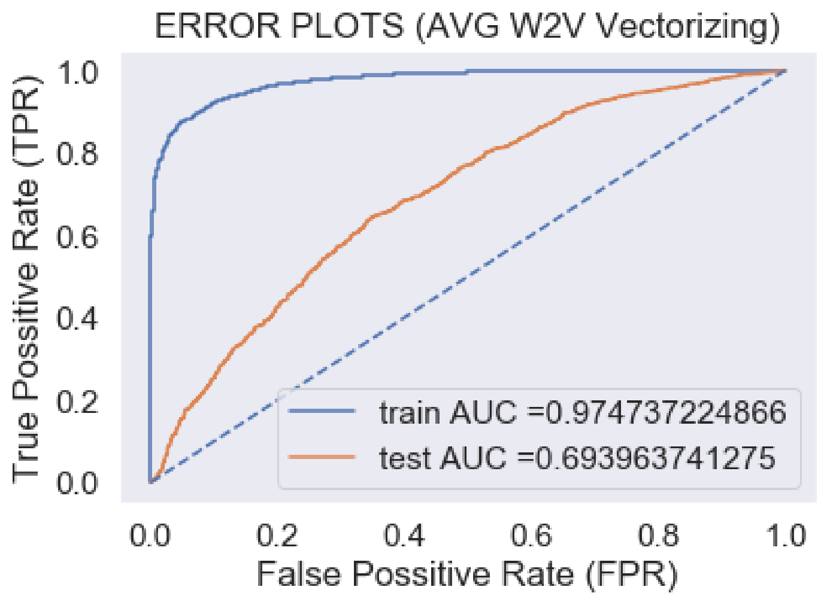
In [388]:
```
best_params=find_best_hyperparam(x1,y1,cv_auc)
best_params
```

```
Max_auc is 0.692991403363
```

Out[388]:
```
{'n_estimators': 1000, 'depth': 7}
```

In [389]:
```
RF=RandomForestClassifier(class_weight='balanced',n_estimators=best_params['n_estimators'],
                           max_depth=best_params['depth'],random_state=4)
RF.fit(X_tr,y_tr)

## Predict the test
train_predicts=RF.predict_proba(X_tr)[:,1]
test_predicts=RF.predict_proba(X_te)[:,1]

## Store fpr and tpr rates

train_fpr, train_tpr, tr_thresholds = roc_curve(y_tr, train_predicts)
test_fpr, test_tpr, te_thresholds = roc_curve(y_te, test_predicts)
```

In [390]:
```
#plot
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Possitive Rate (FPR)")
plt.ylabel("True Possitive Rate (TPR)")
plt.title("ERROR PLOTS (AVG W2V Vectorizing)")
plt.plot([0, 1], [0, 1],'b--')
plt.grid()
plt.show()


## Store auc results in a dictionary
results_dict.update({'AVGW2v' : {'trainauc':str(auc(train_fpr, train_tpr)),
                        'testauc': str(auc(test_fpr, test_tpr)),
                         'max_depth' : best_params['depth'],
                          'n_estimators' :best_params['n_estimators']} })
```

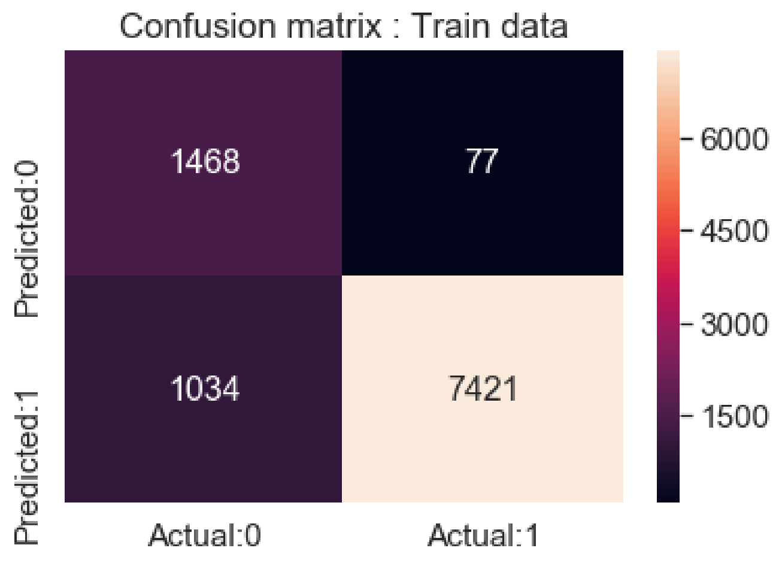


5.3.4 Confusion Matrix on train and test data

In [391]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t=best_threshold(tr_thresholds,train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_tr, predict_with_best_t(train_predicts, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_te, predict_with_best_t(test_predicts, best_t)))
```

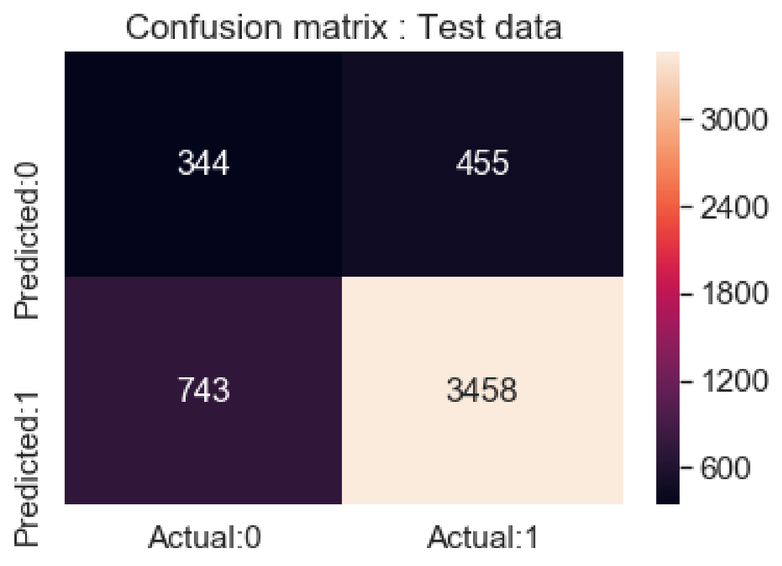
```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.833962248263 for threshold 0.537
Train confusion matrix
[[1468   77]
 [1034 7421]]
Test confusion matrix
[[ 344  455]
 [ 743 3458]]
```

In [392]:
```
### PLOT the matrix for Train
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_tr, predict_with_best_t(train_predicts, best_t))
                     , range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',
            xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Train data')
plt.show()
```



In [393]:

```python
### PLOT the matrix for Train
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_te, predict_with_best_t(test_predicts, best_t)), range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',xticklabels=['Actual:0','Actual:1']
            ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Test data')
plt.show()
```



5.4 Applying Random Forests on TFIDF W2V SET 4

In [394]:

```python
## converting to csr_matrix to avoid error : "could not broadcast input array from shape
## reference : https://stackoverflow.com/questions/24924940/convert-list-and-list-of-lists-to-scipy-sparse-arrays
tfidf_w2v_vectors_train=scipy.sparse.csr_matrix(tfidf_w2v_vectors_train)
tfidf_w2v_vectors_title_train=scipy.sparse.csr_matrix(tfidf_w2v_vectors_title_train)
tfidf_w2v_vectors_test=scipy.sparse.csr_matrix(tfidf_w2v_vectors_test)
tfidf_w2v_vectors_title_test=scipy.sparse.csr_matrix(tfidf_w2v_vectors_title_test)
tfidf_w2v_vectors_cv=scipy.sparse.csr_matrix(tfidf_w2v_vectors_cv)
tfidf_w2v_vectors_title_cv=scipy.sparse.csr_matrix(tfidf_w2v_vectors_title_cv)

```

In [395]:

```python

X_tr = hstack((vectorize_tr_trpr, vectorize_tr_proj,vectorize_tr_cat,vectorize_tr_subcat,
            vectorize_tr_sklst,tfidf_w2v_vectors_train,tfidf_w2v_vectors_title_train,
            sentiment_pos_train_norm,sentiment_neg_train_norm,sentiment_compound_train_norm,sentiment_neu_train_norm,
            X_train_price_norm,quantity_train_norm,prev_projects_train_norm,title_word_count_train_norm,
            essay_word_count_train_norm)).tocsr()

X_te = hstack((vectorize_te_trpr, vectorize_te_proj,vectorize_te_cat,vectorize_te_subcat,
            vectorize_te_sklst,tfidf_w2v_vectors_test,tfidf_w2v_vectors_title_test,
sentiment_pos_test_norm,sentiment_neg_test_norm,sentiment_compound_test_norm,sentiment_neu_test_norm,
            X_test_price_norm,quantity_test_norm,prev_projects_test_norm,title_word_count_test_norm,
            essay_word_count_test_norm)).tocsr()

X_cr = hstack((vectorize_cv_trpr, vectorize_cv_proj,vectorize_cv_cat,vectorize_cv_subcat,
            vectorize_cv_sklst,tfidf_w2v_vectors_cv,tfidf_w2v_vectors_title_cv,sentiment_pos_cv_norm,
            sentiment_neg_cv_norm,sentiment_compound_cv_norm,sentiment_neu_cv_norm,
            X_cv_price_norm,quantity_cv_norm,prev_projects_cv_norm,title_word_count_cv_norm,
            essay_word_count_cv_norm)).tocsr()


print(X_tr.shape)
print(X_te.shape)
print(X_cr.shape)
```

```
(22445, 619)
(16500, 619)
(11055, 619)
```

In [396]:

```python
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 619) (22445,)
(11055, 619) (11055,)
(16500, 619) (16500,)
====================================================================================================
```

In [397]:

```python
### taking only 20k points for this set from the randomly sampled 50k dataset.
X_tr=X_tr[:10000]
X_cr=X_cr[:5000]
X_te=X_te[:5000]
y_tr=y_train[:10000]
y_cr=y_cv[:5000]
y_te=y_test[:5000]
```

In [398]:

```python
print("Final Data matrix")
print(X_tr.shape, y_tr.shape)
print(X_cr.shape, y_cr.shape)
print(X_te.shape, y_te.shape)
print("="*100)
```

```
Final Data matrix
(10000, 619) (10000,)
(5000, 619) (5000,)
(5000, 619) (5000,)
====================================================================================================
```

5.4.1 Write loop to find best hyperparameters and do simple cross validation

In [399]:

```python
## Iterative looop to find best parameter
n_estimators = [10, 50, 100, 150, 200, 300, 500, 1000]
max_depth = [2, 3, 4, 5, 6, 7, 8, 9, 10,100,500]
train_auc,cv_auc,es,de=hyperparam_auc(max_depth,n_estimators,X_tr,X_cr,y_tr,y_cr)
```

```
  0%|                                                                                              | 0/8 [00:00<?, ?it/s]

 12%|███████████                                                                          | 1/8 [00:28<03:16, 28.05s/it]

 25%|█████████████████████                                                                | 2/8 [02:35<05:46, 57.83s/it]

 38%|███████████████████████████████▏                                                     | 3/8 [06:48<09:42, 116.40s/it]

 50%|██████████████████████████████████████████                                          | 4/8 [13:06<12:59, 194.82s/it]

 62%|████████████████████████████████████████████████████▋                                | 5/8 [21:21<14:14, 284.88s/it]

 75%|███████████████████████████████████████████████████████████████▊                     | 6/8 [34:00<14:14, 427.20s/it]

 88%|█████████████████████████████████████████████████████████████████████████▊           | 7/8 [54:51<11:14, 674.39s/it]

100%|██████████████████████████████████████████████████████████████████████████████████| 8/8 [1:37:25<00:00, 730.65s/it]
```

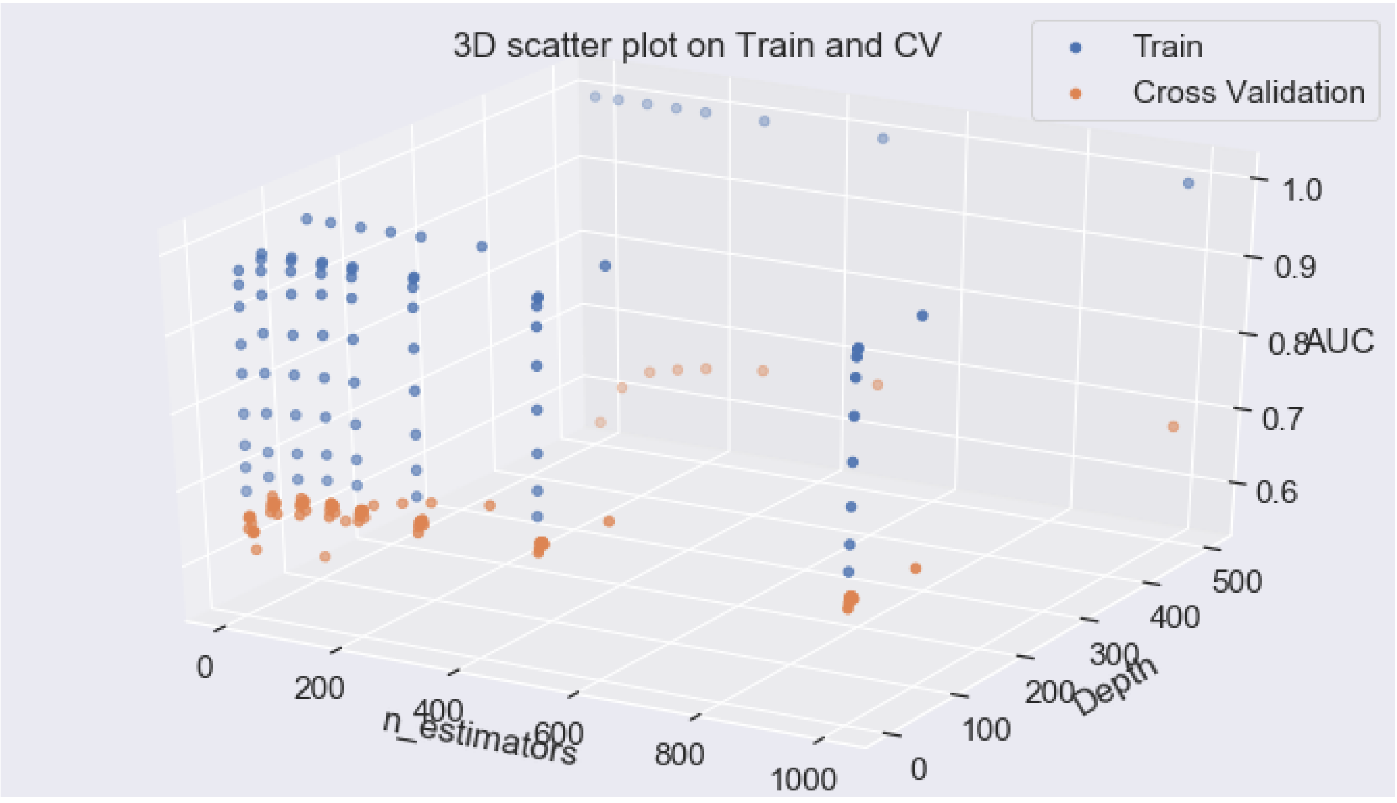
5.4.2 Representation using 3D scatter plot

In [400]:

```
#from mpl_toolkits.mplot3d import Axes3D
plt.figure(figsize=(12,7))
ax = plt.axes(projection='3d')
zipped_=list(map(list,zip(es,de)))
x1=[i[0] for i in zipped_]
y1=[i[1] for i in zipped_]
# Data for three-dimensional scattered points
# reference : https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html
ax.scatter3D(x1, y1, train_auc,label='Train')
ax.scatter3D(x1, y1, cv_auc,label='Cross Validation')
ax.set_xlabel('n_estimators')
ax.set_ylabel('Depth')
ax.set_zlabel('AUC')
ax.set_title('3D scatter plot on Train and CV')
ax.legend()
```

Out[400]:

```
<matplotlib.legend.Legend at 0x19228ee2da0>
```



5.4.3 Finding best parameter for training the model

In [401]:

```
best_params=find_best_hyperparam(x1,y1,cv_auc)
best_params
```

```
Max_auc is 0.695786664206
```

Out[401]:

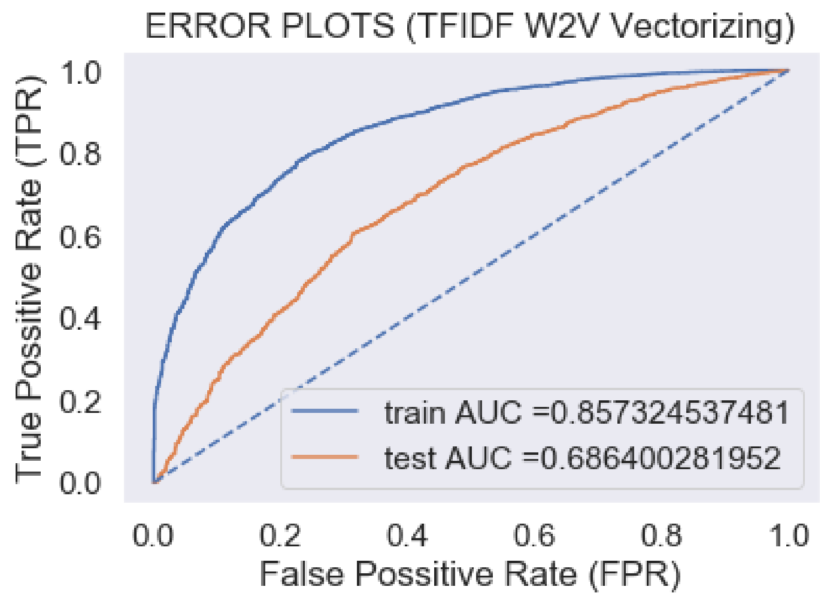
```
{'n_estimators': 150, 'depth': 5}
```

In [402]:

```
RF=RandomForestClassifier(class_weight='balanced',n_estimators=best_params['n_estimators'],
                              max_depth=best_params['depth'],random_state=4)
RF.fit(X_tr,y_tr)

## Predict the test
train_predicts=RF.predict_proba(X_tr)[:,1]
test_predicts=RF.predict_proba(X_te)[:,1]

## Store fpr and tpr rates

train_fpr, train_tpr, tr_thresholds = roc_curve(y_tr, train_predicts)
test_fpr, test_tpr, te_thresholds = roc_curve(y_te, test_predicts)
```

In [408]:

```
#plot
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Possitive Rate (FPR)")
plt.ylabel("True Possitive Rate (TPR)")
plt.title("ERROR PLOTS (TFIDF W2V Vectorizing)")
plt.plot([0, 1], [0, 1],'b--')
plt.grid()
plt.show()


## Store auc results in a dictionary
results_dict.update({'TFIDFW2V' : {'trainauc':str(auc(train_fpr, train_tpr)),
                        'testauc': str(auc(test_fpr, test_tpr)),
                         'max_depth' : best_params['depth'],
                          'n_estimators' :best_params['n_estimators']} })
```



5.4.4 Confusion Matrix on train and test data

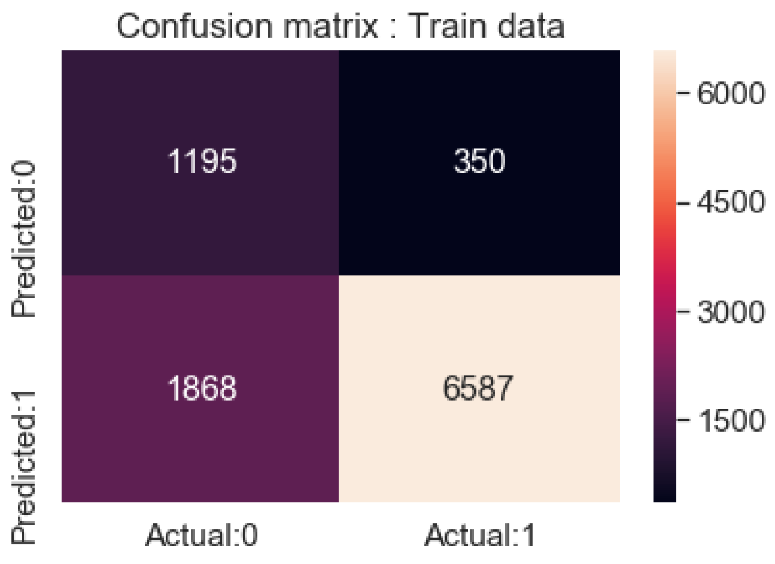
In [405]:

```
print("="*100)
#from sklearn.metrics import confusion_matrix
best_t=best_threshold(tr_thresholds,train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_tr, predict_with_best_t(train_predicts, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_te, predict_with_best_t(test_predicts, best_t)))
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.60257827945 for threshold 0.501
Train confusion matrix
[[1195  350]
 [1868 6587]]
Test confusion matrix
[[ 414  385]
 [1017 3184]]
```

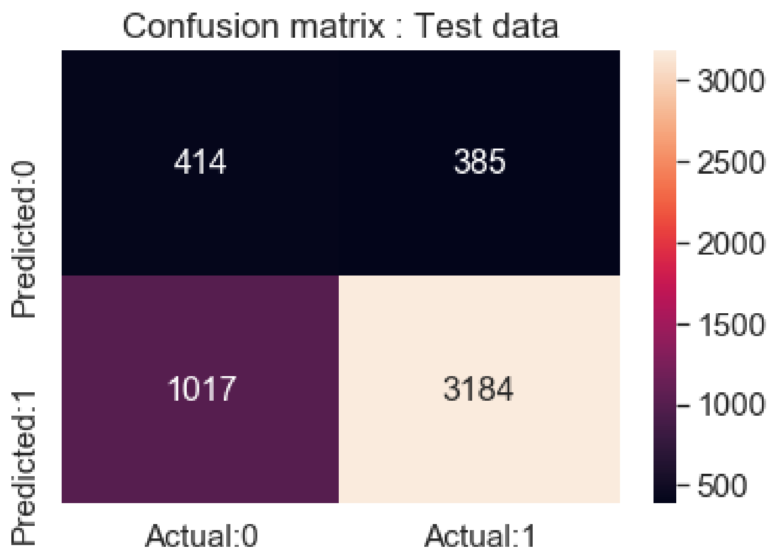
In [406]:

```
### PLOT the matrix for Train
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_tr, predict_with_best_t(train_predicts, best_t))
                     , range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',
            xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Train data')
plt.show()
```



In [407]:

```
### PLOT the matrix for Train
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_te, predict_with_best_t(test_predicts, best_t)), range(2), range(2))
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16},fmt='g',xticklabels=['Actual:0','Actual:1']
           ,yticklabels=['Predicted:0','Predicted:1']) # font size
plt.title('Confusion matrix : Test data')
plt.show()
```



In [ ]:

```

```