DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!</code`

Feature	Description
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 [1]:

```
## import all the modules
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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 wordcloud import WordCloud, STOPWORDS
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.linear_model import LogisticRegression
# 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 sklearn.externals.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
```

1. Reading the Data

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

In [3]:
```
## 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 [4]:
```
## 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[4]:

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

1.1 Preprocessing Categorical Features: project_grade_category

In [5]:
```
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 [6]:
```
# 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[6]:
```
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 [7]:
```
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 [8]:
```
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 [9]:
```
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 [10]:
```
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 [11]:
```
# 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 [12]:
```
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 [13]:
```
## 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 [14]:
```
project_data = pd.merge(project_data,price_data,on='id',how='left')
```

1.8 Preprocessing Text Features: project_title

In [15]:
```
# 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 [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'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 [17]:
```
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 [18]:
```
# 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 [19]:
```
preprocessed_titles = preprocess_text(project_data['project_title'].values)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248 [00:04<00:00, 22629.43it/s]
```

In [20]:
```
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 [21]:
```
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 inqui
sitive 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 b
ig 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 to provide each student an o
pportunity 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. Thes
e books will be used for read alouds, partner reading and for Independent reading. \r\nThey will engage in reading to build their \"Love for Reading\" by reading for pure enjoy
ment. They will be introduced to some new authors as well as some old favorites. I want my students to be ready for the 21st Century and know the pleasure of holding a good har
d 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 build 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 fro
m 7:30 am to 6:00 pm (2:30 to 6:00 pm in the after-school program), and they 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 si
blings, 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 manner. As a result, I have a community of students who love helping each other in and outside of the classroom. They consistently l
ook 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. Stu
dies 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, t
hey are able to explore and create in a welcoming environment. Alternative classroom seating has been experimented with more frequently in recent years. I believe (along with m
any others), that every child learns differently. This does not only apply to how multiplication 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 whe
rever 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 p
overty area and still come to school every day to get their education. I am trying to make it fun and educational for them so they can get the most out of their schooling. I cr
eated a caring environment for the students to bloom! They deserve the best.\r\nThank you!\r\nI am requesting 1 Chromebook to access online interventions, differentiate instruc
tion, 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 pa
rticipate 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 have 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 suc
cessful in school. Having access to technology in the classroom would help bridge the achievement gap.nannan
```

In [22]:
```
preprocessed_essays = preprocess_text(project_data['essay'].values)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248 [02:08<00:00, 848.11it/s]
```

In [23]:
```
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 not travel public library duty teacher provide student opportunity succeed every aspect life reading fund amental students read books boosting comprehension skills books used read alouds partner reading independent reading engage reading build love reading reading pure enjoyment in troduced 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 consid eration generous funding contribution help build 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 m eals breakfast lunch want students feel comfortable classroom home many students take multiple roles home well school sometimes caretakers younger siblings cooks babysitters ac ademics friends developing going become adults consider essential part job model helping others gain knowledge positive manner result community students love helping outside cl assroom consistently look opportunities support learning kind helpful way excited experimenting alternative seating classroom school year studies shown giving students option s it classroom increases focus well motivation allowing 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 applies space asked work students past ask work library w ork 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 e ducational get schooling created caring environment students bloom deserve best thank requesting 1 chromebook access online interventions differentiate instruction get extra pr actice chromebook used supplement ela math instruction students play ela math games engaging fun well participate assignments online turn help students improve skills chromeboo k classroom would not allow students use programs pace would ensure students getting adequate time use programs online programs especially beneficial students special needs abl e work level well challenged different materials making students confident abilities chromebook would allow students daily access computers increase computing skills change liv es better become successful school access technology classroom would help bridge achievement gap nannan
```

1.10 Preprocessing Text Features: Project Title

In [24]:
```
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 [25]:
```
processed_title = preprocess_text(project_data['project_title'].values)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248 [00:04<00:00, 21935.15it/s]
```

In [26]:
```
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 [27]:
```
## 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 [28]:
```
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 [29]:
```
# train test split using sklearn.model selection
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_
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 [30]:
```
## 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)
```

In [31]:
```
X_train.head(2)
```

Out[31]:

	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_resource_summary	teacher_number_of_previously_posted_projects	clean_categ
75742	p186156	f50f55a2b44b65b54f38f03c5df21922	mrs	tx	2017-03-01 16:21:46	9_12	My students need ipads for our annual art inst...	0	Music
61001	p180433	9e0fb5827f551d7e6966f8b3985e387b	ms	ny	2017-03-09 10:19:06	6_8	My students need a class set of headphones for...	0	Literacy_Lang

3. Vectorization on categorical and text features

3.1 One hot encoding on Categorical

In [32]:
```
# we use count vectorizer to convert the values into one hot vectors
## clean categories

cat_vectorize = CountVectorizer(lowercase=False, binary=True)
cat_vectorize.fit(X_train['clean_categories'].values)

train_categories = cat_vectorize.transform(X_train['clean_categories'].values)
test_categories = cat_vectorize.transform(X_test['clean_categories'].values)
cv_categories = cat_vectorize.transform(X_cv['clean_categories'].values)

print(cat_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",train_categories.shape)
print("Shape of matrix of Test data after one hot encoding ",test_categories.shape)
print("Shape of matrix of CV data after one hot encoding ",cv_categories.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language', 'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix of Train data after one hot encoding  (49041, 9)
Shape of matrix of Test data after one hot encoding  (36052, 9)
Shape of matrix of CV data after one hot encoding  (24155, 9)
```

In [33]:
```
# we use count vectorizer to convert the values into one hot vectors
## clean subcategories

subcat_vectorize = CountVectorizer(lowercase=False, binary=True)
subcat_vectorize.fit(X_train['clean_subcategories'].values)

train_subcategories = subcat_vectorize.transform(X_train['clean_subcategories'].values)
test_subcategories = subcat_vectorize.transform(X_test['clean_subcategories'].values)
cv_subcategories = subcat_vectorize.transform(X_cv['clean_subcategories'].values)

print(subcat_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",train_subcategories.shape)
print("Shape of matrix of Test data after one hot encoding ",test_subcategories.shape)
print("Shape of matrix of CV data after one hot encoding ",cv_subcategories.shape)

```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government', 'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics', 'EnvironmentalSc
ience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness', 'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writin
g', 'Mathematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'SocialSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
Shape of matrix of Train data after one hot encoding  (49041, 30)
Shape of matrix of Test data after one hot encoding  (36052, 30)
Shape of matrix of CV data after one hot encoding  (24155, 30)
```

In [34]:
```
# we use count vectorizer to convert the values into one hot vectors
##  school state

sklstate_vectorize = CountVectorizer(lowercase=False, binary=True)
sklstate_vectorize.fit(X_train['school_state'].values)

sklstate_train = sklstate_vectorize.transform(X_train['school_state'].values)
sklstate_test = sklstate_vectorize.transform(X_test['school_state'].values)
sklstate_cv = sklstate_vectorize.transform(X_cv['school_state'].values)

print(sklstate_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",sklstate_train.shape)
print("Shape of matrix of Test data after one hot encoding ",sklstate_test.shape)
print("Shape of matrix of CV data after one hot encoding ",sklstate_cv.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd',
'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
Shape of matrix of Train data after one hot encoding  (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of CV data after one hot encoding  (24155, 51)
```

In [35]:
```
# we use count vectorizer to convert the values into one hot vectors
## teacher_prefix

teacher_prefix_vectorize = CountVectorizer(lowercase=False, binary=True)
teacher_prefix_vectorize.fit(X_train['teacher_prefix'].values)

teacher_prefix_train = teacher_prefix_vectorize.transform(X_train['teacher_prefix'].values)
teacher_prefix_test = teacher_prefix_vectorize.transform(X_test['teacher_prefix'].values)
teacher_prefix_cv = teacher_prefix_vectorize.transform(X_cv['teacher_prefix'].values)

print(teacher_prefix_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",teacher_prefix_train.shape)
print("Shape of matrix of Test data after one hot encoding ",teacher_prefix_test.shape)
print("Shape of matrix of CV data after one hot encoding ",teacher_prefix_cv.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix of Train data after one hot encoding  (49041, 5)
Shape of matrix of Test data after one hot encoding  (36052, 5)
Shape of matrix of CV data after one hot encoding  (24155, 5)
```

In [36]:
```
# we use count vectorizer to convert the values into one hot vectors
## project_grade

proj_grade_vectorize = CountVectorizer(lowercase=False, binary=True)
proj_grade_vectorize.fit(X_train['project_grade_category'].values)

proj_grade_train = proj_grade_vectorize.transform(X_train['project_grade_category'].values)
proj_grade_test = proj_grade_vectorize.transform(X_test['project_grade_category'].values)
proj_grade_cv = proj_grade_vectorize.transform(X_cv['project_grade_category'].values)

print(proj_grade_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",proj_grade_train.shape)
print("Shape of matrix of Test data after one hot encoding ",proj_grade_test.shape)
print("Shape of matrix of CV data after one hot encoding ",proj_grade_cv.shape)
```

```
['3_5', '6_8', '9_12', 'prek_2']
Shape of matrix of Train data after one hot encoding  (49041, 4)
Shape of matrix of Test data after one hot encoding  (36052, 4)
Shape of matrix of CV data after one hot encoding  (24155, 4)
```

3.2 Vectorizing Text data

3.2.1 TFIDF on Essay data

In [37]:
```
 ##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  (49041, 5000)
Shape of matrix after one hot encoding  (36052, 5000)
Shape of matrix after one hot encoding  (24155, 5000)
```

3.2.2 TFIDF on Title data

In [38]:
```
##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  (49041, 2008)
Shape of matrix after one hot encoding  (36052, 2008)
Shape of matrix after one hot encoding  (24155, 2008)
```

3.2.3 Weighted tfidf on Essay data using Pretrained Models

In [39]:

```
# 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 [40]:

```
# 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)
```

```
49041
300
24155
300
36052
300
```

3.2.5 Weighted tfidf on Title data using Pretrained Models

In [41]:
```
# 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)

```

```
49041
300
24155
300
36052
300
```

4. Vectorizing Numerical Features

4.1 Price

In [42]:
```
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, 49041) (49041,)
(1, 24155) (24155,)
(1, 36052) (36052,)
====================================================================================================
```

4.2 Quantity

In [43]:

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

4.3 Number of Previously posted projects

In [44]:

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

4.4 Title Word counts

In [45]:
```
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, 49041) (49041,)
(1, 24155) (24155,)
(1, 36052) (36052,)
====================================================================================================
```

4.5 Essay Words Counts

In [46]:
```
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, 49041) (49041,)
(1, 24155) (24155,)
(1, 36052) (36052,)
```

4.6 Vectorizing sentiment Columns

In [47]:
```
### 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)
```

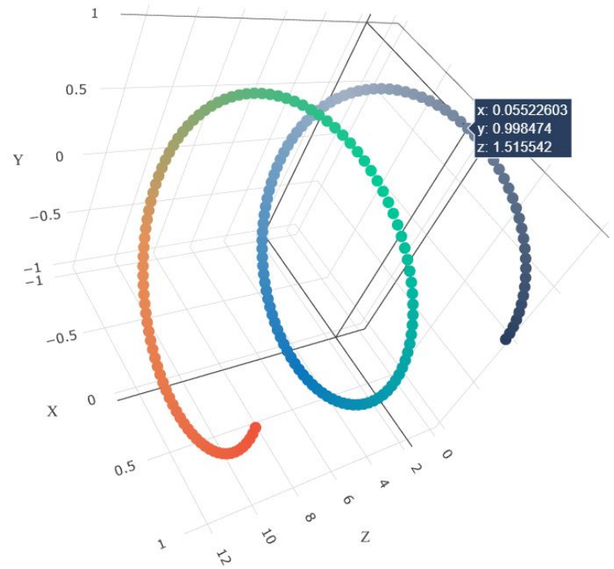
In [48]:
```
### vectorize neg
Normalize_neg=Normalizer()
Normalize_neg.fit(X_train['neg'].values.reshape(1,-1))
sentiment_neg_train_norm=Normalize_neg.transform(X_train['neg'].values.reshape(1,-1))
sentiment_neg_test_norm=Normalize_neg.transform(X_test['neg'].values.reshape(1,-1))
sentiment_neg_cv_norm=Normalize_neg.transform(X_cv['neg'].values.reshape(1,-1))
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 [49]:
```
### 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 [50]:
```
### 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 10:Decision Trees

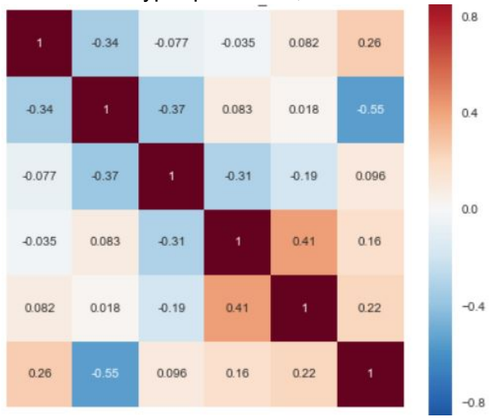
1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
   - Set 1: categorical, numerical features + preprocessed_eassay (TFIDF)
   - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF W2V)
2. **The hyper paramter tuning (best `depth` in range [1, 5, 10, 50], and the best `min_samples_split` in range [5, 10, 100, 500])**
   - 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(use gridsearch cv or randomsearch cv)/simple cross validation data(you can write your own for loops refer sample solution)
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 **min_sample_split**, 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*

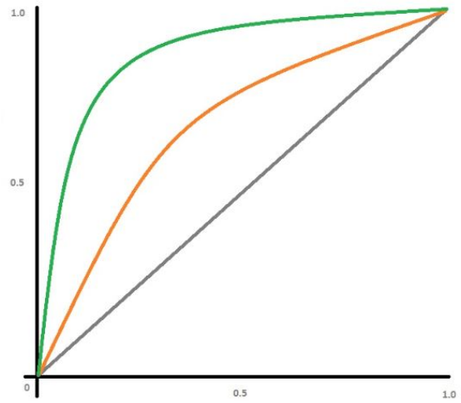
or

- 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 choose either of the plotting techniques out of 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 = ??

- Once after you plot the confusion matrix with the test data, get all the `false positive data points`
  - Plot the WordCloud(https://www.geeksforgeeks.org/generating-word-cloud-python/) with the words of essay text of these `false positive data points`
  - Plot the box plot with the `price` of these `false positive data points`
  - Plot the pdf with the `teacher_number_of_previously_posted_projects` of these `false positive data points`

4. **Task 2:** For this task consider set-1 features. Select all the features which are having non-zero feature importance.You can get the feature importance using 'feature_importances_` (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html), discard the all other remaining features and then apply any of the model of you choice i.e. (Dession tree, Logistic Regression, Linear SVM), you need to do hyperparameter tuning corresponding to the model you selected and procedure in step 2 and step 3
   Note: when you want to find the feature importance make sure you don't use max_depth parameter keep it None.

5. You need to summarize the results at the end of the notebook, summarize it in the table format

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 SET1 : categorical, numerical features + project_title(TFIDF) + preprocessed_eassay (TFIDF)

In [51]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
X_tr = hstack((train_categories, train_subcategories,sklstate_train,teacher_prefix_train,
            proj_grade_train,tfidf_essay_train,tfidf_title_train,
            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((test_categories, test_subcategories,sklstate_test,teacher_prefix_test,
            proj_grade_test,tfidf_essay_test,tfidf_title_test,
            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((cv_categories, cv_subcategories,sklstate_cv,teacher_prefix_cv,
            proj_grade_cv,tfidf_essay_cv,tfidf_title_cv,
            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)
```

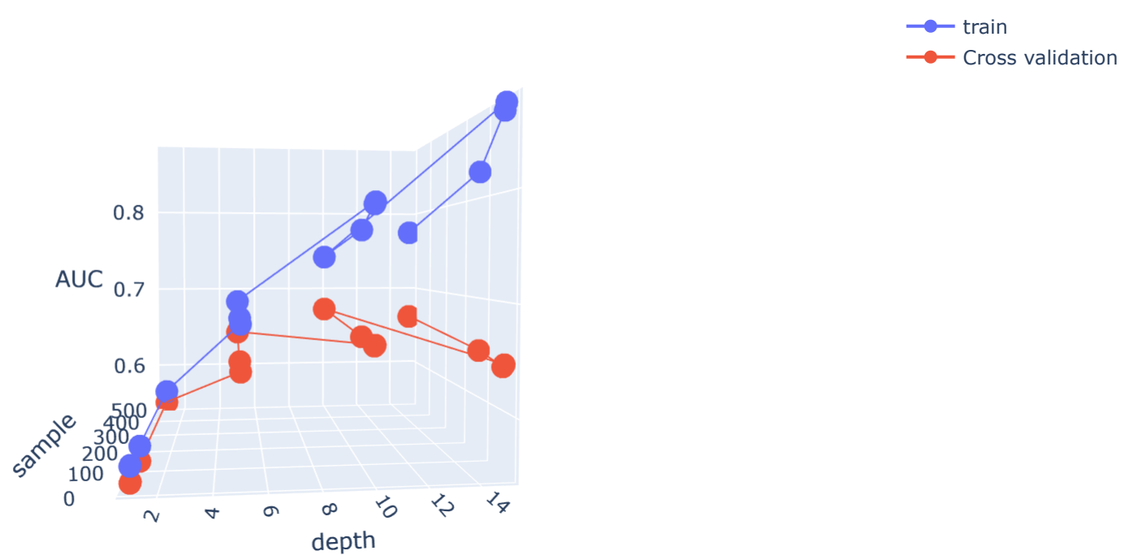
```
(49041, 7112)
(36052, 7112)
(24155, 7112)
```

5.1 Write own function to find which hyperparameter gives maximum auc

In [52]:
```
## Iterative looop to find best parameter
depth=[1,5,10,15]
min_samp=[5,10,100,500]

def hyperparam_auc(depth,min_samp,X_tr,X_cr,y_train,y_cv):
    train_auc=[]
    cv_auc=[]
    for dep in depth:
        for min_sample in min_samp:
            tree=DecisionTreeClassifier(max_depth=dep,min_samples_split=min_sample,
                                        class_weight='balanced',
                                        random_state=4)
            tree.fit(X_tr,y_train)
            #predict train and cv
            train_predicts=tree.predict_proba(X_tr)[:,1]
            cv_predicts=tree.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))
    return train_auc,cv_auc

train_auc,cv_auc=hyperparam_auc(depth,min_samp,X_tr,X_cr,y_train,y_cv)
```

In [53]:

```
### Visualizing the best hyperparameters value where the auc is maximum
# https://plot.ly/python/3d-axes/
x1=[1,1,1,1,5,5,5,5,10,10,10,10,15,15,15,15]
y1=[5,10,100,500,5,10,100,500,5,10,100,500,5,10,100,500]
trace1 = go.Scatter3d(x=x1,y=y1,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=x1,y=y1,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

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

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



In [54]:
```
## 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={'depth':i[0],'min_sample_size':i[1]}
        else:
            pass
    return best_params
```

In [55]:
```
best_params=find_best_hyperparam(x1,y1,cv_auc)
best_params
```

```
Max_auc is 0.673337205471
```

Out[55]:
```
{'depth': 10, 'min_sample_size': 500}
```

Observations :

1. We see that we get the max_auc of 0.67 in the CV is observed where depth is 10 and min_sample_size is 500

Find best hyper params using Grid Searchcv with cv=5

In [56]:
```
## Consider alpha values between 0.0001 to 10  (L1 penalty)
parameters={"max_depth" : [1,5,10,15] , 'min_samples_split' : [5,10,100,500],
            'class_weight':['balanced']
            ,'random_state':[4]}
clf = GridSearchCV(DecisionTreeClassifier(), parameters, cv=5,
                   scoring='roc_auc',verbose=1,n_jobs=3)
clf.fit(X_tr, y_train)
results = pd.DataFrame.from_dict(clf.cv_results_)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits

[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Done  44 tasks      | elapsed:  1.4min
[Parallel(n_jobs=3)]: Done  80 out of  80 | elapsed:  5.9min finished
```

In [57]:
```
params=results['params']
train_auc=results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
```

In [58]:
```
best_C = clf.best_params_
print('Best hyperparameter as a result of Grid Search',best_C)
```

```
Best hyperparameter as a result of Grid Search {'class_weight': 'balanced', 'max_depth': 10, 'min_samples_split': 500, 'random_state': 4}
```

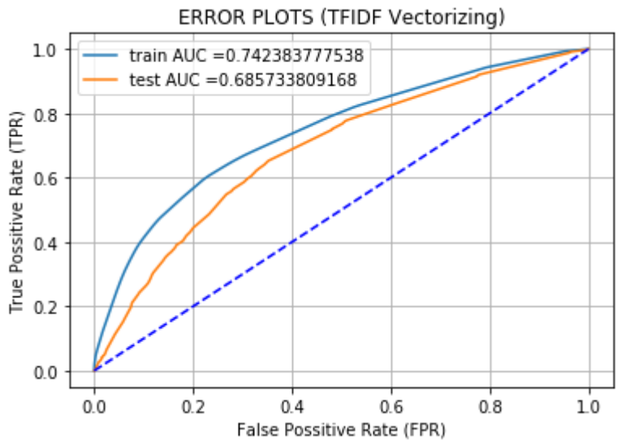
Observations:

1.Using grid search that the optimal max_depth is 5 and min_samples_split is 500

5.2 Training the Decision Tree with best hyper parameters (max_depth,min_samples_split)

In [59]:

```
tree=DecisionTreeClassifier(class_weight='balanced',max_depth=best_C['max_depth'],
                            min_samples_split=best_C['min_samples_split'],random_state=4)
tree.fit(X_tr,y_train)

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

#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={'TFIDF' : {'trainauc':str(auc(train_fpr, train_tpr)),
                         'testauc': str(auc(test_fpr, test_tpr)),
                          'max_depth' : best_C['max_depth'],
                           'min_sample_split' :best_C['min_samples_split']} }
```



Observations :

1. Model performs far better on train than test with test AUC found to be 0.68 and Train AUC as 0.74 after training model using optimal depth 10 and min sample split as 500 and vectorizing text data using TFIDF vectorizing .

Visualizing the decision tree trained with best hyperparameters

In [60]:

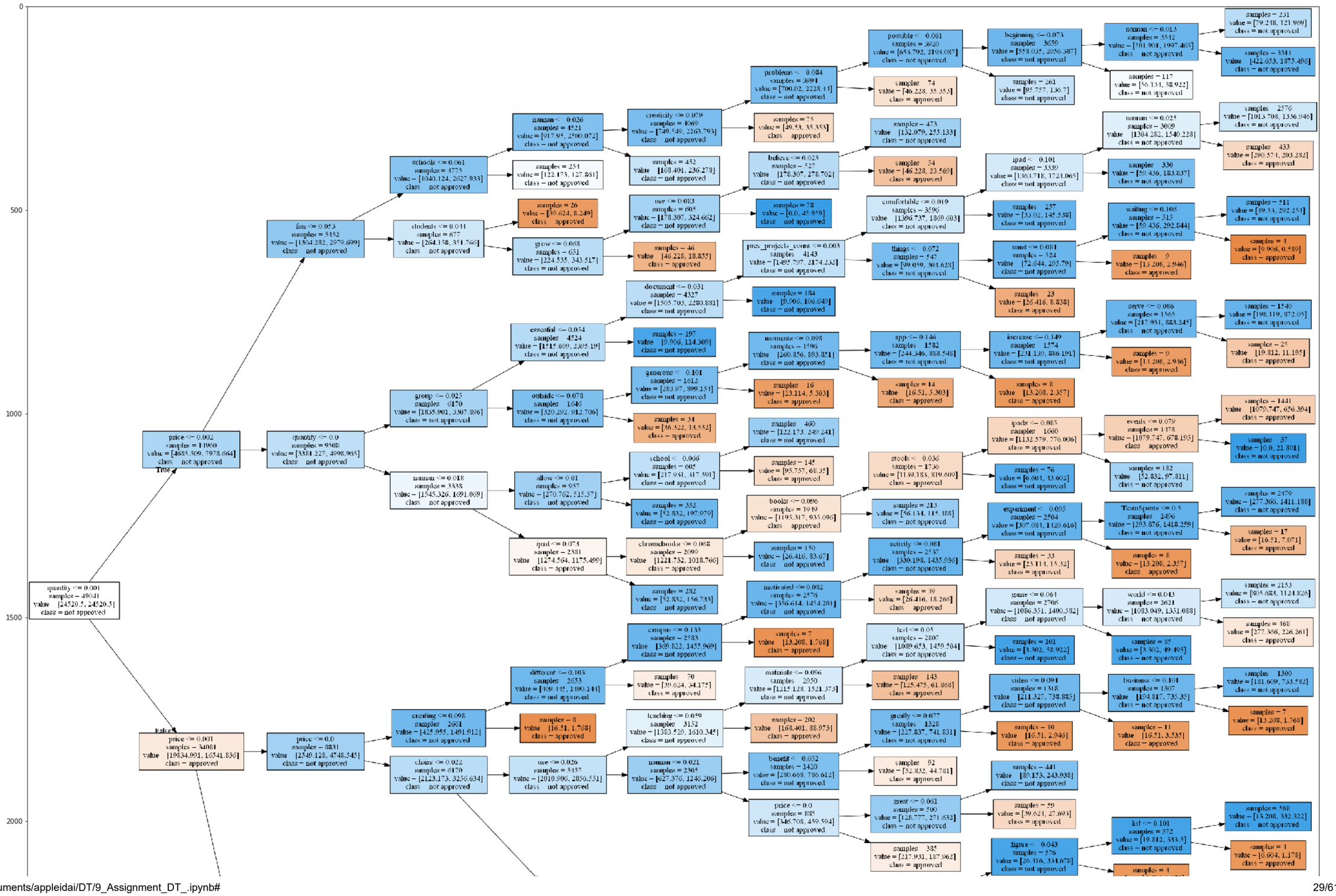
```
#reference :https://medium.com/@rnbrown/creating-and-visualizing-decision-trees-with-python-f8e8fa394176
cat_vectorize.get_feature_names()
feature_names= [cat_vectorize.get_feature_names()+subcat_vectorize.get_feature_names()+
                sklstate_vectorize.get_feature_names()+teacher_prefix_vectorize.get_feature_names()+
               proj_grade_vectorize.get_feature_names()+
                tfidf_essay.get_feature_names()+tfidf_title.get_feature_names()+
               ['price']+['quantity']+['prev_projects_count']+['title_word_count']+
               ['essay_word_count']]

#system("dot -Tpng D:.dot -o D:/dtree2.png")
dot_data=export_graphviz(tree, out_file="dttree.dot", class_names=["approved", "not approved "],
                         feature_names=feature_names[0], impurity=False,
                         filled=True,rotate=True)

## Use shell command to store the graph as a png file
## refernece https://stackoverflow.com/questions/1494492/graphviz-how-to-go-from-dot-to-a-graph
! dot -Tpng dttree.dot > treegraph1.png
```

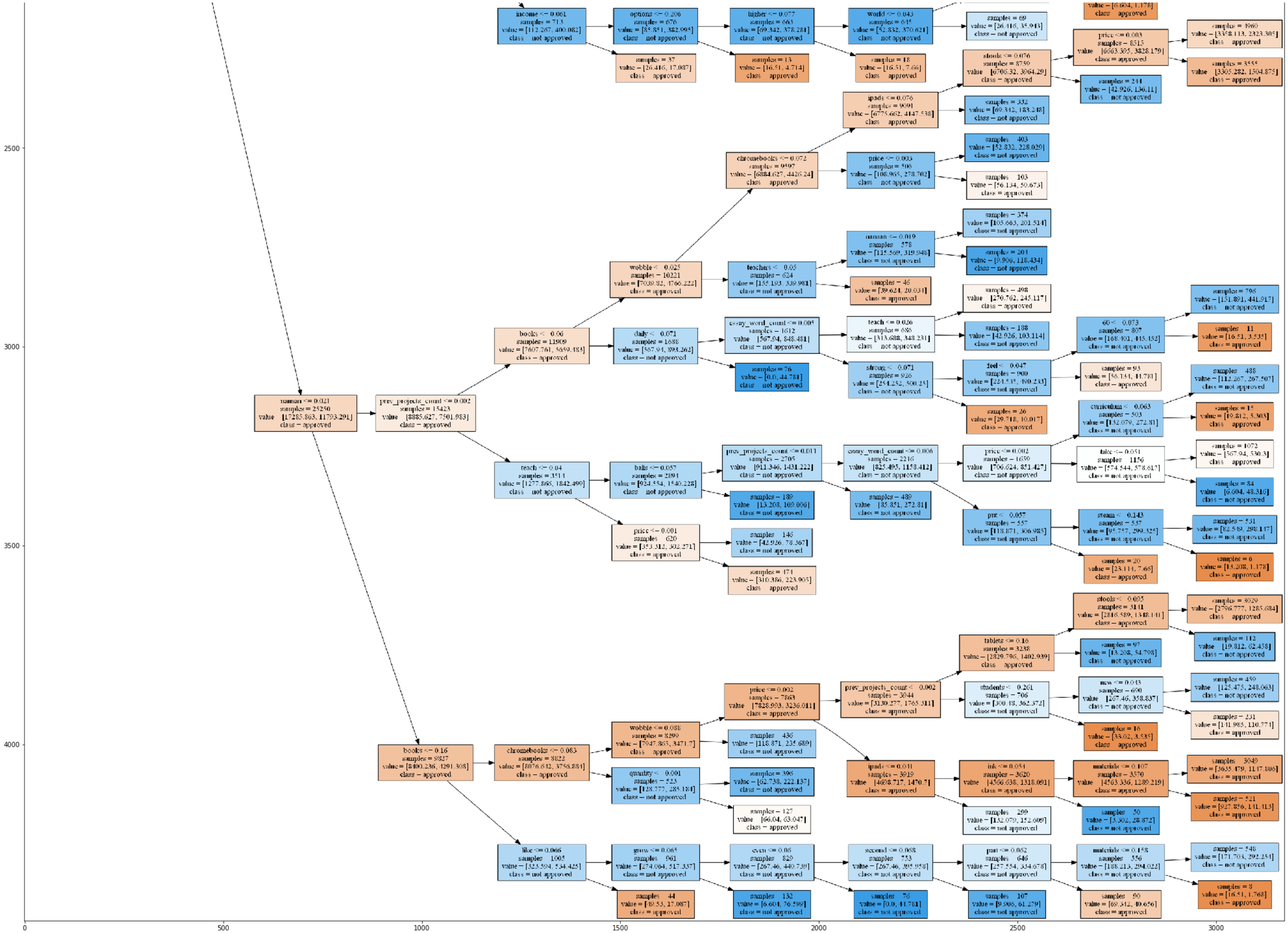
Visualizing Decision tree after training the decision tree with best hyperparameters

1. Only a part of the decision tree is visualized here.

In [61]:
```
## Final decision tree visualization
# reference :https://stackoverflow.com/questions/35286540/display-an-image-with-python
import matplotlib.image as mpimg
plt.figure(figsize=(50,50))
plt.imshow(mpimg.imread('treegraph1.png'))
```

Out[61]: <matplotlib.image.AxesImage at 0x26b82d181d0>





5.3 Confusion Matrix

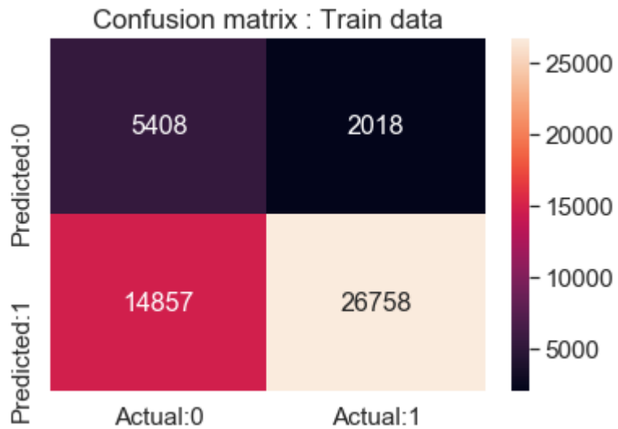
5.3.1 Train data

In [62]:
```
## 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 [63]:
```
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)))
```

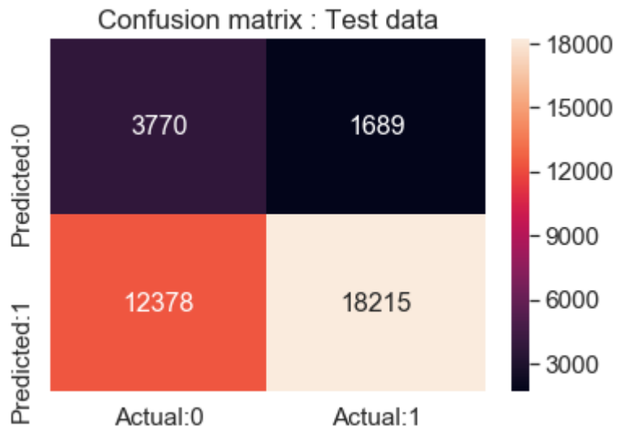
```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.46825830472 for threshold 0.474
Train confusion matrix
[[ 5408  2018]
 [14857 26758]]
Test confusion matrix
[[ 3770  1689]
 [12378 18215]]
```

In [64]:
```
### 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()
```



5.3.2 Test data

In [65]:
```
### 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()
```



Observations :

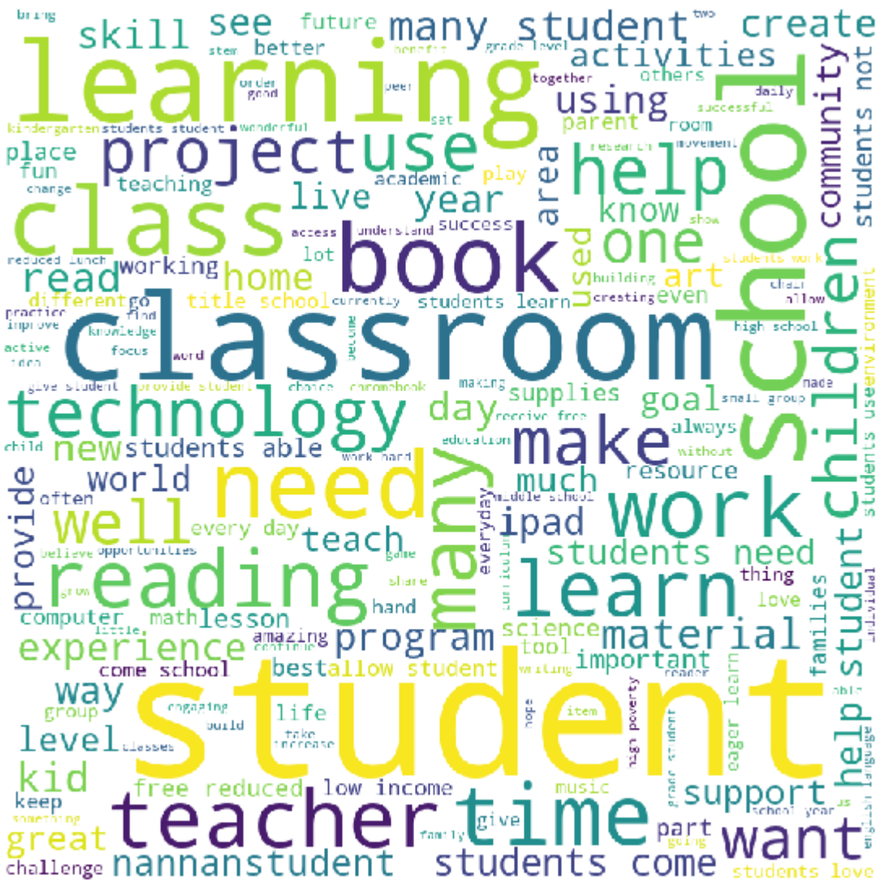
1.We can observe from train and test we are getting majority True positives
2.Least number of data falls in False negative,which refers as least number of projects were incorrectly predicted as not approved in both Test and Train.
3.For a model to perform well we need High True Positive Rate and Low False Positive Rate.From the above our train data has True Positive Rate as 93% and False Positive Rate as 73.3%
4.In our test data : True Positive Rate as 91.51% and False Positive Rate as 76.6%.

Word Cloud for false possitives

In [66]:

```
def word_cloud(y_test,test_predicts,best_t):
    ## create a dataframe which has all the points which are false possitive
    fp_i=[]
    stopwords = set(STOPWORDS)
    pred_=predict_with_best_t(test_predicts, best_t)
    actuals=list(y_test)
    df_test_predicts=pd.DataFrame(data={'predicted': pred_,'actual':actuals})
    ##save the index of the datapoints
    for i,row in df_test_predicts.iterrows():
        if row['predicted'] ==1 and row['actual']==0:
            fp_i.append(i)
    ##merge the text of false positive points
    test_essay=X_test['processed_essay'].values
    word_cloud_str=''
    for i in fp_i:
        word_cloud_str+=test_essay[i]
    #create wordcloud
    wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(word_cloud_str)
    return wordcloud,fp_i
```

In [67]:
```
# plot the WordCloud image
wordcloud,fp_i=word_cloud(y_test,test_predicts,best_t)
plt.figure(figsize = (8, 8), facecolor ="none")
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observations :

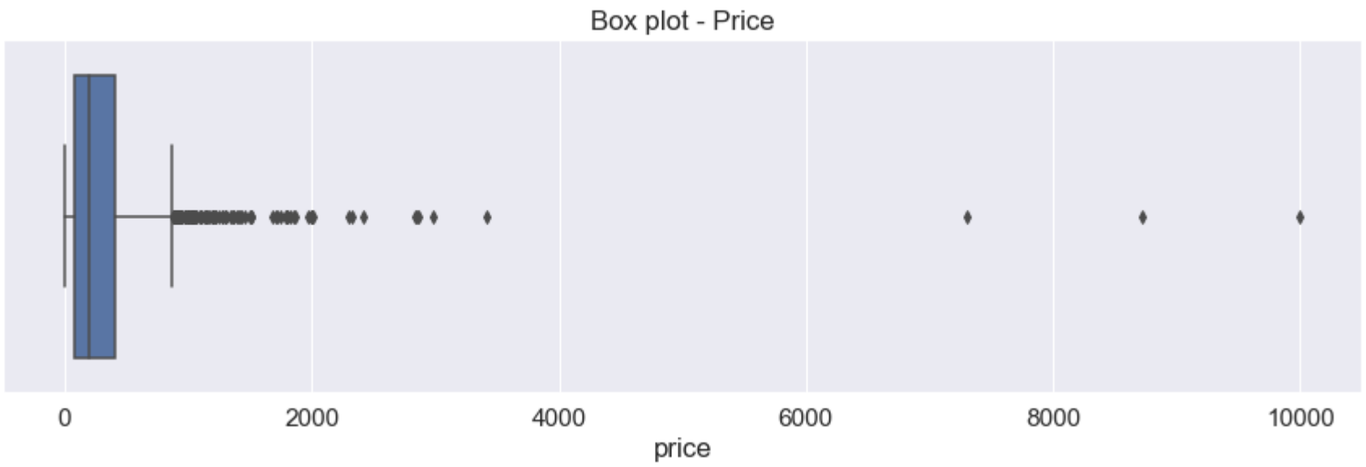
1. The data points which were false possitives were incorrectly classified as positive due to words like learning ,classroom,student etc.

BOX plot for price column

In [68]:
```
test_price=X_test.values
fp_df=pd.concat([pd.DataFrame([test_price[i]],columns=X_test.columns) for i in fp_i ],ignore_index=True)
```

In [69]:
```
plt.figure(figsize=(15,4))
sns.boxplot(x="price",data=fp_df)
plt.title('Box plot - Price')
```

Out[69]: Text(0.5, 1.0, 'Box plot - Price')



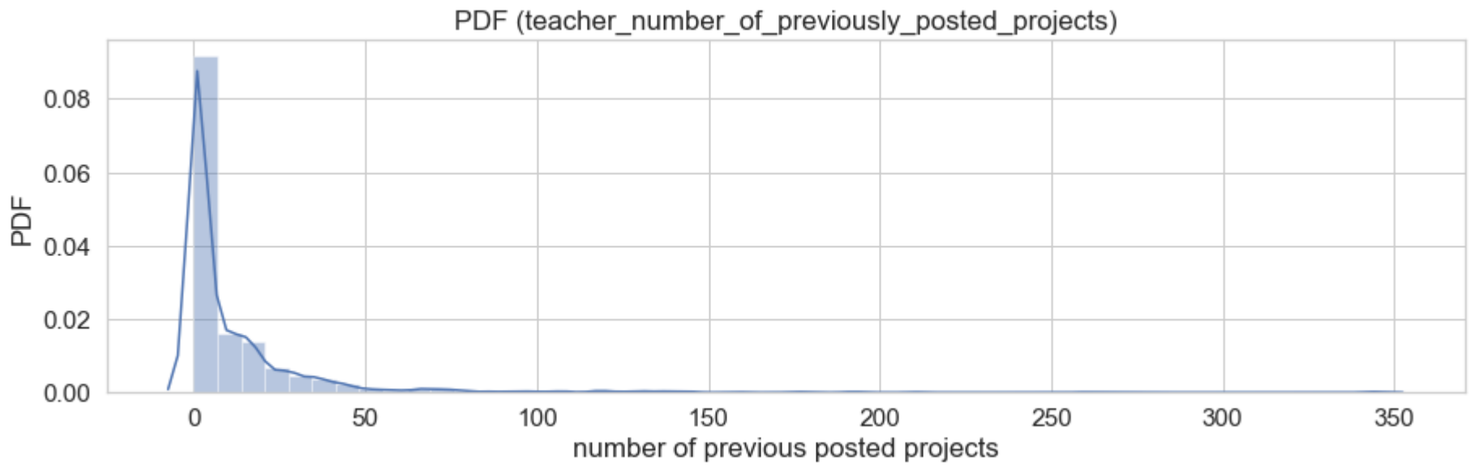
Observations :

1. We see that most false positive points have very less price.
2. There are few projects which had higher than 2000 as price that was not approved but our model predicts as approved,these points are outlier points .

PDF for number_previous_posted_project for false positive points

In [70]:
```
sns.set_style('whitegrid')
plt.figure(figsize=(15,4))
sns.distplot(fp_df['teacher_number_of_previously_posted_projects'])
plt.xlabel('number of previous posted projects')
plt.ylabel('PDF')
plt.title('PDF (teacher_number_of_previously_posted_projects)')
```

Out[70]: Text(0.5, 1.0, 'PDF (teacher_number_of_previously_posted_projects)')



Observations :

1.Most false positive points had number of previously posted projects to be less than 25

6 SET2 : categorical, numerical features + project_title(TFIDF w2v) + preprocessed_eassay (TFIDF w2v)

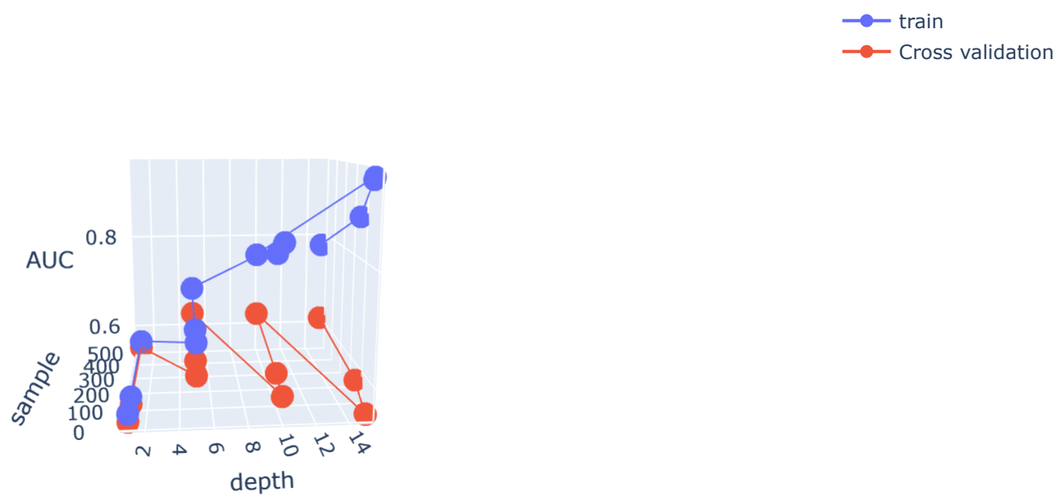
In [71]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039


X_tr = hstack((train_categories, train_subcategories,sklstate_train,teacher_prefix_train,
            proj_grade_train,tfidf_w2v_vectors_train,tfidf_w2v_vectors_title_train,
            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((test_categories, test_subcategories,sklstate_test,teacher_prefix_test,
            proj_grade_test,tfidf_w2v_vectors_test,tfidf_w2v_vectors_title_test,
            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((cv_categories, cv_subcategories,sklstate_cv,teacher_prefix_cv,
            proj_grade_cv,tfidf_w2v_vectors_cv,tfidf_w2v_vectors_title_cv,
            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)
```

```
(49041, 704)
(36052, 704)
(24155, 704)
```

6.1 Execute own function to find which hyperparameter gives maximum auc

In [72]:
```
## Use hyperparam_auc function to find auc's for multiple hyperparameters
depth=[1,5,10,15]
min_samp=[5,10,100,500]
train_auc,cv_auc=hyperparam_auc(depth,min_samp,X_tr,X_cr,y_train,y_cv)
```

In [73]:
```
### Visualizing the best hyperparameters value where the auc is maximum
# https://plot.ly/python/3d-axes/
x1=[1,1,1,1,5,5,5,5,10,10,10,10,15,15,15,15]
y1=[5,10,100,500,5,10,100,500,5,10,100,500,5,10,100,500]
trace1 = go.Scatter3d(x=x1,y=y1,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=x1,y=y1,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

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

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



In [74]:
```
best_params=find_best_hyperparam(x1,y1,cv_auc)
best_params
```

```
Max_auc is 0.630024910492
```

Out[74]:
```
{'depth': 5, 'min_sample_size': 100}
```

Observations :

1. We see that we get the max_auc of 0.63 in the CV is observed
where depth is 5 and min_sample_size is 100

Find best hyper params using Grid Searchcv with cv=5

In [75]:
```
parameters={"max_depth" : [1,5,10,15] , 'min_samples_split' : [5,10,100,500],
            'class_weight':['balanced'],'random_state':[4]}
clf = GridSearchCV(DecisionTreeClassifier(), parameters, cv=5,
                   scoring='roc_auc',verbose=1,n_jobs=3)
clf.fit(X_tr, y_train)
results = pd.DataFrame.from_dict(clf.cv_results_)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits

[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Done  44 tasks      | elapsed:  9.2min
[Parallel(n_jobs=3)]: Done  80 out of  80 | elapsed: 36.9min finished
```

In [76]:
```
params=results['params']
train_auc=results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
```

In [77]:
```
best_C = clf.best_params_
print('Best hyperparameter as a result of Grid Search',best_C)
```

```
Best hyperparameter as a result of Grid Search {'class_weight': 'balanced', 'max_depth': 5, 'min_samples_split': 500, 'random_state': 4}
```

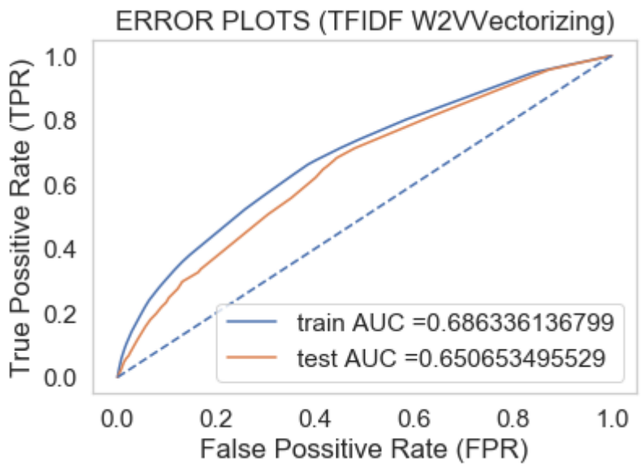
Observations:

1.Using grid search we found that the optimal max_depth is 5 and
min_samples_split is 500.

6.2 Training the Decision Tree with best hyper parameters (max_depth,min_samples_split)

In [78]:

```
tree=DecisionTreeClassifier(class_weight='balanced',max_depth=best_C['max_depth'],
                            min_samples_split=best_C['min_samples_split'],random_state=4)
tree.fit(X_tr,y_train)

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

#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 W2VVectorizing)")
plt.plot([0, 1], [0, 1],'b--')
plt.grid()
plt.show()


## Store auc results in a dictionary
results_dict.update({'TFIDF W2V' : {'trainauc':str(auc(train_fpr, train_tpr)),
                        'testauc': str(auc(test_fpr, test_tpr)),
                        'max_depth' : best_C['max_depth'],
                          'min_sample_split' :best_C['min_samples_split']   } })
```



Observations :

1. Model performs far better on train than test with test AUC found to be 0.65 and Train AUC as 0.68 after training model using optimal depth 10 and min sample split as 500 and vectorizing text data using TFIDF W2V vectorizing .

Visualizing the decision tree trained with best hyperparameters

In [79]:

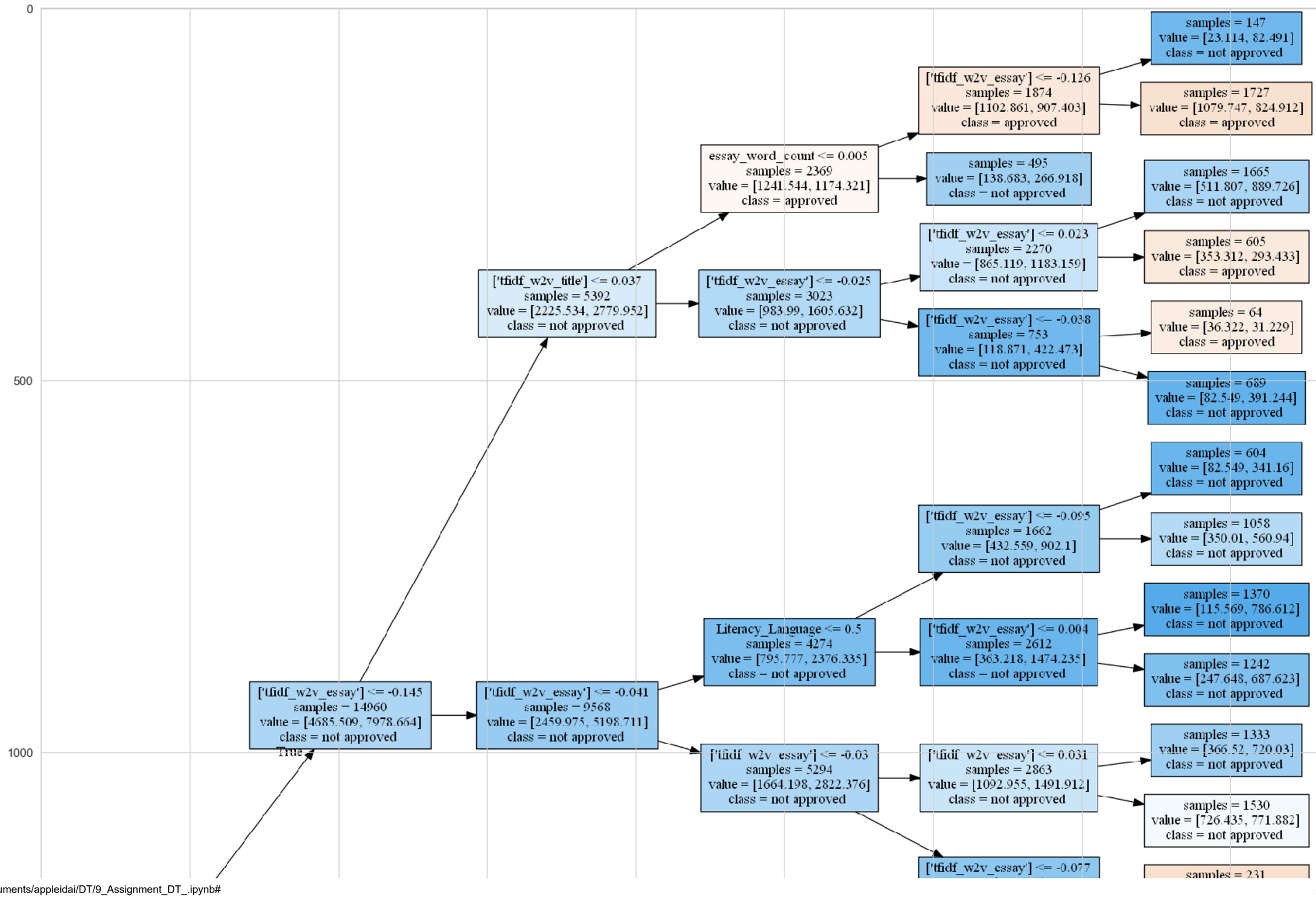
```
#reference :https://medium.com/@rnbrown/creating-and-visualizing-decision-trees-with-python-f8e8fa394176

feature_names= [cat_vectorize.get_feature_names()+subcat_vectorize.get_feature_names()+
                sklstate_vectorize.get_feature_names()+
                teacher_prefix_vectorize.get_feature_names()+
               proj_grade_vectorize.get_feature_names()+ [['tfidf_w2v_essay']] * 300 +
                [['tfidf_w2v_title']] *300  +
               ['price']+['quantity']+['prev_projects_count']+
                ['title_word_count']+['essay_word_count']]

#system("dot -Tpng D:.dot -o D:/dtree2.png")
dot_data=export_graphviz(tree, out_file="dttree.dot", class_names=["approved", "not approved "],feature_names=feature_names[0], impurity=False,
                         filled=True,rotate=True)

## Use shell command to store the graph as a png file
! dot -Tpng dttree.dot > treegraph2.png
```

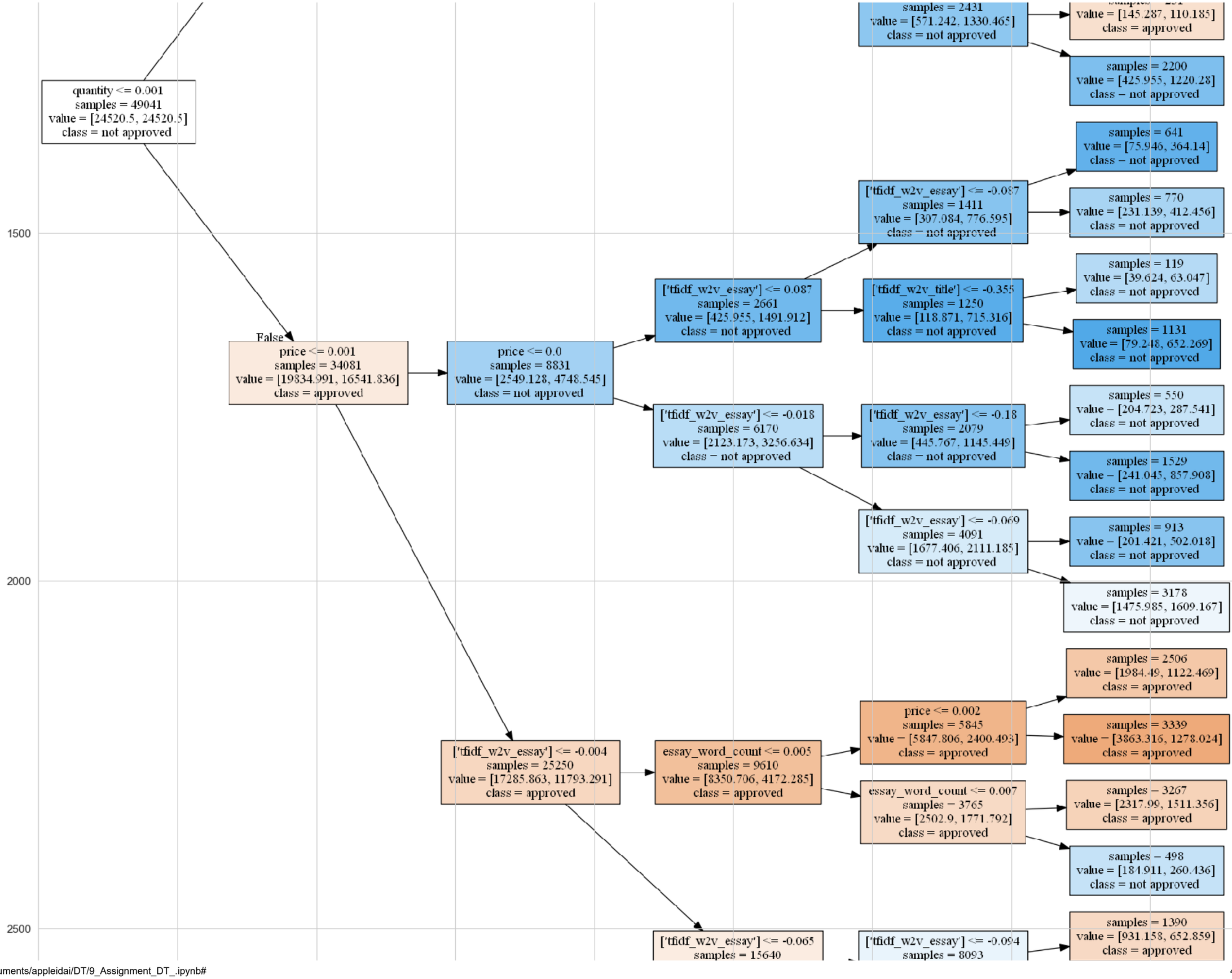
Visualizing Decision tree after training the decision tree with best hyperparameters

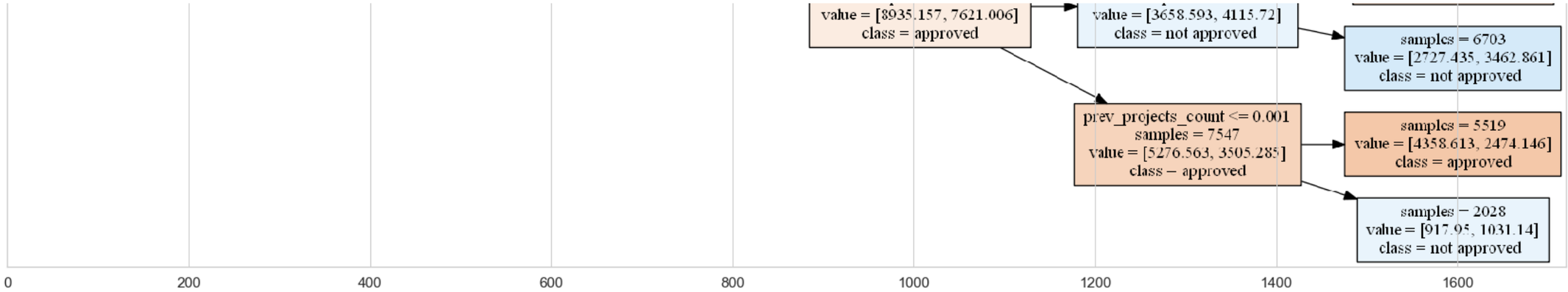
1. Only a part of the decision tree is visualized here.

In [80]:
```
## Final decision tree visualization
# reference :https://stackoverflow.com/questions/35286540/display-an-image-with-python
import matplotlib.image as mpimg
plt.figure(figsize=(50,50))
plt.imshow(mpimg.imread('treegraph2.png'))
```

Out[80]: <matplotlib.image.AxesImage at 0x26b82e0d080>




quantity <= 0.001
samples = 49041
value = [24520.5, 24520.5]
class = not approved
True
False
price <= 0.001
samples = 34081
value = [19834.991, 16541.836]
class = approved
price <= 0.0
samples = 8831
value = [2549.128, 4748.545]
class = not approved
['tfidf_w2v_essay'] <= -0.004
samples = 25250
value = [17285.863, 11793.291]
class = approved
['tfidf_w2v_essay'] <= 0.087
samples = 2661
value = [425.955, 1491.912]
class = not approved
['tfidf_w2v_essay'] <= -0.018
samples = 6170
value = [2123.173, 3256.634]
class = not approved
essay_word_count <= 0.005
samples = 9610
value = [8350.706, 4172.285]
class = approved
['tfidf_w2v_essay'] <= -0.065
samples = 15640
['tfidf_w2v_essay'] <= 0.087
samples = 1411
value = [307.084, 776.595]
class = not approved
['tfidf_w2v_title'] <= -0.355
samples = 1250
value = [118.871, 715.316]
class = not approved
['tfidf_w2v_essay'] <= -0.18
samples = 2079
value = [445.767, 1145.449]
class = not approved
['tfidf_w2v_essay'] <= -0.069
samples = 4091
value = [1677.406, 2111.185]
class = not approved
price <= 0.002
samples = 5845
value = [5847.806, 2400.493]
class = approved
essay_word_count <= 0.007
samples = 3765
value = [2502.9, 1771.792]
class = approved
['tfidf_w2v_essay'] <= -0.094
samples = 8093
samples = 2431
value = [571.242, 1330.465]
class = not approved
samples = 2200
value = [425.955, 1220.28]
class = not approved
samples = 641
value = [75.946, 364.14]
class = not approved
samples = 770
value = [231.139, 412.456]
class = not approved
samples = 119
value = [39.624, 63.047]
class = not approved
samples = 1131
value = [79.248, 652.269]
class = not approved
samples = 550
value = [204.723, 287.541]
class = not approved
samples = 1529
value = [241.045, 857.908]
class = not approved
samples = 913
value = [201.421, 502.018]
class = not approved
samples = 3178
value = [1475.985, 1609.167]
class = not approved
samples = 2506
value = [1984.49, 1122.469]
class = approved
samples = 3339
value = [3863.316, 1278.024]
class = approved
samples = 3267
value = [2317.99, 1511.356]
class = approved
samples = 498
value = [184.911, 260.436]
class = not approved
samples = 1390
value = [931.158, 652.859]
class = approved
1500
2000
2500



6.3 Confusion Matrix

6.3.1 Train data

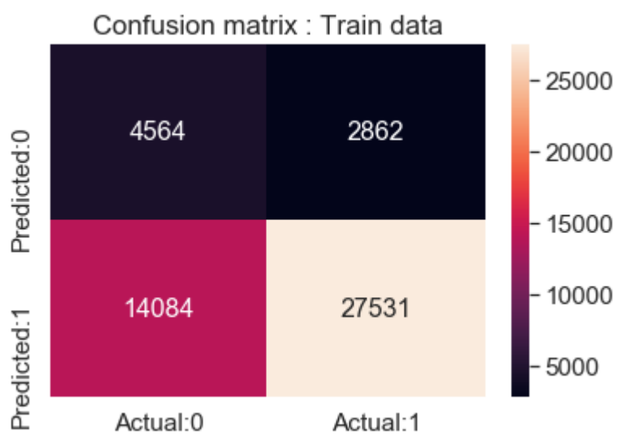
In [81]:

```
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) 0.406595697113 for threshold 0.515
Train confusion matrix
[[ 4564  2862]
 [14084 27531]]
Test confusion matrix
[[ 3193  2266]
 [10865 19728]]
```

In [82]:

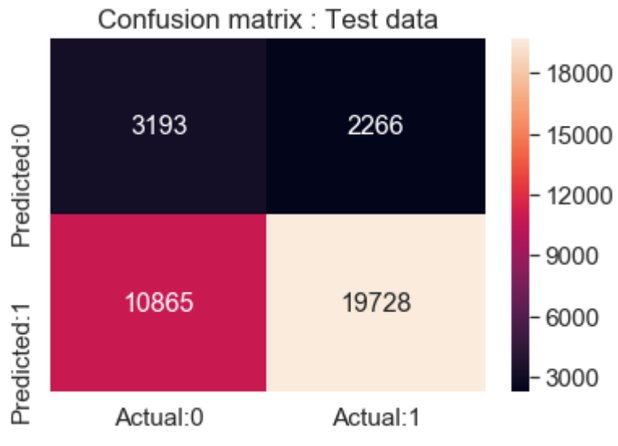
```
### PLOT the matrix for Train
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(train_predicts, best_t)), range(2), range(2))
# 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()
```



6.3.2 Test data

In [83]:

```
### 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()
```



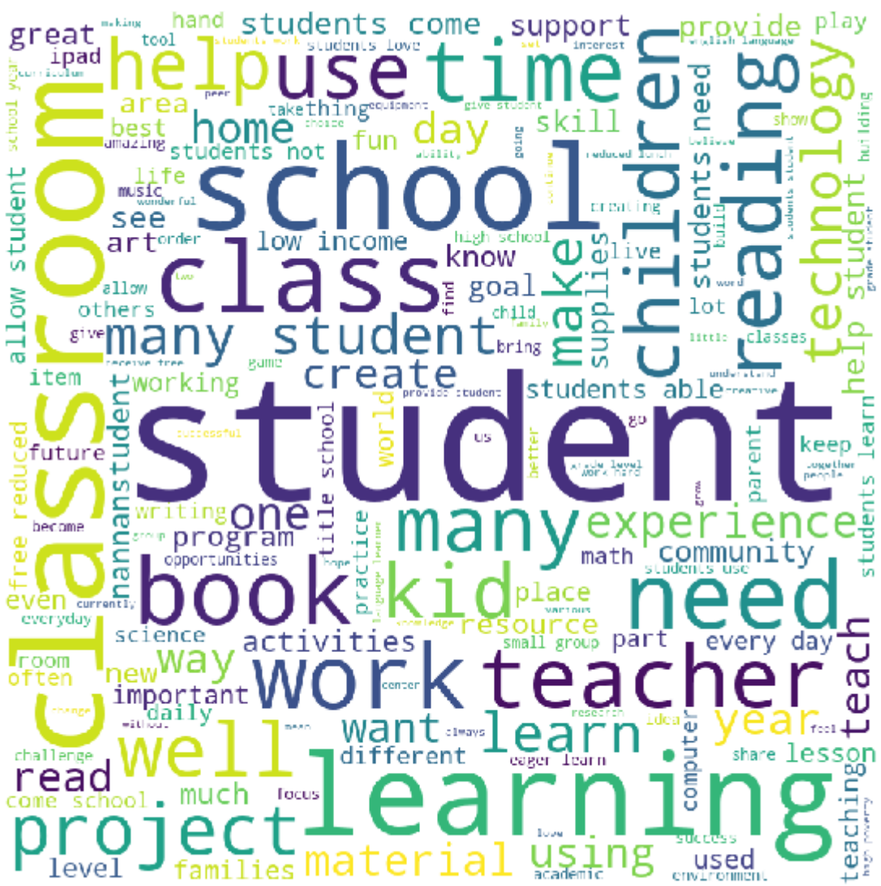
Observations :

1.We can observe from train and test we are getting majority True positives
2.Least number of data falls in False negative,which refers as least number
of projects were incorrectly predicted as not approved in both Test and Train.
3.For a model to perform well we need High True Positive Rate and Low False
Positive Rate.From the above our train data has True Positive Rate as 90.5%
and False Positive Rate as 75.5%
4.In our test data : True Positive Rate as 89.6% and False Positive Rate as 77.2%.

Word Cloud for false possitives

In [84]:

```
# plot the WordCloud image  using function word_cloud
wordcloud,fpi_i=word_cloud(y_test,test_predicts,best_t)
plt.figure(figsize = (8, 8), facecolor ="none")
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observations :

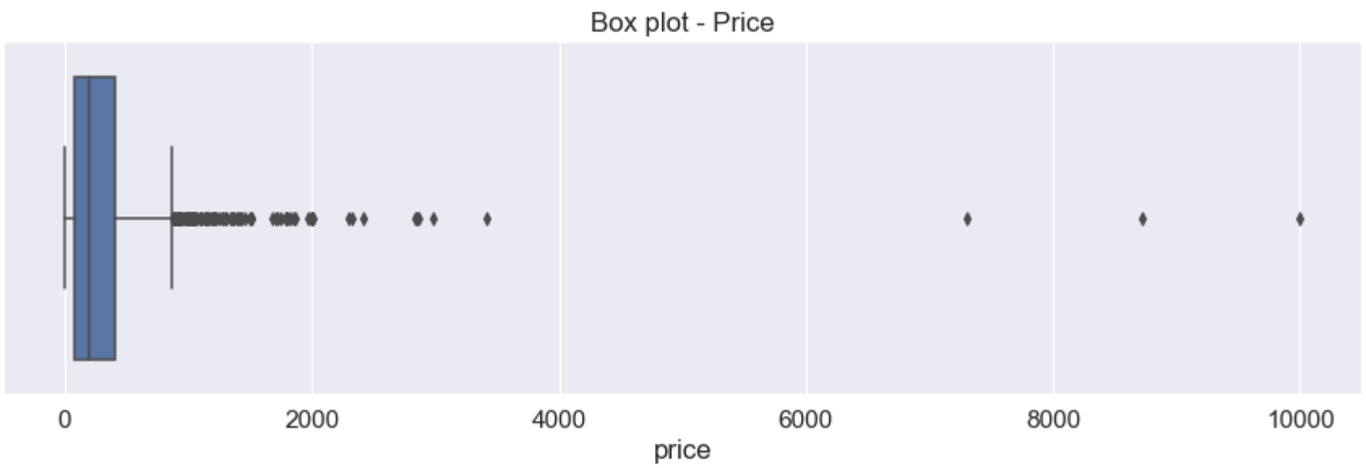
1. The data points which were false possitives were incorrectly
classified as positive due to words like learning ,classroom,student etc.

BOX plot for price column

In [85]:
```
test_price=X_test.values
fp_df=pd.concat([pd.DataFrame([test_price[i]],columns=X_test.columns) for i in fp_i ],ignore_index=True)

```

In [86]:
```
plt.figure(figsize=(15,4))
sns.boxplot(x="price",data=fp_df)
plt.title('Box plot - Price')
```

Out[86]: Text(0.5, 1.0, 'Box plot - Price')



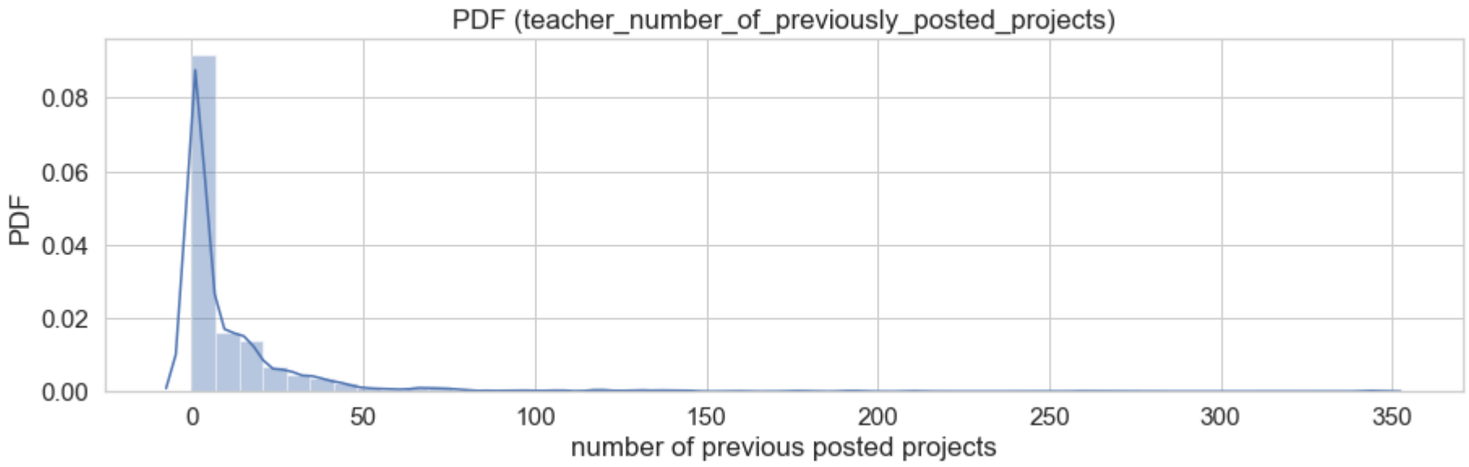
Observations :

1. We see that most false positive points have very less price.
2. There are few projects which had higher than 2000 as price that
was not approved but our model predicts as approved,these points are outlier points .

PDF for number_previous_posted_project for false positive points

In [87]:
```
plt.figure(figsize=(15,4))
sns.set_style('whitegrid')
sns.distplot(fp_df['teacher_number_of_previously_posted_projects'])
plt.xlabel('number of previous posted projects')
plt.ylabel('PDF')
plt.title('PDF (teacher_number_of_previously_posted_projects)')
```

Out[87]: Text(0.5, 1.0, 'PDF (teacher_number_of_previously_posted_projects)')



Observations :

1.Most false positive points had number of previously posted projects to be less than 25.

7. Task 2 : Model performance after removing zero importance features

1.With additional features where Sentiment is Analysed and classified as positive,neutral,negetive and compound

In [88]:
```
### making our datasets ready
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
X_tr = hstack((train_categories, train_subcategories,sklstate_train,teacher_prefix_train,
            proj_grade_train,tfidf_essay_train,tfidf_title_train,sentiment_pos_train_norm,
            sentiment_neg_train_norm,sentiment_neu_train_norm,sentiment_compound_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((test_categories, test_subcategories,sklstate_test,teacher_prefix_test,
            proj_grade_test,tfidf_essay_test,tfidf_title_test,sentiment_pos_test_norm,
            sentiment_neg_test_norm,sentiment_neu_test_norm,sentiment_compound_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((cv_categories, cv_subcategories,sklstate_cv,teacher_prefix_cv,
            proj_grade_cv,tfidf_essay_cv,tfidf_title_cv,sentiment_pos_cv_norm,
            sentiment_neg_cv_norm,sentiment_neu_cv_norm,sentiment_compound_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)
```

```
(49041, 7116)
(36052, 7116)
(24155, 7116)
```

7.1 Fit Decision tree model with max depth None

In [89]:
```
# call the model function with max depth as none
tree=DecisionTreeClassifier(class_weight='balanced',max_depth=None,
                            random_state=4)
```

In [90]:
```
## create a new training set later discarding zero importance features

feature_names= [cat_vectorize.get_feature_names()+subcat_vectorize.get_feature_names()+
                sklstate_vectorize.get_feature_names()+
                teacher_prefix_vectorize.get_feature_names()+
               proj_grade_vectorize.get_feature_names()+tfidf_essay.get_feature_names()+
                tfidf_title.get_feature_names()+
               ['sentiment_positive']+['sentiment_negetive']+['sentiment_neutral']+
                ['sentiment_compound']+['price']+['quantity']+['prev_projects_count']+
                ['title_word_count']+['essay_word_count']]

dataset_tr=pd.DataFrame(data=X_tr.todense(),columns=feature_names[0])
dataset_te=pd.DataFrame(data=X_te.todense(),columns=feature_names[0])
dataset_cv=pd.DataFrame(data=X_cr.todense(),columns=feature_names[0])
# fit the tree in the datset
tree.fit(X_tr,y_train)
```

Out[90]:
```
DecisionTreeClassifier(class_weight='balanced', criterion='gini',
            max_depth=None, max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=4,
            splitter='best')
```

In [91]:
```
## store feature importance in a dictionary
feat_importance=dict(zip(dataset_tr.columns, tree.feature_importances_))
```

In [92]:
```
#removing zero importance feature from the dataset
cols_=[key for key,value in feat_importance.items() if value==0.0]
dataset_tr.drop(columns=cols_,axis=1,inplace=True)
dataset_te.drop(columns=cols_,axis=1,inplace=True)
dataset_cv.drop(columns=cols_,axis=1,inplace=True)
```

7.2 Training logistic Regression model with Non-zero importance feature

7.2.1 Find best hyper params using Grid Searchcv with cv=5

In [93]:
```
X_tr=sparse.csr_matrix(dataset_tr.values)
X_te=sparse.csr_matrix(dataset_te.values)
X_cr=sparse.csr_matrix(dataset_cv.values)
print(X_tr.shape)
print(X_te.shape)
print(X_cr.shape)
```

```
(49041, 1131)
(36052, 1131)
(24155, 1131)
```

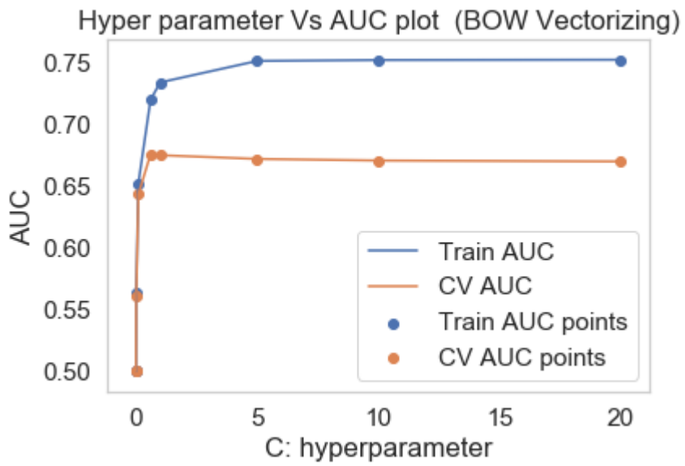
In [94]:
```
parameters={"C" : [0.0001,0.001,0.01,0.1,0.6,1,5,10,20] }
LR=LogisticRegression(class_weight='balanced',penalty='l1',random_state=4)
clf = GridSearchCV(LR, parameters, cv=3, scoring='roc_auc',verbose=1,n_jobs=3)
clf.fit(X_tr, y_train)
results = pd.DataFrame.from_dict(clf.cv_results_)
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits

[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Done  27 out of  27 | elapsed:   34.8s finished
```

In [95]:
```
results = results.sort_values(['param_C'])
RS_alphas=results['param_C']
train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
```

In [96]:

```
plt.plot(RS_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(RS_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(RS_alphas, train_auc, label='Train AUC points')
plt.scatter(RS_alphas, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot  (BOW Vectorizing)")
plt.grid()
plt.show()

results.head()
```



Out[96]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	params	split0_test_score	split1_test_score	split2_test_score	mean_test_score	std_test_score	rank_test_score	split0_train_score	split1_train_
0	0.200462	0.005086	0.012967	0.002154	0.0001	{'C': 0.0001}	0.500000	0.500000	0.500000	0.500000	0.000000	8	0.500000	0.5
1	0.168216	0.009543	0.009308	0.001244	0.001	{'C': 0.001}	0.500000	0.500000	0.500000	0.500000	0.000000	8	0.500000	0.5
2	0.367023	0.019960	0.013293	0.002622	0.01	{'C': 0.01}	0.565339	0.550533	0.568503	0.561458	0.007832	7	0.568408	0.5
3	0.517615	0.060242	0.017624	0.000948	0.1	{'C': 0.1}	0.639154	0.647760	0.643388	0.643434	0.003513	6	0.654890	0.6
4	2.743997	0.034357	0.015293	0.002618	0.6	{'C': 0.6}	0.673963	0.679636	0.674048	0.675882	0.002654	1	0.718692	0.7

Observations

1.Using Grid Search technique we find that as C values increase from 0.0001, AUC maintains a steadiness beyond 1.
2.In CV data and train data we see that maximum AUC reached is at C =0.6 beyond that AUC remains almost constant.

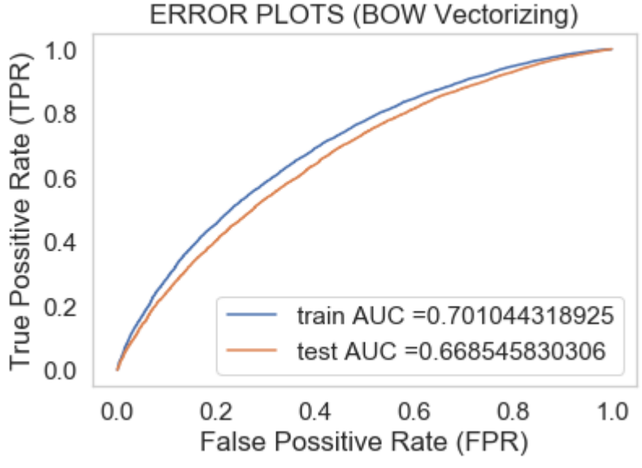
7.2.2 Train the model using the best Hyperparameter value

In [97]:
```
### https://forums.fast.ai/t/hyperparameter-random-search-interpretation/8591 ---to get the best hyper parameter as a reuslt of Random search
best_C = clf.best_params_
print('Best Alpha as a result of Grid Search',best_C)
```

```
Best Alpha as a result of Grid Search {'C': 0.6}
```

In [98]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#sklearn.metrics.roc_curve

LR=LogisticRegression(penalty="l1",C=best_C['C'],n_jobs=3)
LR.fit(X_tr, y_train)

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

y_train_pred = LR.predict_proba(X_tr)[:,1]
y_test_pred = LR.predict_proba(X_te)[:,1]

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Possitive Rate (FPR)")
plt.ylabel("True Possitive Rate (TPR)")
plt.title("ERROR PLOTS (BOW Vectorizing)")
#store results
results_dict.update({'TFIDF_without_zero_weight_feat' : {'trainauc':str(auc(train_fpr, train_tpr)),
                        'testauc': str(auc(test_fpr, test_tpr)),
                        'C' : best_C['C']}})
plt.grid()
plt.show()
```



Observations :

1. Test AUC found to be 0.66 and Train AUC as 0.71 after training model using best hyperparameter 0.6 and vectorizing text data using TFIDF after discarding zero importance words.

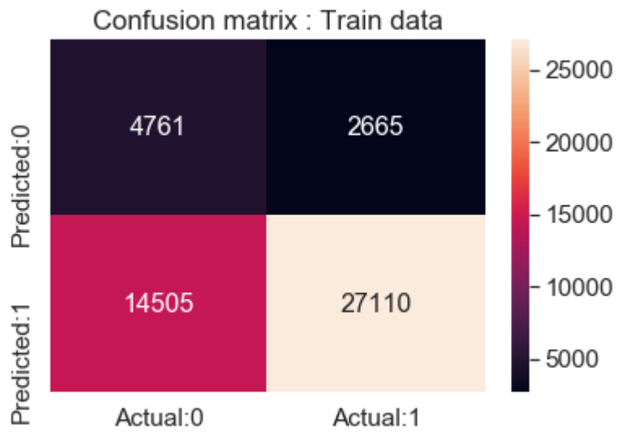
7.3 Confusion Matrix

7.3.1 Train data

In [99]:
```
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) 0.41765997216 for threshold 0.844
Train confusion matrix
[[ 7367    59]
 [39173  2442]]
Test confusion matrix
[[ 5389    70]
 [29178  1415]]
```

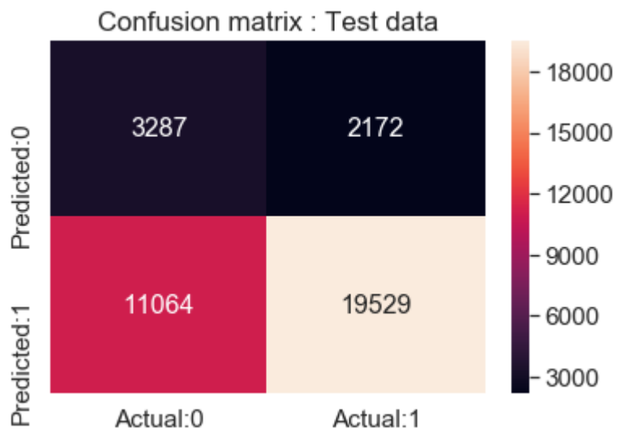
In [100]:
```
### PLOT the matrix for Train
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
# source : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), range(2), 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()
```



7.3.2 Test data

In [101]:

```
### 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(y_test_pred, 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()
```



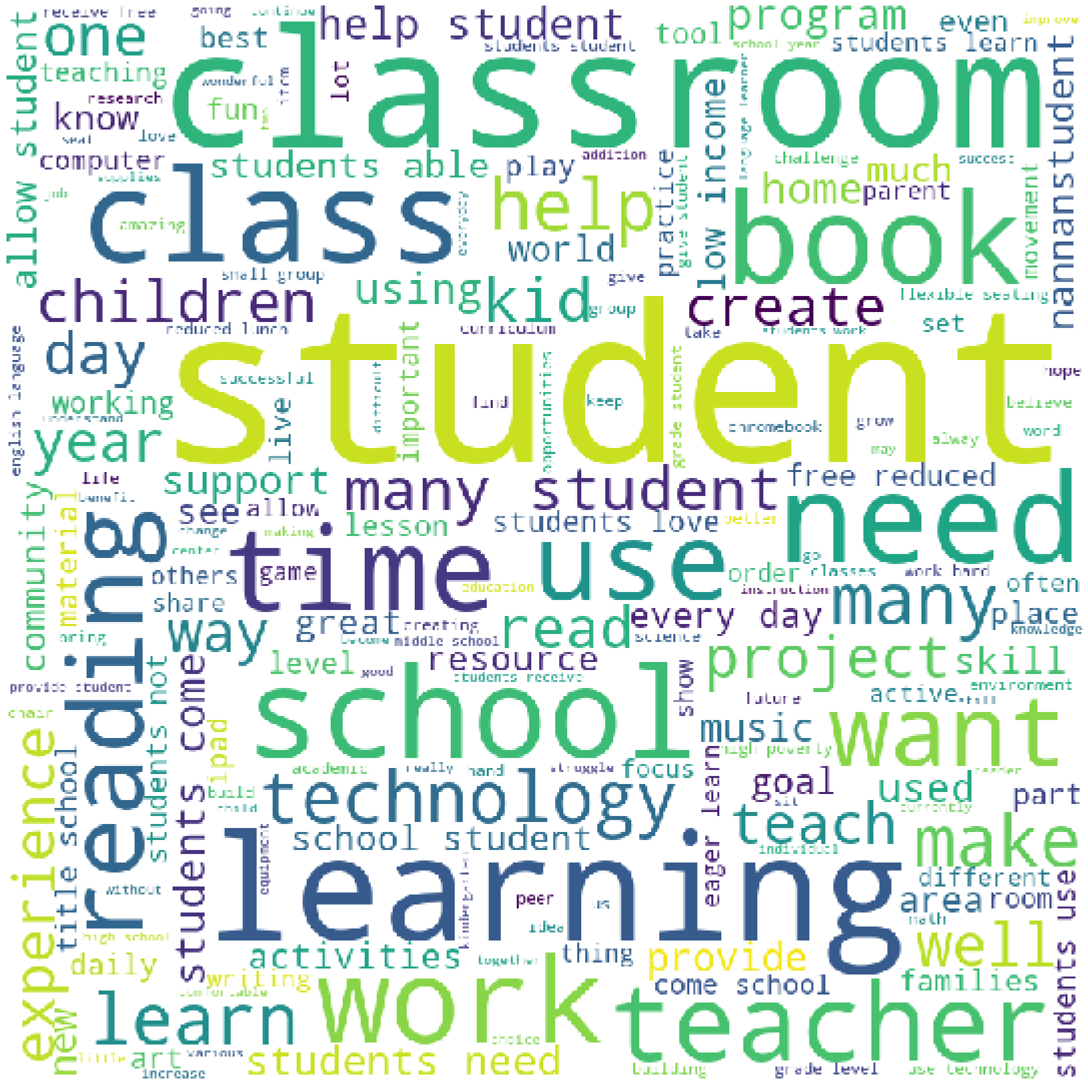
Observations :

1.We can observe from train and test we are getting majority True positives
2.Least number of data falls in False negative,which refers as least
number of projects were incorrectly predicted as not approved in both Test and Train.
3.For a model to perform well we need High True Positive Rate and
Low False Positive Rate.From the above our train data has True Positive Rate as
91.05% and False Positive Rate as 75.2%
4.In our test data : True Positive Rate as 89.9% and False Positive Rate as 77%.

Word Cloud for false possitives

In [102]:

```
# plot the WordCloud image  using function word_cloud
wordcloud,fp_i=word_cloud(y_test,y_test_pred,best_t)
plt.figure(figsize = (8, 8), facecolor ="none")
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observations :

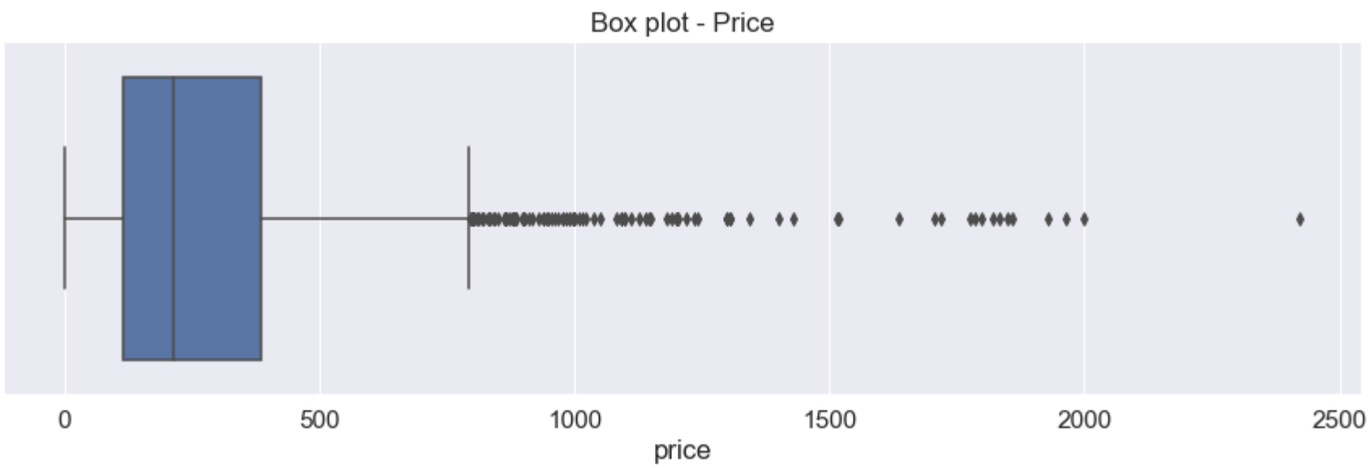
1. The data points which were false possitives were incorrectly classified as positive due to words like learning ,classroom,student etc.

BOX plot for price column

In [103]:
```
test_price=X_test.values
fp_df=pd.concat([pd.DataFrame([test_price[i]],columns=X_test.columns)
                 for i in fp_i ],ignore_index=True)

```

In [104]:
```
plt.figure(figsize=(15,4))
sns.boxplot(x="price",data=fp_df)
plt.title('Box plot - Price')
```

Out[104]: Text(0.5, 1.0, 'Box plot - Price')



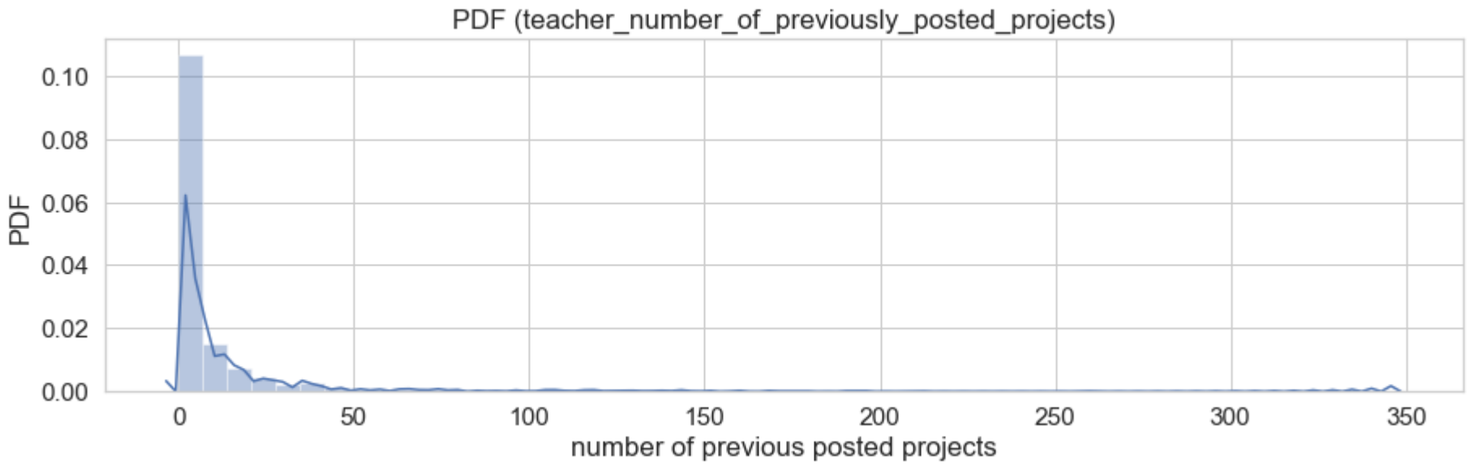
Observations :

1. We see that most false positive points have very less price between 0 - 500.
2. There are few projects which had higher than 500 as price that
was not approved but our model predicts as approved,these points are outlier points .

PDF for number_previous_posted_project for false positive points

In [105]:
```
plt.figure(figsize=(15,4))
sns.set_style('whitegrid')
sns.distplot(fp_df['teacher_number_of_previously_posted_projects'])
plt.xlabel('number of previous posted projects')
plt.ylabel('PDF')
plt.title('PDF (teacher_number_of_previously_posted_projects)')
```

Out[105]: Text(0.5, 1.0, 'PDF (teacher_number_of_previously_posted_projects)')



Observations :

1.Most false positive points had number of previously posted projects to be less than 25

In [106]:

```
##http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()
x.field_names = [ "Vectorizer", "Model" , "hyperparameter",'Train AUC ','Test AUC']

x.add_row(["TFIDF",'DecisionTree',{'max_depth':results_dict['TFIDF']['max_depth'],
                                   'min_sample':results_dict['TFIDF']['min_sample_split']},
                                    round(float(results_dict['TFIDF']['trainauc']),2),
                                     round(float(results_dict['TFIDF']['testauc']),2) ])
x.add_row(["Weighted TFIDF",'DecisionTree',{'max_depth':results_dict['TFIDF W2V']['max_depth'],
                                   'min_sample':results_dict['TFIDF W2V']['min_sample_split']},
                                    round(float(results_dict['TFIDF W2V']['trainauc']),2),
                                    round(float(results_dict['TFIDF W2V']['testauc']),2)])
x.add_row(["TFIDF without zero weight features",'Logistic Regression',
          {'C':results_dict['TFIDF_without_zero_weight_feat']['C']},
            round(float(results_dict['TFIDF_without_zero_weight_feat']['trainauc']),2),
            round(float(results_dict['TFIDF_without_zero_weight_feat']['testauc']),2)])

print(x)
```

```
+-----------------------------------+---------------------+-----------------------------------+------------+----------+
|            Vectorizer             |        Model        |            hyperparameter         | Train AUC  | Test AUC |
+-----------------------------------+---------------------+-----------------------------------+------------+----------+
|              TFIDF                |     DecisionTree    | {'max_depth': 10, 'min_sample': 500} |    0.74    |   0.69   |
|          Weighted TFIDF           |     DecisionTree    | {'max_depth': 5, 'min_sample': 500}  |    0.69    |   0.65   |
| TFIDF without zero weight features | Logistic Regression |             {'C': 0.6}            |    0.7     |   0.67   |
+-----------------------------------+---------------------+-----------------------------------+------------+----------+
```

In [107]:

```
#REFERENCES of codes:
#reference from course material : reference_EDA.ipynb  and other reference notebooks
#reference :https://stackoverflow.com/questions/35286540/display-an-image-with-python
#reference :https://medium.com/@rnbrown/creating-and-visualizing-decision-trees-with-python-f8e8fa39417
#reference :https://stackoverflow.com/questions/1494492/graphviz-how-to-go-from-dot-to-a-graph
#source  : https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix etc.
```