

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
<code>project_id</code>		A unique identifier for the proposed project. Example: 123456789
<code>project_title</code>		Title of the project. Example: Art Will Make You
<code>project_grade_category</code>		Grade level of students for which the project is targeted. One of the enumerated categories: • Kindergarten • Grades 1-2 • Grades 3-5 • Grades 6-8 • Grades 9-12
<code>project_subject_categories</code>		One or more (comma-separated) subject categories for the project from the following enumerated list: • Applied Learning • Care & Community • Health & Physical Education • History & Social Studies • Literacy & Language • Math & Science • Music & Arts • Special Education
<code>project_subject_subcategories</code>		One or more (comma-separated) subject subcategories for the project from the following enumerated list: • Music & Arts • Literacy & Language, Math & Science
<code>school_state</code>		State where school is located (Two-letter U.S. postal abbreviations) (https://en.wikipedia.org/wiki/List of U.S. state abbreviations#Postal abbreviations)
<code>project_resource_summary</code>		An explanation of the resources needed for the project. Example: My students need hands on literacy materials to enhance their sensory
<code>project_essay_1</code>		First application essay
<code>project_essay_2</code>		Second application essay
<code>project_essay_3</code>		Third application essay
<code>project_essay_4</code>		Fourth application essay
<code>project_submitted_datetime</code>		Datetime when project application was submitted. Example: 2018-09-12T12:43:21
<code>teacher_id</code>		A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4f1

Feature	Description
<code>teacher_prefix</code>	Teacher's title. One of the following enumerated values: <ul style="list-style-type: none"> 1: Assistant Professor 2: Associate Professor 3: Full Professor 4: Lecturer 5: Other
<code>teacher_number_of_previously_posted_projects</code>	Number of project applications previously submitted by the same teacher

* 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	Description 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

Notes on the Essay Data

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

- __project_essay_1: __ "Introduce us to your classroom"
- __project_essay_2: __ "Tell us more about your students"
- __project_essay_3: __ "Describe how your students will use the materials you're requesting"
- project_essay_3: __ "Close by sharing why your project will make a difference"

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

- __project_essay_1: __ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2: __ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

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

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv', nrows=50000)
resource_data = pd.read_csv('resources.csv')
```

In [3]:

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

Number of data points in train data (50000, 17)

The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix'
 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']

In [4]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

Number of data points in train data (1541272, 4)

['id' 'description' 'quantity' 'price']

Out[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

1.2 preprocessing of project_subject_categories

In [5]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in categories:
    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 category 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 replacing all the ' ' (space) with '' (empty) ex: "Math & Science"=> "Math&Science"
            temp+=j.strip()+" " # " abc ".strip() will return "abc", remove the trailing spaces
    temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of project_subject_subcategories

In [6]:

```

sub_categories = 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_categories:
    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 category 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 replacing all the ' ' (space) with '' (empty) ex: "Math & Science" => "Math&Science"
            temp += j.strip() + " #" "abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&', '_')
    sub_cat_list.append(temp.strip())

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

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

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

```

1.3 Text preprocessing

In [7]:

```

# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) + \
    project_data["project_essay_2"].map(str) + \
    project_data["project_essay_3"].map(str) + \
    project_data["project_essay_4"].map(str)

```

In [8]:

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

Out[8]:

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

In [9]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```


In [10]:

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

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 countries represented with the families within our school. Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, and respect.\r\n\r\nThe limits of your language are the limits of your world.\r\n\r\n-Ludwig Wittgenstein Our English learner's have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English along side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one at home is able to assist. All families with students within the Level 1 proficiency status, will be offered to be a part of this program. These educational videos will be specially chosen by the English Learner Teacher and will be sent home regularly to watch. The videos are to help the child develop early reading skills.\r\n\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\n\r\nnannan

=====

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\n\r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity. My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the students who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still. nannan

=====

How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed r

aces in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more. With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

=====
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and in turn fine motor skills. \r\n\r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

=====

In [11]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can't", "can not", phrase)

    # general
    phrase = re.sub(r"n't", " not", phrase)
    phrase = re.sub(r"'re", " are", phrase)
    phrase = re.sub(r"'s", " is", phrase)
    phrase = re.sub(r"'d", " would", phrase)
    phrase = re.sub(r"'ll", " will", phrase)
    phrase = re.sub(r"'t", " not", phrase)
    phrase = re.sub(r"'ve", " have", phrase)
    phrase = re.sub(r"'m", " am", phrase)
    return phrase
```

In [12]:

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

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. They want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and in turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves. nannan

=====

In [13]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\r', ' ')
sent = sent.replace('\n', ' ')
sent = sent.replace('\t', ' ')
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. The materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. They want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and in turn fine motor skills. They also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves. nannan

In [14]:

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

My kindergarten students have varied disabilities ranging from speech and language delays cognitive delays gross fine motor delays to autism They are eager beavers and always strive to work their hardest working past their limitations The materials we have are the ones I seek out for my students I teach in a Title I school where most of the students receive free or reduced price lunch Despite their disabilities and limitations my students love coming to school and come eager to learn and explore Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting This is how my kids feel all the time They want to be able to move as they learn or so they say Wobble chairs are the answer and I love them because they develop their core which enhances gross motor and in turn fine motor skills They also want to learn through games my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Physical engagement is the key to our success The number toss and color and shape mats can make that happen My students will forget they are doing work and just have the fun a 6 year old deserves nannan

```
# 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'r
e", "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', 't
hey', 'them', 'their', \
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "th
at'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', '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', 'ov
er', 'under', 'again', 'further', \
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
y', '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", 'no
w', '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', 'migh
tn', "mightn't", 'mustn', \
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'w
asn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

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

```
100% |██████████████████████████████████████████████████████████|
      | 50000/50000 [00:52<00:00, 944.22it/s]
```

In [17]:

```
# after preprocessing
preprocessed_essays[20000]
```

Out[17]:

```
'kindergarten students varied disabilities ranging speech language delays
cognitive delays gross fine motor delays autism eager beavers always striv
e work hardest working past limitations materials ones seek students teach
title school students receive free reduced price lunch despite disabilitie
s limitations students love coming school come eager learn explore ever fe
lt like ants pants needed groove move meeting kids feel time want able mov
e learn say wobble chairs answer love develop core enhances gross motor tu
rn fine motor skills also want learn games kids not want sit worksheets wa
nt learn count jumping playing physical engagement key success number toss
color shape mats make happen students forget work fun 6 year old deserves
nannan'
```

In [18]:

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

1.4 Preprocessing of `project_title`

In [19]:

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

In [20]:

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

```
100%|████████████████████████████████████████████████████████████████████████████████|
██████| 50000/50000 [00:02<00:00, 17612.70it/s]
```

In [21]:

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

In [22]:

```
#Preprocessing project_grade_category

#reference link: https://stackoverflow.com/questions/28986489/python-pandas-how-to-replace-a-characters-in-a-column-of-a-dataframe

project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-', '_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ', '_')
```

1.5 Preparing data for models

In [23]:

```
project_data.columns
```

Out[23]:

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

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data (optional)
- quantity : numerical (optional)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

1.5.1 Vectorizing Categorical data

- <https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/> (<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/>)

In [24]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encoding ",categories_one_hot.shape)
```

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

In [25]:

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutrition
Education', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'A
ppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Lit
eracy']
Shape of matrix after one hot encoding (50000, 30)
```

In [26]:

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

In [27]:

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

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

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

In [28]:

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

teacher_prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].values.ast
ype('U')) #here also i did same as above
print('Shape of matrix of one hot encoding', teacher_prefix_one_hot.shape)

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

In [29]:

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

project_grade_category_one_hot = vectorizer.fit_transform(project_data['project_grade_c
ategory'].values.astype('U'))
print('Shape of matrix of one hot encoding', project_grade_category_one_hot.shape)

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

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [30]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).  
vectorizer = CountVectorizer(min_df=10)  
text_bow = vectorizer.fit_transform(preprocessed_essays)  
print("Shape of matrix after one hot encoding ",text_bow.shape)
```

Shape of matrix after one hot encoding (50000, 12101)

In [31]:

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

In [32]:

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

Shape of matrix after one hot encoding (50000, 2039)

1.5.2.2 TFIDF vectorizer

In [33]:

```
from sklearn.feature_extraction.text import TfidfVectorizer  
vectorizer = TfidfVectorizer(min_df=10)  
text_tfidf = vectorizer.fit_transform(preprocessed_essays)  
print("Shape of matrix after one hot encoding ",text_tfidf.shape)
```

Shape of matrix after one hot encoding (50000, 12101)

In [34]:

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

Shape of matrix after one hot encoding (50000, 2039)

1.5.2.3 Using Pretrained Models: Avg W2V

In [35]:

```

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

# =====
Output:

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

# =====

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

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

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

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

# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/

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

...

```

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Mode
l")\n    f = open(gloveFile,\r', encoding="utf8")\n    model = {}\n    f
or line in tqdm(f):\n        splitLine = line.split()\n        word = spli
tline[0]\n        embedding = np.array([float(val) for val in splitLine
[1:]])\n        model[word] = embedding\n        print ("Done.",len(model)," w
ords loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.
txt\')\n\n# =====\n\nOutput:\n    \nLoading Glove Mod
el\n1917495it [06:32, 4879.69it/s]\nDone. 1917495 words loaded!\n\n# ====
=====
\n\nwords = []\nfor i in preproced_texts:\n    wor
ds.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend
(i.split(\' \'))\n\nprint("all the words in the coupus", len(words))\nwords
= set(words)\n\nprint("the unique words in the coupus", len(words))\n\ninter
_words = set(model.keys()).intersection(words)\n\nprint("The number of words
that are present in both glove vectors and our coupus", len(inter_wo
rds), "(" ,np.round(len(inter_words)/len(words)*100,3), "%")\n\nwords_courpu
s = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in word
s_glove:\n        words_courpus[i] = model[i]\n\nprint("word 2 vec length",
len(words_courpus))\n\n\n# stronging variables into pickle files python: h
ttp://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-
python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n
pickle.dump(words courpus, f)\n\n\n'
```

```
# stringing 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
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove words = set(model.keys())
```

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

50000
300

In [43]:

```
# check this one: https://www.youtube.com/watch?v=0H0q0cLn3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScaler.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329. ...
# 399. 287.73 5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

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

Mean : 299.33367619999996, Standard deviation : 378.20927190421384

In [44]:

```
price_standardized
```

Out[44]:

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

1.5.4 Merging all the above features

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

In [45]:

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

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


In [46]:

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

Out[46]:

(50000, 12141)

In [47]:

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

Computing Sentiment Scores

In [48]:

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

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest
students with the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multi
ple intelligences i use a wide range\
of techniques to help all my students succeed students in my class come from a variety
of different backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school
is a caring community of successful \
learners which can be seen through collaborative student project based learning in and
out of the classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities
to practice a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspec
t of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love
to role play in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with
real food i will take their idea \
and create common core cooking lessons where we learn important math and writing concep
ts while cooking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that wen
t into making the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this p
roject would expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make hom
emade applesauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create o
ur own cookbooks to be printed and \
shared with families students will gain math and literature skills as well as a life lo
ng enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

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

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

neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,

Assignment 8: DT

1. Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets

- **Set 1:** categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
- **Set 2:** categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
- **Set 3:** categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
- **Set 4:** categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)




2. Hyper paramter tuning (best `depth` in range [1, 5, 10, 50, 100, 500, 100], 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 or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. Graphviz

- Visualize your decision tree with Graphviz. It helps you to understand how a decision is being made, given a new vector.
- Since feature names are not obtained from word2vec related models, visualize only BOW & TFIDF decision trees using Graphviz
- Make sure to print the words in each node of the decision tree instead of printing its index.
- Just for visualization purpose, limit max_depth to 2 or 3 and either embed the generated images of graphviz in your notebook, or directly upload them as .png files.

4. Representation of results

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure 
- 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-tp-r-fpr-fnr-tnr-1/) with predicted and original labels of test data points 
- Once after you plot the confusion matrix with the test data, get all the `false positive data points`
 - Plot the WordCloud [WordCloud](https://www.geeksforgeeks.org/generating-word-cloud-python/) (https://www.geeksforgeeks.org/generating-word-cloud-python/)
 - 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`

5. [Task-2]

- Select 5k best features from features of **Set 2** using `feature_importances_` (<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>), discard all the 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

6. Conclusion

- You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library [link \(http://zetcode.com/python/prettytable/\)](http://zetcode.com/python/prettytable/)



2. Decision Tree

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

In [49]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
# a. Title, that describes your plot, this will be very helpful to the reader
# b. Legends if needed
# c. X-axis Label
# d. Y-axis Label
```

In [50]:

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

In [51]:

```
from sklearn.model_selection import train_test_split

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

```
(33500, 19)
(16500, 19)
(33500,)
(16500,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [52]:

```
# please write all the code with proper documentation, and proper titles for each subsection  
# go through documentations and blogs before you start coding  
# first figure out what to do, and then think about how to do.  
# reading and understanding error messages will be very much helpfull in debugging your code  
# make sure you featurize train and test data separatly  
  
# when you plot any graph make sure you use  
    # a. Title, that describes your plot, this will be very helpful to the reader  
    # b. Legends if needed  
    # c. X-axis label  
    # d. Y-axis label
```

In [53]:

```
#Here I'm not encoding numerical features because in decision tress, there's no need to normalize/standardize data.  
#Reshaping numerical features and printing their shapes
```

In [54]:

```
#Price  
  
x_train_price = x_train['price'].values.reshape(1, -1)  
x_test_price = x_test['price'].values.reshape(1, -1)  
  
print(x_train_price.shape)  
print(x_test_price.shape)
```

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

In [55]:

```
#teacher_number_of_previously_posted_projects  
  
x_train_previous_projects = x_train['teacher_number_of_previously_posted_projects'].values.reshape(1, -1)  
x_test_previous_projects = x_test['teacher_number_of_previously_posted_projects'].values.reshape(1, -1)  
  
print(x_train_previous_projects.shape)  
print(x_test_previous_projects.shape)
```

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

In [56]:

```
#quantity

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

print(x_train_quantity.shape)
print(x_test_quantity.shape)
```

(1, 33500)

(1, 16500)

In [57]:

```
#Encoding Categorical Features
```

In [109]:

```
#school_state
state_vectorizer = CountVectorizer()
state_vectorizer.fit(x_train['school_state'].values)

x_train_state_one_hot = state_vectorizer.transform(x_train['school_state'].values)
x_test_state_one_hot = state_vectorizer.transform(x_test['school_state'].values)

print("After Vectorizations:")
print(x_train_state_one_hot.shape, y_train.shape)
print(x_test_state_one_hot.shape, y_test.shape)
print(state_vectorizer.get_feature_names())
```

After Vectorizations:

(33500, 51) (33500,)

(16500, 51) (16500,)

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi',
'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'm
o', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'o
k', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'w
i', 'wv', 'wy']
```

In [110]:

```
#teacher_prefix

teacher_vectorizer = CountVectorizer()
teacher_vectorizer.fit(x_train['teacher_prefix'].values.astype('U'))

x_train_teacher_one_hot = teacher_vectorizer.transform(x_train['teacher_prefix'].values
.astype('U'))
x_test_teacher_one_hot = teacher_vectorizer.transform(x_test['teacher_prefix'].values.a
stype('U'))

print("After Vectorizations:")
print(x_train_teacher_one_hot.shape, y_train.shape)
print(x_test_teacher_one_hot.shape, y_test.shape)
```

After Vectorizations:

(33500, 5) (33500,)

(16500, 5) (16500,)

In [111]:

```
#Project_grade_category

grade_vectorizer = CountVectorizer()
grade_vectorizer.fit(x_train['project_grade_category'].values)

x_train_grade_one_hot = grade_vectorizer.transform(x_train['project_grade_category'].values)
x_test_grade_one_hot = grade_vectorizer.transform(x_test['project_grade_category'].values)

print("After Vectorizations:")
print(x_train_grade_one_hot.shape, y_train.shape)
print(x_test_grade_one_hot.shape, y_test.shape)
```

After Vectorizations:
(33500, 4) (33500,)
(16500, 4) (16500,)

In [112]:

```
#project_subject_categories

categories_vectorizer = CountVectorizer()
categories_vectorizer.fit(x_train['clean_categories'].values)

x_train_categories_one_hot = categories_vectorizer.transform(x_train['clean_categories'].values)
x_test_categories_one_hot = categories_vectorizer.transform(x_test['clean_categories'].values)

print("After Vectorizations:")
print(x_train_categories_one_hot.shape, y_train.shape)
print(x_test_categories_one_hot.shape, y_test.shape)
```

After Vectorizations:
(33500, 9) (33500,)
(16500, 9) (16500,)

In [113]:

```
#project_subject_subcategories

subcategories_vectorizer = CountVectorizer()
subcategories_vectorizer.fit(x_train['clean_subcategories'].values)

x_train_subcategories_one_hot = subcategories_vectorizer.transform(x_train['clean_subcategories'].values)
x_test_subcategories_one_hot = subcategories_vectorizer.transform(x_test['clean_subcategories'].values)

print("After Vectorizations:")
print(x_train_subcategories_one_hot.shape, y_train.shape)
print(x_test_subcategories_one_hot.shape, y_test.shape)
```

```
After Vectorizations:
(33500, 30) (33500,)
(16500, 30) (16500,)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [63]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

BOW: title, essay

In [114]:

```
#Project_title

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

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

print("After Vectorizations:")
print(x_train_title_bow.shape, y_train.shape)
print(x_test_title_bow.shape, y_test.shape)
```

After Vectorizations:
(33500, 769) (33500,)
(16500, 769) (16500,)

In [115]:

```
#Essay

essay_bow_vectorizer = CountVectorizer(min_df=10, ngram_range=(2, 2), max_features=5000
)
essay_bow_vectorizer.fit(x_train['essay'].values)

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

print("After Vectorizations:")
print(x_train_essay_bow.shape, y_train.shape)
print(x_test_essay_bow.shape, y_test.shape)
```

After Vectorizations:
(33500, 5000) (33500,)
(16500, 5000) (16500,)

TFIDF: ttitle, essay

In [145]:

```
#project_title

title_tfidf_vectorizer = TfidfVectorizer(min_df=10, ngram_range=(2, 2), max_features=50
00)
title_tfidf_vectorizer.fit(x_train['project_title'].values)

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

print("After Vectorizations:")
print(x_train_title_tfidf.shape, y_train.shape)
print(x_test_title_tfidf.shape, y_test.shape)
```

After Vectorizations:
(33500, 769) (33500,)
(16500, 769) (16500,)

In [116]:

```
#essay

from sklearn.feature_extraction.text import TfidfVectorizer
essay_tfidf_vectorizer = TfidfVectorizer(min_df=10, ngram_range=(2, 2), max_features=5000)
essay_tfidf_vectorizer.fit(x_train['essay'].values)

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

print("After Vectorizations:")
print(x_train_essay_tfidf.shape, y_train.shape)
print(x_test_essay_tfidf.shape, y_test.shape)
```

After Vectorizations:
(33500, 5000) (33500,)
(16500, 5000) (16500,)

AVG W2V: ttitle, essay

In [68]:

```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words = set(model.keys())
```

Project_title

In [69]:

```
x_train_title_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_train['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    x_train_title_avg_w2v.append(vector)

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

```
100% |██████████████████████████████████████████████████████████|
██████ 33500/33500 [00:00<00:00, 38280.78it/s]
```

33500
300

```
x_test_title_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_test['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    x_test_title_avg_w2v.append(vector)

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

Essay

```
# average Word2Vec
# compute average word2vec for each review.
x_train_essay_avg_w2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(x_train['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    x_train_essay_avg_w2v.append(vector)

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

35/100

2.4 Applying Decision Tree on different kind of featurization as mentioned in the instructions

Apply Decision Tree on different kind of featurization as mentioned in the instructions

For Every model that you work on make sure you do the step 2 and step 3 of instructions

In [79]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [80]:

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

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

2.4.1 Applying Decision Trees on BOW, SET 1

In []:

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

In [82]:

```
#merging features

from scipy.sparse import hstack
x_train_bow = hstack((x_train_price, x_train_previous_projects, x_train_quantity, x_train_state_one_hot, x_train_teacher_one_hot, \
                      x_train_grade_one_hot, x_train_categories_one_hot, x_train_subcategories_one_hot, x_train_essay_bow, \
                      x_train_title_bow)).tocsr()

x_test_bow = hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_test_state_one_hot, x_test_teacher_one_hot, \
                     x_test_grade_one_hot, x_test_categories_one_hot, x_test_subcategories_one_hot, x_test_essay_bow, \
                     x_test_title_bow)).tocsr()
```

In [83]:

```
'''
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

tuned_parameters = {'max_depth':[1, 5, 10], 'min_samples_split':[5, 10, 50]}
DTC = DecisionTreeClassifier(class_weight='balanced')
clf = GridSearchCV(DTC, tuned_parameters, cv=10, scoring='roc_auc')
clf.fit(x_train_bow, y_train)

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

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']
max_depth = results['param_max_depth']

plt.plot(max_depth, 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(max_depth, 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(max_depth, train_auc, Label='Train AUC points')
plt.scatter(max_depth, cv_auc, Label='CV AUC points')

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

results.head()
'''
```


Out[83]:

```

'\nfrom sklearn.model_selection import GridSearchCV\nfrom sklearn.tree imp
ort DecisionTreeClassifier\n\ntuned_parameters = {\n'max_depth':[1, 5, 1
0], \n'min_samples_split':[5, 10, 50]}\nDTC = DecisionTreeClassifier(class
_weight=\n'balanced\')\nclf = GridSearchCV(DTC, tuned_parameters, cv=10, sc
oring=\n'roc_auc\')\nclf.fit(x_train_bow, y_train)\n\nresults = pd.DataFram
e.from_dict(clf.cv_results_)\n\nmax_depth_results = results.sort_values([\n'p
aram_max_depth\'])\n\ntrain_auc= results[\n'mean_train_score\']\ntrain_auc_
std= results[\n'std_train_score\']\nncv_auc = results[\n'mean_test_score\']
\ncv_auc_std= results[\n'std_test_score\']\n\nmax_depth = results[\n'param_ma
x_depth\']\n\nplt.plot(max_depth, train_auc, label=\n'Train AUC\')\n\n# this
code is copied from here: https://stackoverflow.com/a/48803361/4084039\n#
plt.gca().fill_between(K, train_auc - train_auc_std,train_auc + train_auc_
std,alpha=0.2,color=\n'darkblue\')\n\n\nplt.plot(max_depth, cv_auc, label=
\n'CV AUC\')\n\n# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039\n# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.2,color=\n'darkorange\')\n\n\nplt.scatter(max_depth, train
_auc, label=\n'Train AUC points\')\n\nplt.scatter(max_depth, cv_auc, label=
\n'CV AUC points\')\n\n\nplt.legend()\nplt.xlabel("max_depth: hyperparamete
r")\nplt.ylabel("AUC")\nplt.title("Hyper parameter Vs AUC plot")\nplt.grid
()\nplt.show()\n\nresults.head()\n'

```

In [84]:

```

from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

tuned_parameters = {'max_depth':[1, 5, 10, 50, 100, 500], 'min_samples_split':[5, 10, 50, 100]}
DTC = DecisionTreeClassifier(class_weight='balanced')
clf = GridSearchCV(DTC, tuned_parameters, cv=10, scoring='roc_auc')
clf.fit(x_train_bow, y_train)

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

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

#For min_samples_split
train_auc_min_samples = results['mean_train_score']
train_auc_std_min_samples = results['std_train_score']
cv_auc_min_samples = results['mean_test_score']
cv_auc_std_min_samples = results['std_test_score']
min_samples_split = results['param_min_samples_split']

plt.plot(max_depth, train_auc_max_depth, label='Train AUC max_depth')
# 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(min_samples_split, train_auc_min_samples, label='Train AUC min_samples') #Plot
ting _min_samples-split curve

plt.plot(max_depth, cv_auc_max_depth, 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='da
rkorange')
plt.plot(min_samples_split, cv_auc_min_samples, label='CV AUC min_samples')

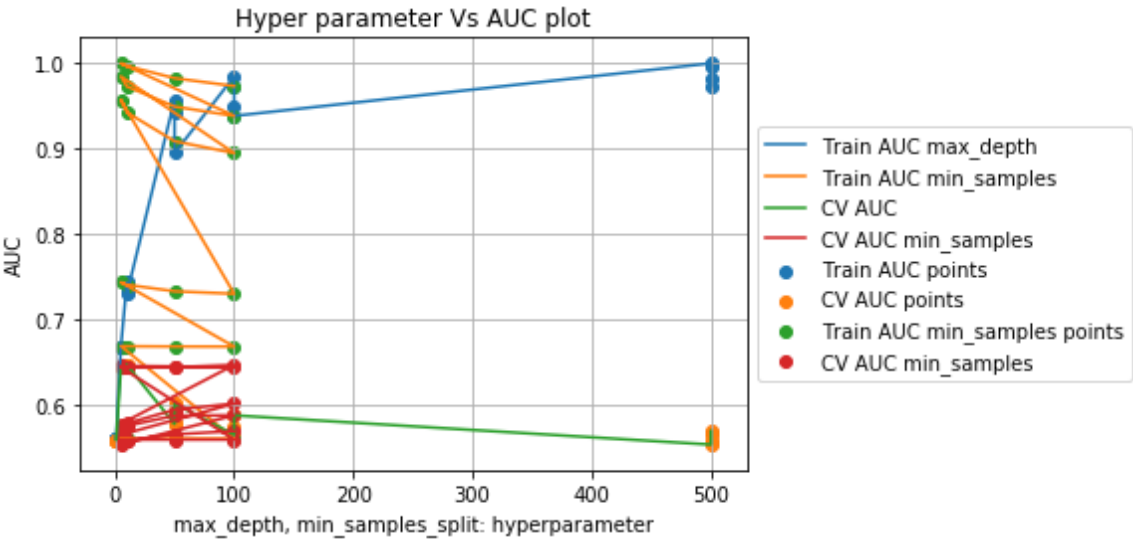
plt.scatter(max_depth, train_auc_max_depth, label='Train AUC points')
plt.scatter(max_depth, cv_auc_max_depth, label='CV AUC points')

plt.scatter(min_samples_split, train_auc_min_samples, label=('Train AUC min_samples poi
nts'))
plt.scatter(min_samples_split, cv_auc_min_samples, label=('CV AUC min_samples'))

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel("max_depth, min_samples_split: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()

results.head()

```



Out[84]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n
0	0.142558	0.029244	0.008374	0.001310	1	
1	0.136062	0.013681	0.008222	0.000558	1	
2	0.144735	0.030765	0.009876	0.003091	1	
3	0.136866	0.020790	0.008470	0.001016	1	
4	0.669828	0.029657	0.008010	0.003563	5	

5 rows × 32 columns

In [85]:

```
clf.best_params_
```

Out[85]:

```
{'max_depth': 10, 'min_samples_split': 100}
```

In []:

In [86]:

```
from sklearn.metrics import roc_curve, auc

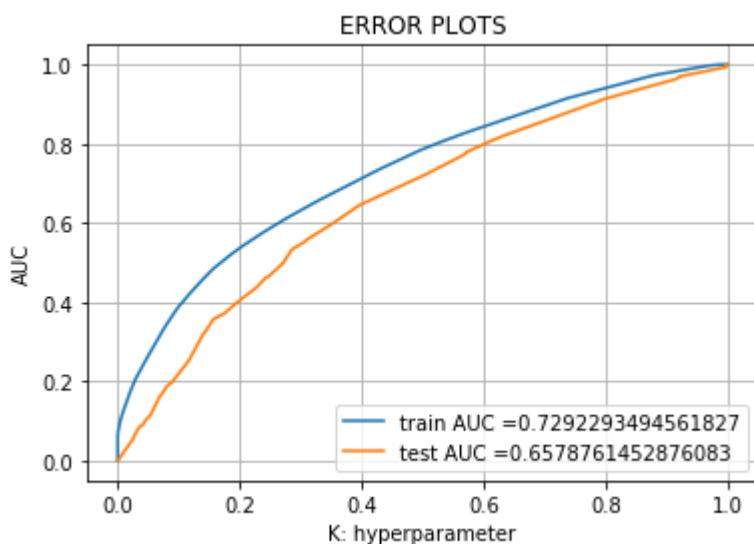
DTC = DecisionTreeClassifier(max_depth=5, min_samples_split=100, class_weight='balance
d')
DTC.fit(x_train_bow, y_train)

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

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

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

plt.show()
```



In [87]:

```
# we are writing our own function for predict, with defined threshould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
nd(t,3))
    return t

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

In [88]:

```
#Confusion matrix

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

the maximum value of $tpr*(1-fpr)$ 0.4425887798107272 for threshold 0.485

Train confusion matrix

```
[[ 3750  1418]
 [11051 17281]]
```

Test confusion matrix

```
[[1655   891]
 [5634 8320]]
```

In [89]:

```
import seaborn as sns

#Train Confusion matrix

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

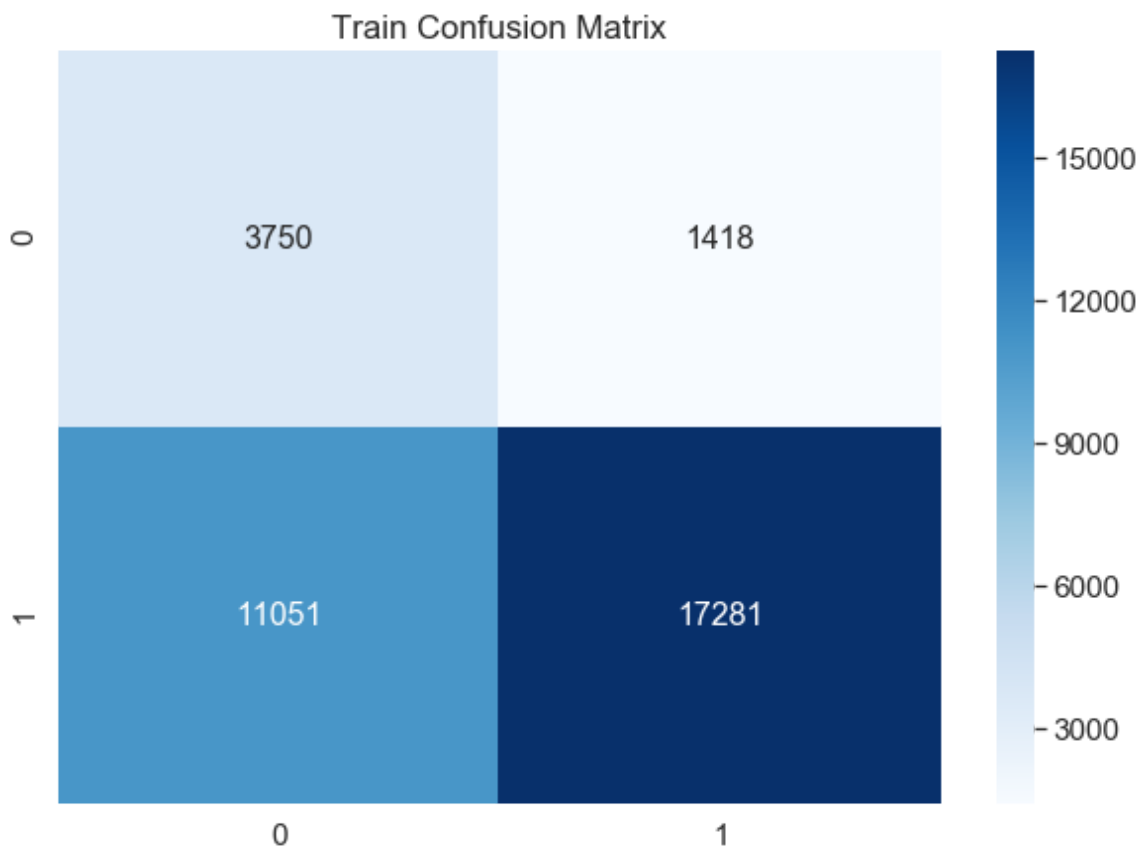
y_train_predicted= predict_with_best_t(y_train_pred, best_t)

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

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

Out[89]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b55ca55940>



In [90]:

```
train_fpr = []
for i in range(len(y_train)):
    if y_train[i]==0 and y_train_predicted[i]==1:
        train_fpr.append(y_train[i])
len(train_fpr)
```

Out[90]:

1418

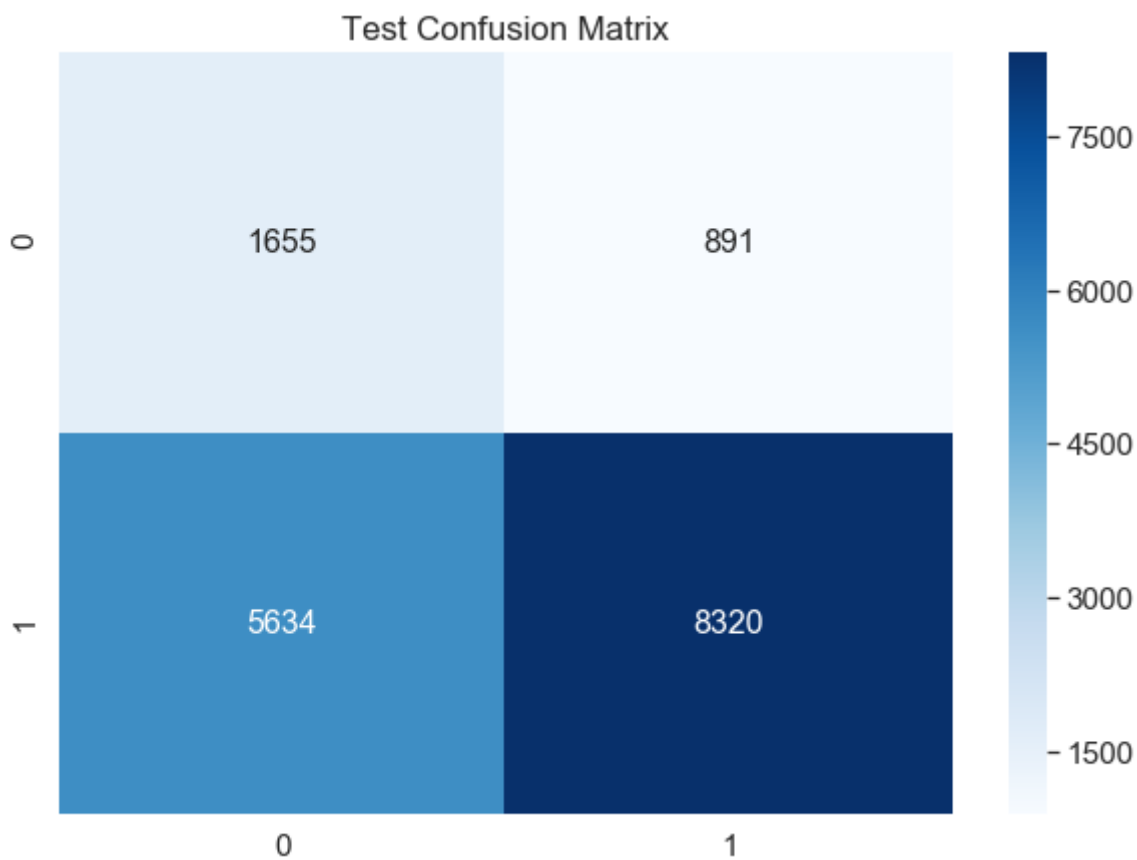
In [91]:

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

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

Out[91]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b55ca55f28>



In [92]:

```
#Plotting wordcloud using essay
```

In [93]:

```
test_fpr = []
for i in range(len(y_test)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        test_fpr.append(i)
```

In [94]:

```
essay_fpr = [x_test['essay'].iloc[i] for i in test_fpr]
print(len(essay_fpr))
```

891

In [95]:

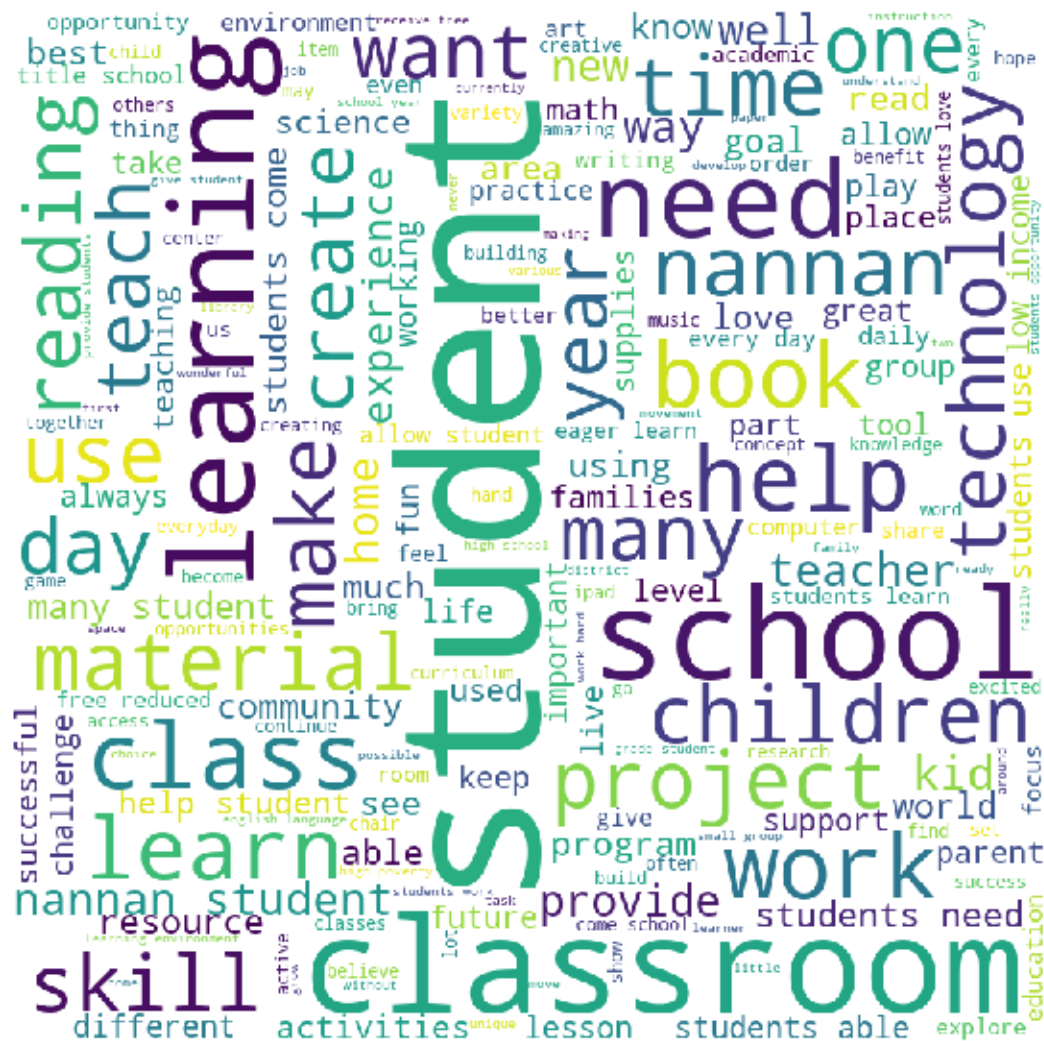
```
word=[]
for i in essay_fpr:
    word.append(i.split())
```

In [96]:

```
import nltk
from wordcloud import WordCloud, STOPWORDS
```


In [97]:

```
comment_words = ' '  
stopwords = set(STOPWORDS)  
  
for val in essay_fpr:  
  
    # typecaste each val to string  
    val = str(val)  
  
    # split the value  
    tokens = val.split()  
  
    # Converts each token into Lowercase  
    for i in range(len(tokens)):  
        tokens[i] = tokens[i].lower()  
  
    for words in tokens:  
        comment_words = comment_words + words + ' '  
  
wordcloud = WordCloud(width = 800, height = 800,  
                      background_color = 'white',  
                      stopwords = stopwords,  
                      min_font_size = 10).generate(comment_words)  
  
# plot the WordCloud image  
plt.figure(figsize = (8, 8), facecolor = None)  
plt.imshow(wordcloud)  
plt.axis("off")  
plt.tight_layout(pad = 0)  
  
plt.show()
```



In [98]:

```
#Plotting the boxplot with price
```

In [99]:

```
price_fpr = [x_test['price'].iloc[i] for i in test_fpr]
print(len(price_fpr))
```

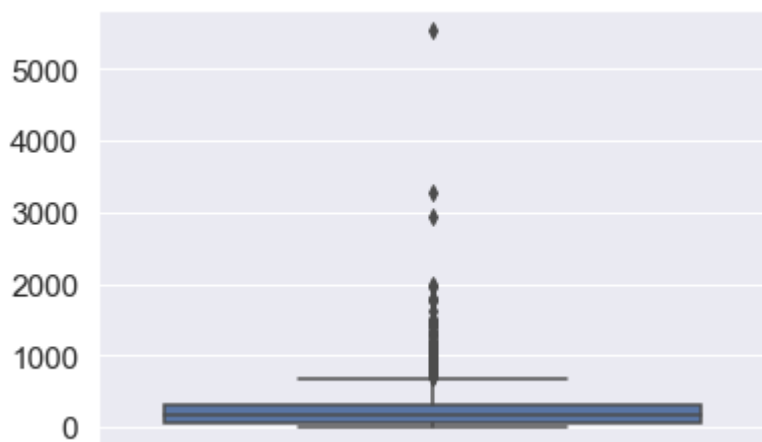
891

In [100]:

```
sns.boxplot(y=price_fpr)
```

Out[100]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b52734e048>



In [101]:

```
#plotting the pdf with teacher_number_of_previously_posted_projects
```

In [102]:

```
values = x_test['teacher_number_of_previously_posted_projects'].values
teacher_previous_projects_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        teacher_previous_projects_fpr.append(values[i])
print(len(teacher_previous_projects_fpr))
```

891

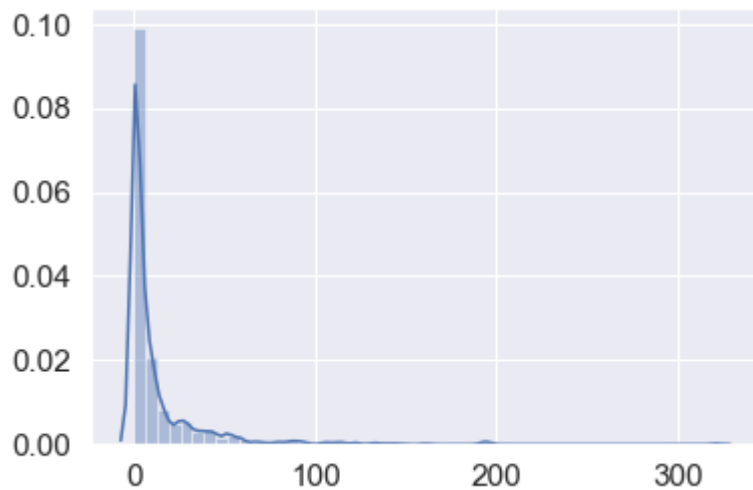
In [103]:

```
#Reference https://stackoverflow.com/questions/52908925/add-a-standard-normal-pdf-over-a-seaborn-histogram
```

```
pdf = sns.distplot(teacher_previous_projects_fpr, norm_hist=True, kde=True)  
pdf
```

Out[103]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b524281f60>



2.4.1.1 Graphviz visualization of Decision Tree on BOW, SET 1

In [104]:

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

In [105]:

```
from sklearn.metrics import roc_curve, auc

DTC = DecisionTreeClassifier(max_depth=2, min_samples_split=3, class_weight='balanced')
DTC.fit(x_train_bow, y_train)
```

Out[105]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=2,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=3,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                        splitter='best')
```

In [117]:

```
#Getting feature names
features = []
features.append(['price'])
features.append(['teacher_number_of_previously_posted_projects'])
features.append(['quantity'])
features.extend(state_vectorizer.get_feature_names())
features.extend(teacher_vectorizer.get_feature_names())
features.extend(grade_vectorizer.get_feature_names())
features.extend(categories_vectorizer.get_feature_names())
features.extend(subcategories_vectorizer.get_feature_names())
features.extend(essay_bow_vectorizer.get_feature_names())
features.extend(title_bow_vectorizer.get_feature_names())
len(features)
```

Out[117]:

5871

In []:

In [120]:

```
from sklearn import tree
import pydot
from io import StringIO

#References https://stackoverflow.com/questions/27817994/visualizing-decision-tree-in-scikit-learn

tree.export_graphviz(DTC, out_file='tree.dot') #produces dot file

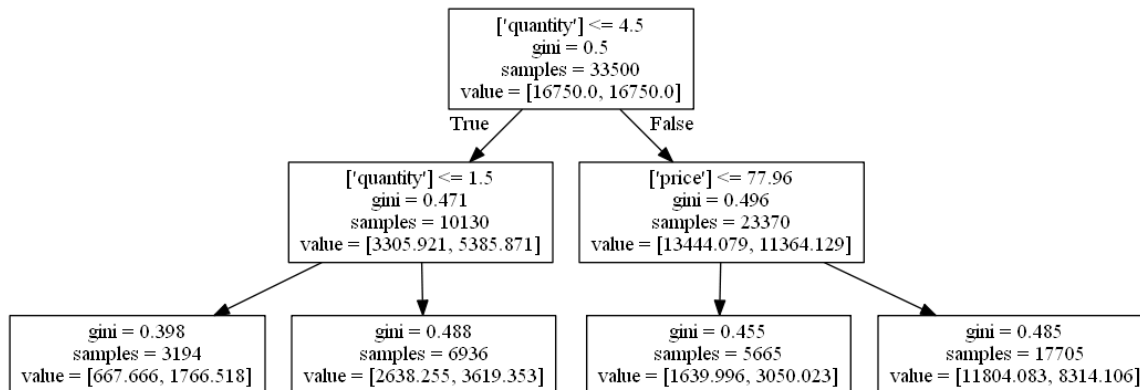
dotfile = StringIO()
tree.export_graphviz(DTC, feature_names=features, out_file=dotfile)
(graph, ) = pydot.graph_from_dot_data(dotfile.getvalue())
graph.write_png("dtree1.png")
```

In [121]:

```
from IPython.display import Image
```

```
Image('dtree1.png')
```

Out[121]:



2.4.2 Applying Decision Trees on TFIDF, SET 2

In [122]:

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

In [123]:

```
#Merging Features
```

```

x_train_tfidf = hstack((x_train_price, x_train_previous_projects, x_train_quantity, x_train_state_one_hot, x_train_teacher_one_hot, \
                        x_train_grade_one_hot, x_train_categories_one_hot, x_train_subcategories_one_hot, x_train_essay_tfidf, \
                        x_train_title_tfidf)).tocsr()

x_test_tfidf = hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_test_state_one_hot, x_test_teacher_one_hot, \
                      x_test_grade_one_hot, x_test_categories_one_hot, x_test_subcategories_one_hot, x_test_essay_tfidf, \
                      x_test_title_tfidf)).tocsr()

```

In [124]:

```

from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

tuned_parameters = {'max_depth':[1, 5, 10, 50, 100, 500], 'min_samples_split':[5, 10, 50, 100]}
DTC = DecisionTreeClassifier(class_weight='balanced')
clf = GridSearchCV(DTC, tuned_parameters, cv=10, scoring='roc_auc')
clf.fit(x_train_tfidf, y_train)

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

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

#For min_samples_split
train_auc_min_samples = results['mean_train_score']
train_auc_std_min_samples = results['std_train_score']
cv_auc_min_samples = results['mean_test_score']
cv_auc_std_min_samples = results['std_test_score']
min_samples_split = results['param_min_samples_split']

plt.plot(max_depth, train_auc_max_depth, label='Train AUC max_depth')
# 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(min_samples_split, train_auc_min_samples, label='Train AUC min_samples') #Plot
ting _min_samples-split curve

plt.plot(max_depth, cv_auc_max_depth, 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='da
rkorange')
plt.plot(min_samples_split, cv_auc_min_samples, label='CV AUC min_samples')

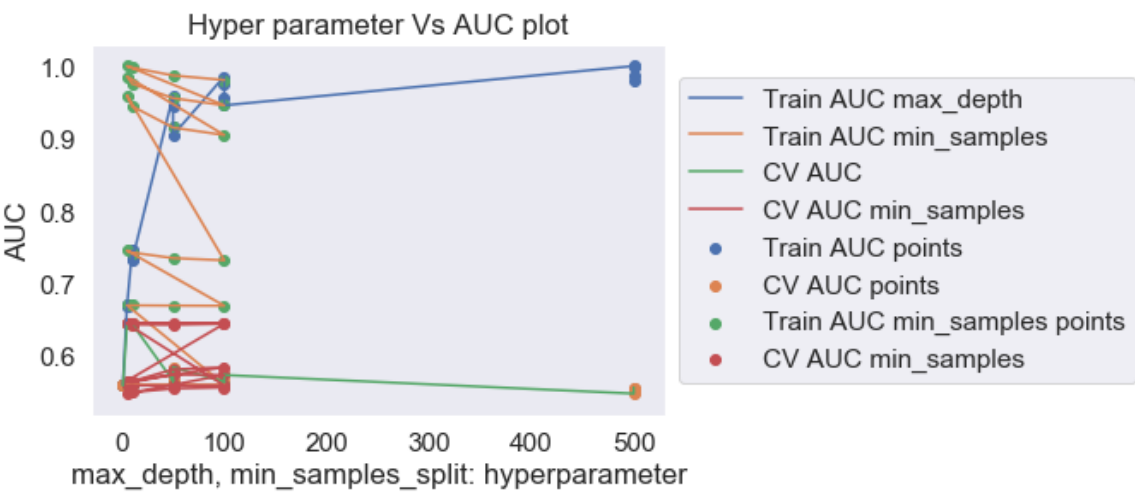
plt.scatter(max_depth, train_auc_max_depth, label='Train AUC points')
plt.scatter(max_depth, cv_auc_max_depth, label='CV AUC points')

plt.scatter(min_samples_split, train_auc_min_samples, label=('Train AUC min_samples poi
nts'))
plt.scatter(min_samples_split, cv_auc_min_samples, label=('CV AUC min_samples'))

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel("max_depth, min_samples_split: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()

results.head()

```



Out[124]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n
0	0.340137	0.072396	0.006927	0.002456	1	
1	0.384475	0.103413	0.011899	0.003835	1	
2	0.342468	0.061538	0.008327	0.001032	1	
3	0.324950	0.032711	0.008123	0.001047	1	
4	1.440195	0.068820	0.008581	0.001148	5	

5 rows × 32 columns

In [125]:

```
clf.best_params_
```

Out[125]:

```
{'max_depth': 10, 'min_samples_split': 5}
```


In [126]:

```
from sklearn.metrics import roc_curve, auc

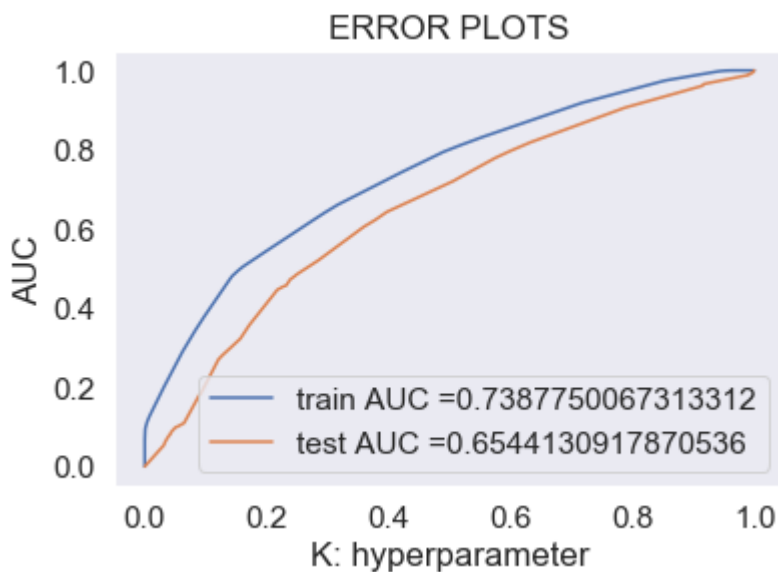
DTC_set2 = DecisionTreeClassifier(max_depth=10, min_samples_split=5, class_weight='balanced')
DTC_set2.fit(x_train_tfidf, y_train)

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

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

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

plt.show()
```



In [127]:

```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshold, fpr, tpr):
    t = threshold[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.rou
nd(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 [128]:

```
#Confusion matrix

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

the maximum value of $tpr*(1-fpr)$ 0.4517061957532294 for threshold 0.5

Train confusion matrix

```
[[ 3642  1526]
 [10172 18160]]
```

Test confusion matrix

```
[[1584   962]
 [5271  8683]]
```

In [129]:

```
import seaborn as sns

#Train Confusion matrix

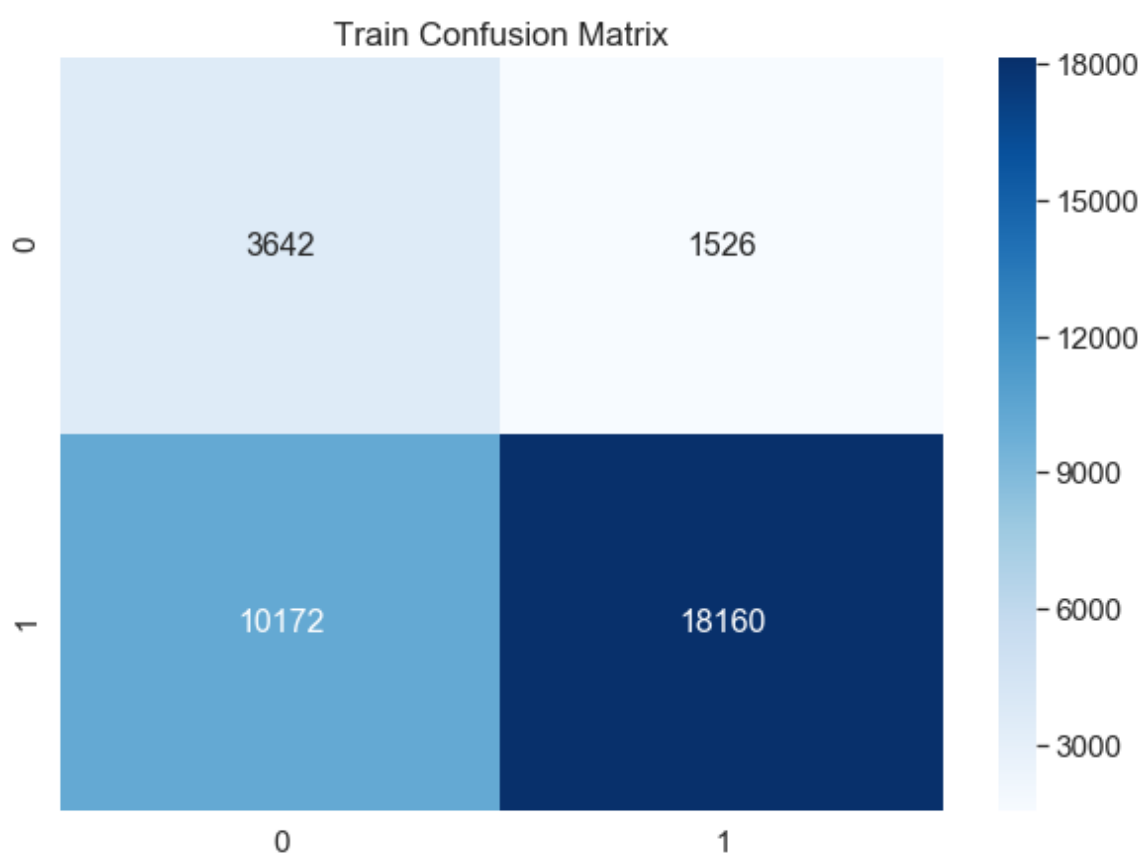
#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t
)), columns=np.unique(y_train), index = np.unique(y_train))

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

Out[129]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b55c559d30>



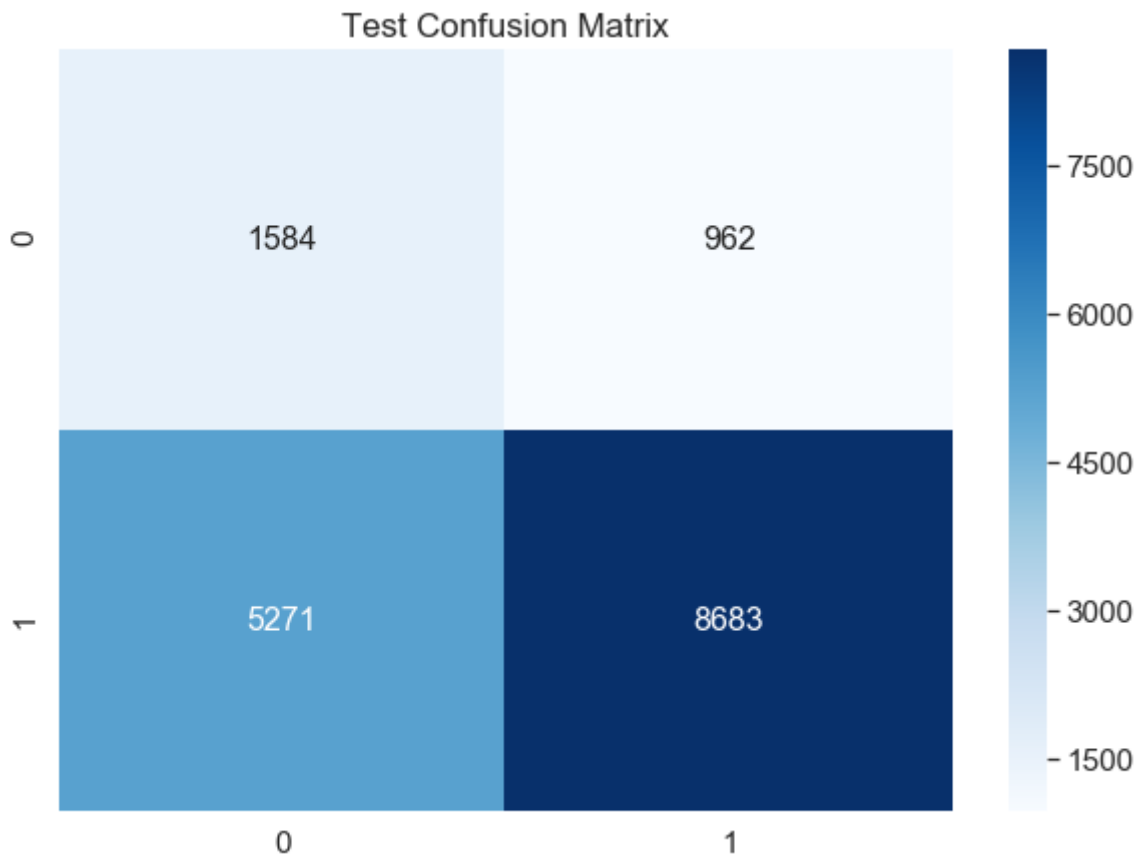
In [130]:

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

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

Out[130]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b500383dd8>



In [131]:

```
#Getting all the false positive points

test_fpr = []
for i in range(len(y_test)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        test_fpr.append(i)
```

In [132]:

```
#Plotting the word cloud
```

In [133]:

```
values = x_test['essay'].values
essay_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        essay_fpr.append(values[i])

print(len(essay_fpr))
```

891

In [134]:

```
word=[]
for i in essay_fpr:
    word.append(i.split())
```

In [135]:

```
import nltk
from wordcloud import WordCloud, STOPWORDS

comment_words = ' '
stopwords = set(STOPWORDS)

for val in essay_fpr:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

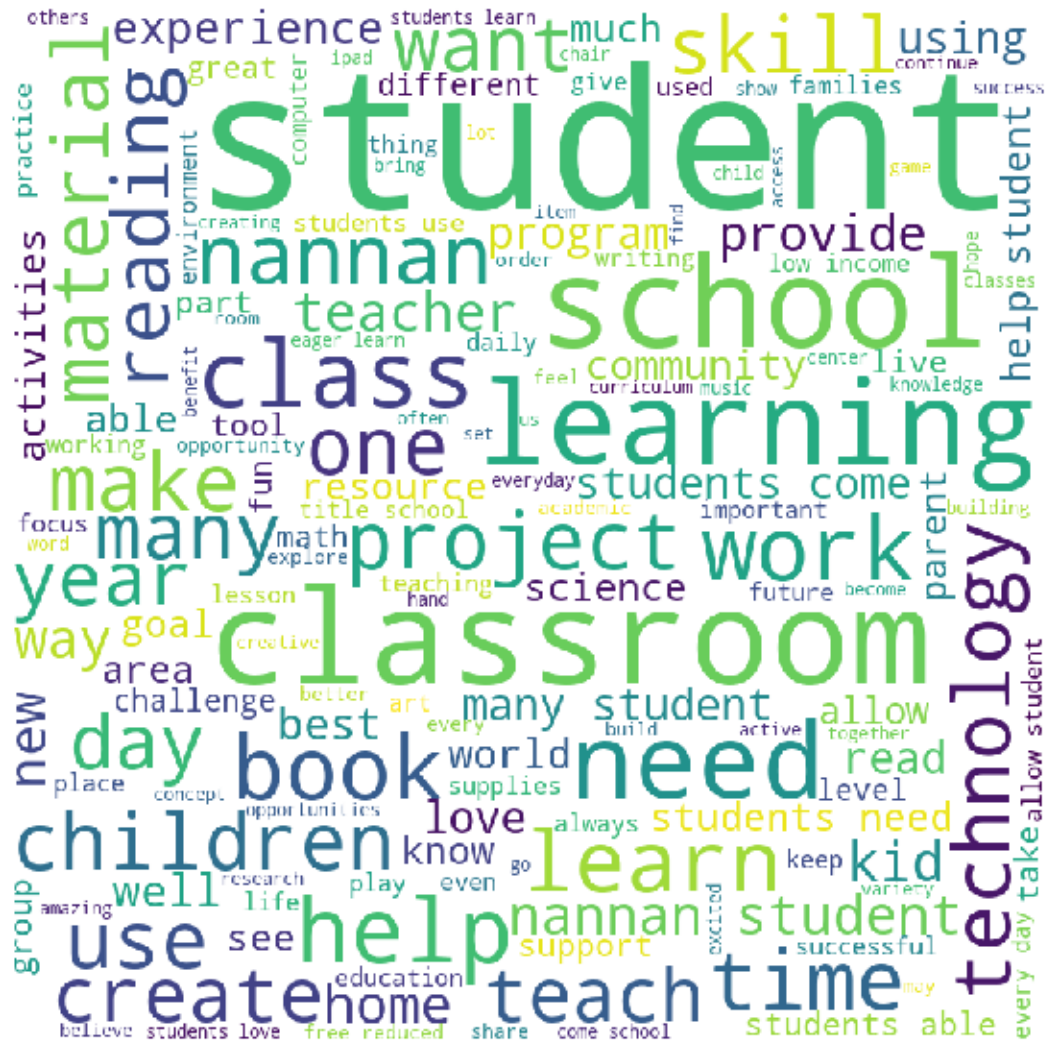
    # Converts each token into Lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800,
                       background_color = 'white',
                       stopwords = stopwords,
                       min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



In [136]:

```
#Plotting the boxplot with price
```

In [137]:

```
values = x_test['price'].values
price_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        price_fpr.append(values[i])
print(len(price_fpr))
```

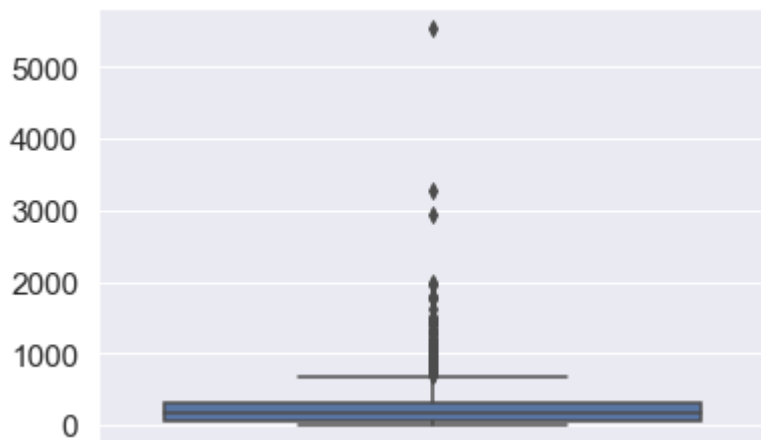
891

In [138]:

```
sns.boxplot(y=price_fpr)
```

Out[138]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b524c444e0>



In [139]:

```
#plotting the pdf with teacher_number_of_previously_posted_projects
```

In [140]:

```
values = x_test['teacher_number_of_previously_posted_projects'].values
teacher_previous_projects_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        teacher_previous_projects_fpr.append(values[i])
print(len(teacher_previous_projects_fpr))
```

891

In [141]:

```
#Reference https://stackoverflow.com/questions/15415455/plotting-probability-density-function-by-sample-with-matplotlib
from scipy.stats.kde import gaussian_kde
from numpy import linspace

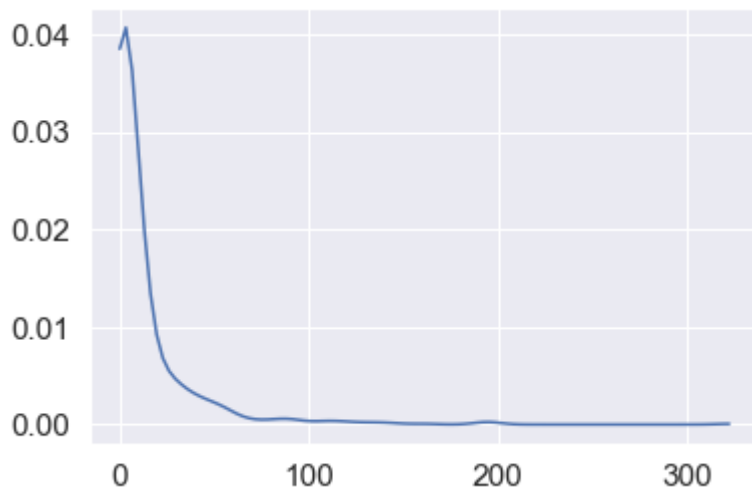
kde = gaussian_kde(teacher_previous_projects_fpr)

dist_space = linspace( min(teacher_previous_projects_fpr), max(teacher_previous_projects_fpr), 100 )

plt.plot( dist_space, kde(dist_space) )
```

Out[141]:

[<matplotlib.lines.Line2D at 0x1b52718d278>]



In []:

2.4.2.1 Graphviz visualization of Decision Tree on TFIDF, SET 2

In [142]:

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

In [143]:

```
from sklearn.metrics import roc_curve, auc

DTC = DecisionTreeClassifier(max_depth=2, min_samples_split=3, class_weight='balanced')
DTC.fit(x_train_tfidf, y_train)
```

Out[143]:

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_dept
h=2,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=3,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
e,
                        splitter='best')
```

In [146]:

```
#getting feature names
features = []

features.append(['price'])
features.append(['teacher_number_of_previously_posted_projects'])
features.append(['quantity'])
features.extend(state_vectorizer.get_feature_names())
features.extend(teacher_vectorizer.get_feature_names())
features.extend(grade_vectorizer.get_feature_names())
features.extend(categories_vectorizer.get_feature_names())
features.extend(subcategories_vectorizer.get_feature_names())
features.extend(essay_tfidf_vectorizer.get_feature_names())
features.extend(title_tfidf_vectorizer.get_feature_names())
len(features)
```

Out[146]:

5871

In [150]:

```
import pydot
from io import StringIO
from sklearn import tree

#References https://stackoverflow.com/questions/27817994/visualizing-decision-tree-in-scikit-learn

tree.export_graphviz(DTC, out_file='tree.dot')

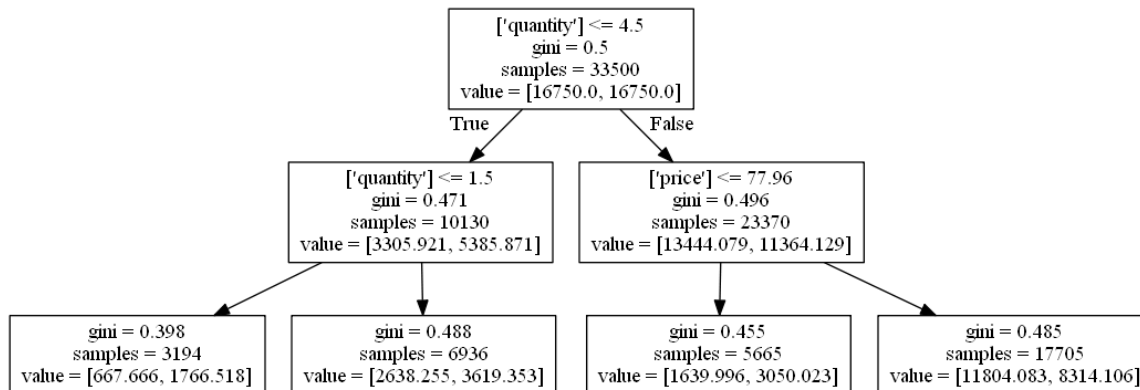
dotfile = StringIO()
tree.export_graphviz(DTC, feature_names=features, out_file=dotfile)
(graph, ) = pydot.graph_from_dot_data(dotfile.getvalue())
graph.write_png("dtree2.png")
```

In [151]:

```
from IPython.display import Image
```

```
Image('dtree2.png')
```

Out[151]:



2.4.3 Applying Decision Trees on AVG W2V, SET 3

In [152]:

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

In [153]:

```
#Merging Features
```

```

x_train_avg_w2v = hstack((x_train_price, x_train_previous_projects, x_train_quantity, x_train_state_one_hot, x_train_teacher_one_hot, \
                           x_train_grade_one_hot, x_train_categories_one_hot, x_train_subcategories_one_hot, x_train_essay_avg_w2v, \
                           x_train_title_avg_w2v)).tocsr()

x_test_avg_w2v = hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_test_state_one_hot, x_test_teacher_one_hot, \
                          x_test_grade_one_hot, x_test_categories_one_hot, x_test_subcategories_one_hot, x_test_essay_avg_w2v, \
                          x_test_title_avg_w2v)).tocsr()

```

In [156]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

tuned_parameters = {'max_depth':[1, 5, 10, 50, 100], 'min_samples_split':[5, 10, 50, 100]}
DTC = DecisionTreeClassifier(class_weight='balanced')
clf = GridSearchCV(DTC, tuned_parameters, cv=10, scoring='roc_auc')
clf.fit(x_train_avg_w2v, y_train)
```

Out[156]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
             estimator=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=None,
             splitter='best'),
             fit_params=None, iid='warn', n_jobs=None,
             param_grid={'max_depth': [1, 5, 10, 50, 100], 'min_samples_split':
[5, 10, 50, 100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
             scoring='roc_auc', verbose=0)
```

In [157]:

```

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

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

#For min_samples_split
train_auc_min_samples = results['mean_train_score']
train_auc_std_min_samples = results['std_train_score']
cv_auc_min_samples = results['mean_test_score']
cv_auc_std_min_samples = results['std_test_score']
min_samples_split = results['param_min_samples_split']

plt.plot(max_depth, train_auc_max_depth, label='Train AUC max_depth')
# 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(min_samples_split, train_auc_min_samples, label='Train AUC min_samples') #Plot
ting _min_samples-split curve

plt.plot(max_depth, cv_auc_max_depth, 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='da
rkorange')
plt.plot(min_samples_split, cv_auc_min_samples, label='CV AUC min_samples')

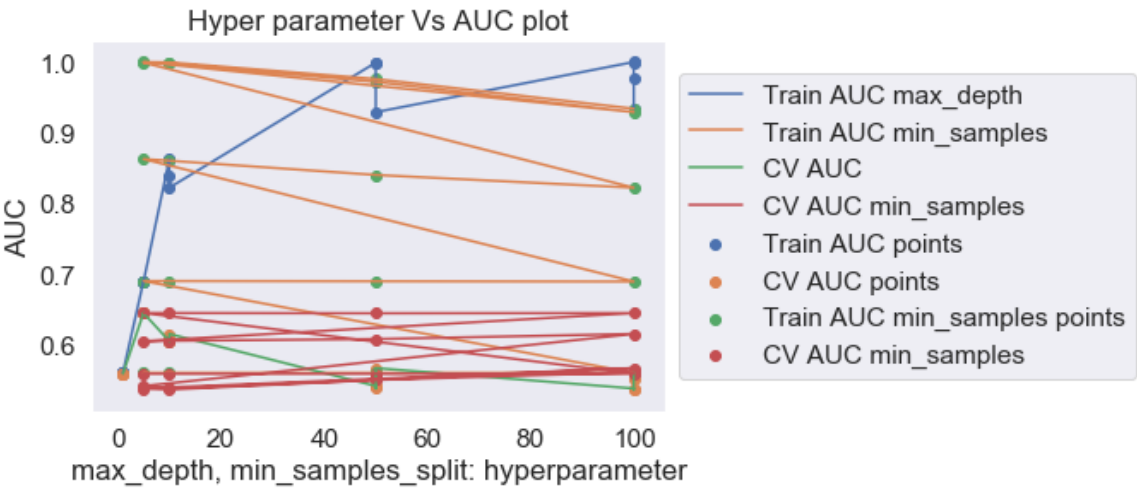
plt.scatter(max_depth, train_auc_max_depth, label='Train AUC points')
plt.scatter(max_depth, cv_auc_max_depth, label='CV AUC points')

plt.scatter(min_samples_split, train_auc_min_samples, label=('Train AUC min_samples poi
nts'))
plt.scatter(min_samples_split, cv_auc_min_samples, label=('CV AUC min_samples'))

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel("max_depth, min_samples_split: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()

results.head()

```



Out[157]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n
0	5.150788	0.339405	0.112220	0.021373	1	
1	4.790684	0.067502	0.103318	0.006303	1	
2	4.782198	0.059545	0.101782	0.005550	1	
3	4.778852	0.080818	0.101776	0.008331	1	
4	22.934586	0.337322	0.107333	0.007506	5	

5 rows × 32 columns

In [158]:

```
clf.best_params_
```

Out[158]:

```
{'max_depth': 5, 'min_samples_split': 5}
```

In [159]:

```
from sklearn.metrics import roc_curve, auc

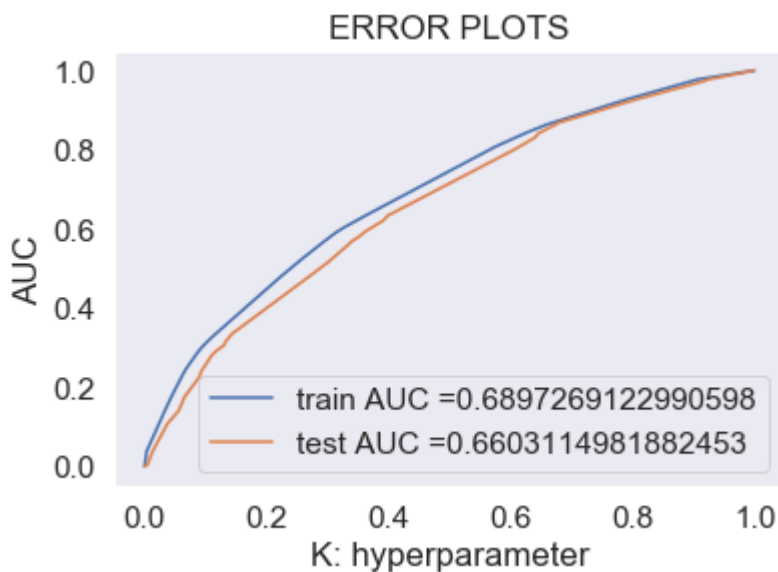
DTC_set3 = DecisionTreeClassifier(max_depth=5, min_samples_split=5, class_weight='balanced')
DTC_set3.fit(x_train_avg_w2v, y_train)

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

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

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

plt.show()
```



In [160]:

```
# we are writing our own function for predict, with defined threshould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
nd(t,3))
    return t

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

In [161]:

```
#Confusion matrix

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

the maximum value of $tpr*(1-fpr)$ 0.4061738217657156 for threshold 0.49

Train confusion matrix

```
[[ 3484 1684]
 [11262 17070]]
```

Test confusion matrix

```
[[1625  921]
 [5677 8277]]
```


In [162]:

```
import seaborn as sns

#Train Confusion matrix

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

df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t
)), columns=np.unique(y_train), index = np.unique(y_train))

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

Out[162]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b5242174e0>



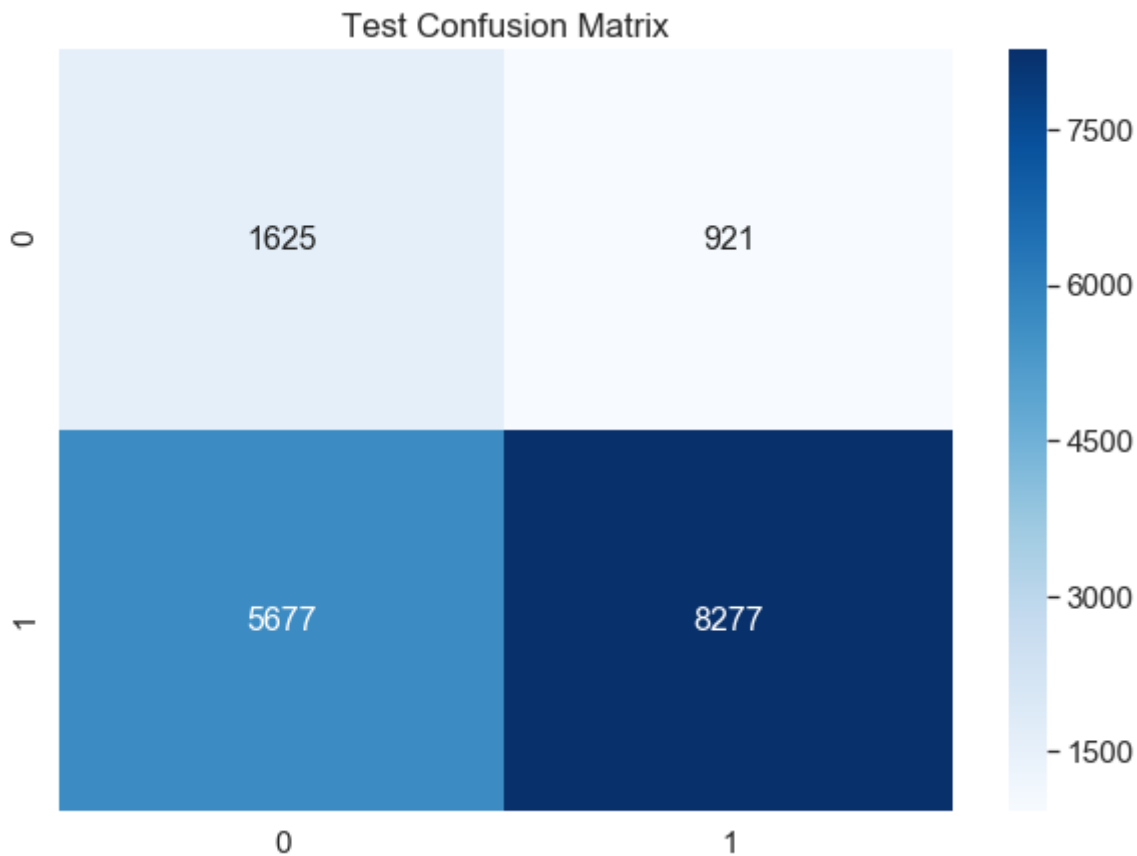
In [163]:

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

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

Out[163]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b52388d7b8>



In [164]:

```
#Getting all the false positive points
```

In [165]:

```
test_fpr = []
for i in range(len(y_test)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        test_fpr.append(i)
```

In [166]:

```
#Plotting the word cloud
```

In [167]:

```
values = x_test['essay'].values
essay_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        essay_fpr.append(values[i])

print(len(essay_fpr))
```

891

In [168]:

```
word=[]
for i in essay_fpr:
    word.append(i.split())
```

In [169]:

```
import nltk
from wordcloud import WordCloud, STOPWORDS

comment_words = ' '
stopwords = set(STOPWORDS)

for val in essay_fpr:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

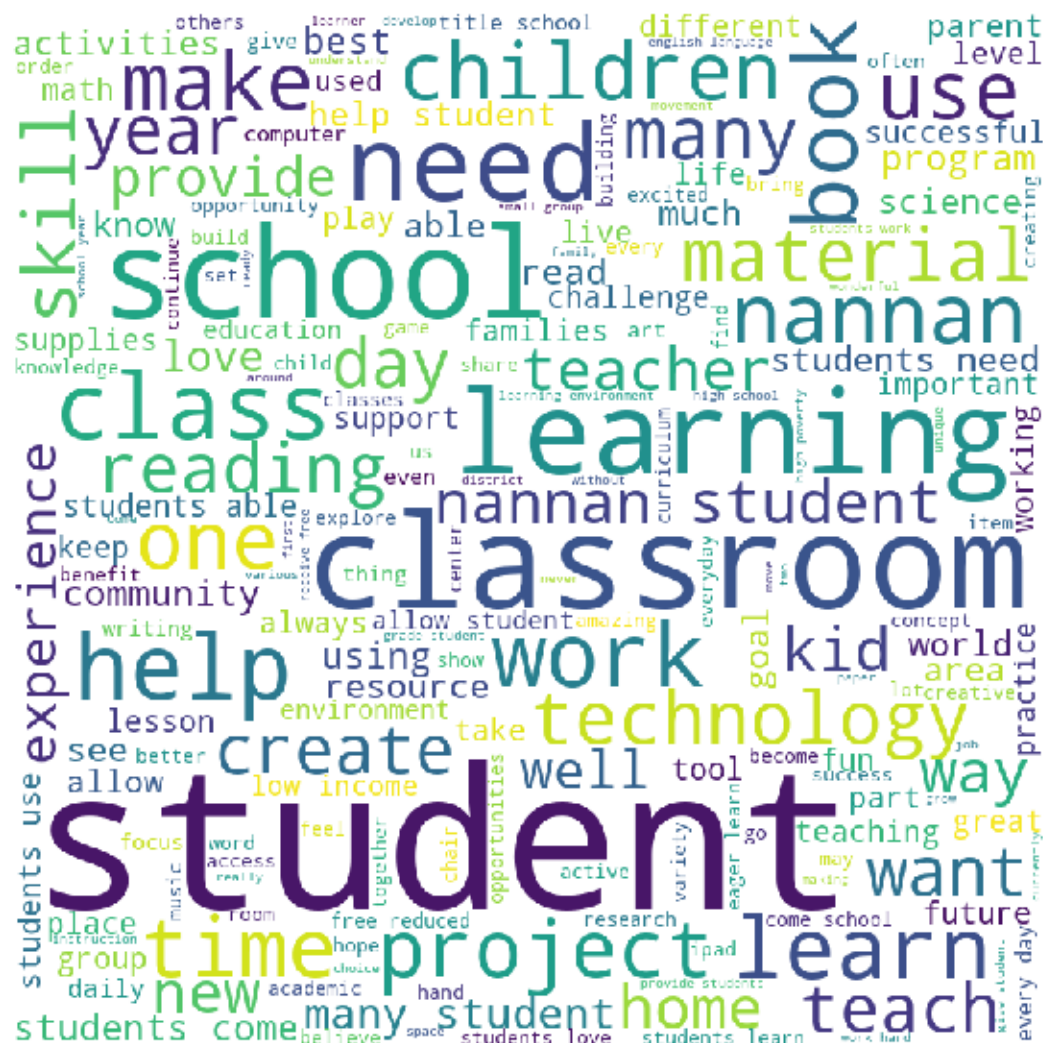
    # Converts each token into Lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800,
                       background_color = 'white',
                       stopwords = stopwords,
                       min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



```
#Plotting the boxplot with price

values = x_test['price'].values
price_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        price_fpr.append(values[i])

print(len(price_fpr))
```

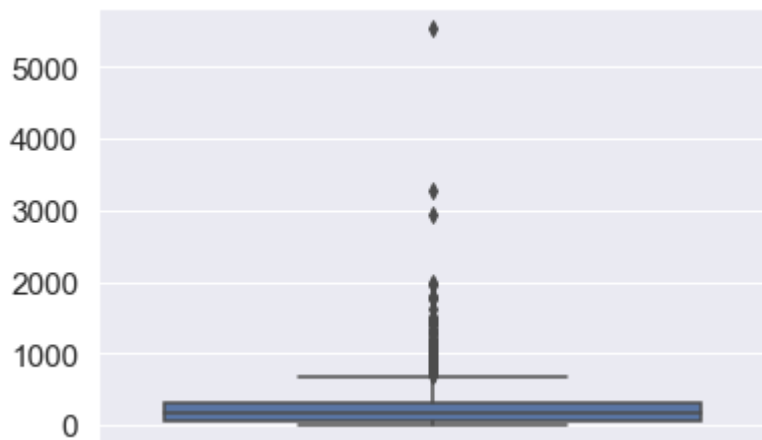
file:///C:/Users/hp/Downloads/8 DonorsChoose DT.html

In [171]:

```
sns.boxplot(y=price_fpr)
```

Out[171]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b5891c0e80>



In [172]:

```
#plotting the pdf with teacher_number_of_previously_posted_projects
```

In [173]:

```
values = x_test['teacher_number_of_previously_posted_projects'].values
teacher_previous_projects_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        teacher_previous_projects_fpr.append(values[i])

print(len(teacher_previous_projects_fpr))
```

891

In [174]:

```
#Reference https://stackoverflow.com/questions/15415455/plotting-probability-density-function-by-sample-with-matplotlib
from scipy.stats.kde import gaussian_kde
from numpy import linspace

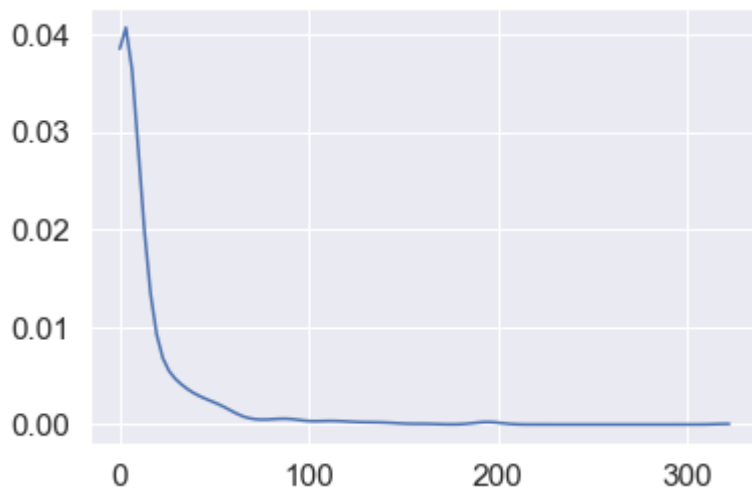
kde = gaussian_kde(teacher_previous_projects_fpr)

dist_space = linspace( min(teacher_previous_projects_fpr), max(teacher_previous_projects_fpr), 100 )

plt.plot( dist_space, kde(dist_space) )
```

Out[174]:

[<matplotlib.lines.Line2D at 0x1b521f809b0>]



In []:

In []:

2.4.4 Applying Decision Trees on TFIDF W2V, SET 4

In [175]:

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

In [176]:

#Merging Features

```
x_train_tfidf_w2v = hstack((x_train_price, x_train_previous_projects, x_train_quantity,
x_train_state_one_hot, x_train_teacher_one_hot, \
                           x_train_grade_one_hot, x_train_categories_one_hot, x_train_subcate
gories_one_hot, x_train_essay_tfidf_w2v, \
                           x_train_title_tfidf_w2v)).tocsr()
```

```
x_test_tfidf_w2v = hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_t
est_state_one_hot, x_test_teacher_one_hot, \
                           x_test_grade_one_hot, x_test_categories_one_hot, x_test_subcategor
ies_one_hot, x_test_essay_tfidf_w2v, \
                           x_test_title_tfidf_w2v)).tocsr()
```


In [177]:

```

from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

tuned_parameters = {'max_depth':[1, 5, 10, 50, 100, 500], 'min_samples_split':[5, 10, 50, 100]}
DTC = DecisionTreeClassifier(class_weight='balanced')
clf = GridSearchCV(DTC, tuned_parameters, cv=10, scoring='roc_auc')
clf.fit(x_train_avg_w2v, y_train)

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

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

#For min_samples_split
train_auc_min_samples = results['mean_train_score']
train_auc_std_min_samples = results['std_train_score']
cv_auc_min_samples = results['mean_test_score']
cv_auc_std_min_samples = results['std_test_score']
min_samples_split = results['param_min_samples_split']

plt.plot(max_depth, train_auc_max_depth, label='Train AUC max_depth')
# 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(min_samples_split, train_auc_min_samples, label='Train AUC min_samples') #Plot
ting _min_samples-split curve

plt.plot(max_depth, cv_auc_max_depth, 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='da
rkorange')
plt.plot(min_samples_split, cv_auc_min_samples, label='CV AUC min_samples')

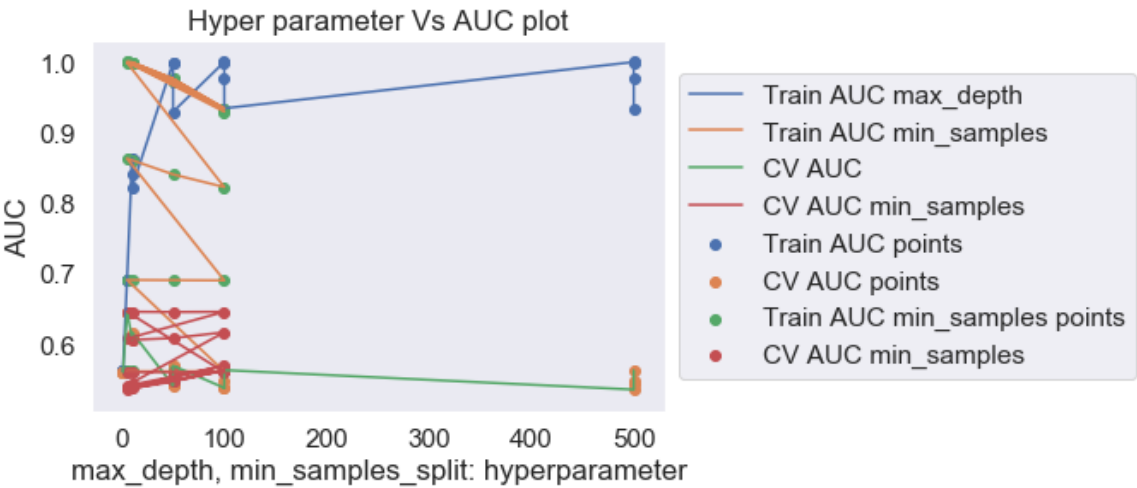
plt.scatter(max_depth, train_auc_max_depth, label='Train AUC points')
plt.scatter(max_depth, cv_auc_max_depth, label='CV AUC points')

plt.scatter(min_samples_split, train_auc_min_samples, label=('Train AUC min_samples poi
nts'))
plt.scatter(min_samples_split, cv_auc_min_samples, label=('CV AUC min_samples'))

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel("max_depth, min_samples_split: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()

results.head()

```



Out[177]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n
0	5.004209	0.295018	0.107509	0.006885	1	
1	4.684733	0.119384	0.103218	0.009065	1	
2	5.015690	0.369614	0.112798	0.019882	1	
3	4.724663	0.039653	0.103178	0.004986	1	
4	23.571398	1.369166	0.104841	0.009355	5	

5 rows × 32 columns

In [178]:

```
clf.best_params_
```

Out[178]:

```
{'max_depth': 5, 'min_samples_split': 5}
```

In [179]:

```
from sklearn.metrics import roc_curve, auc

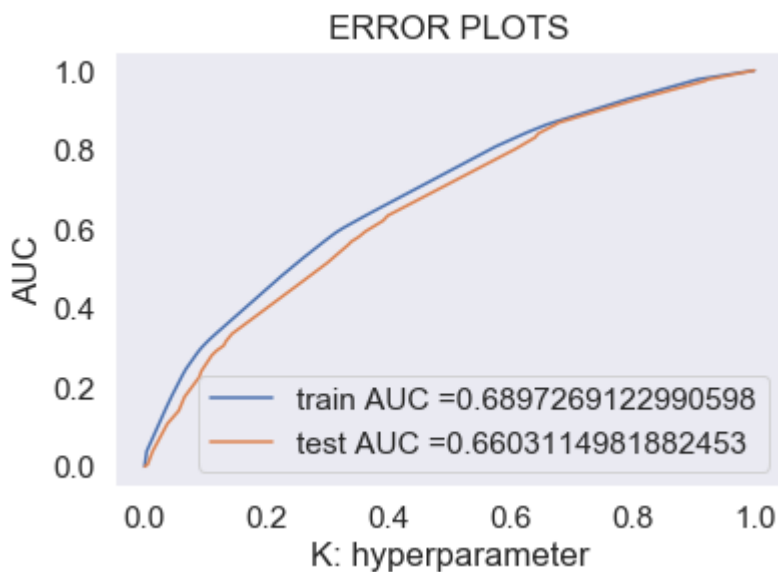
DTC_set4 = DecisionTreeClassifier(max_depth=5, min_samples_split=5, class_weight='balanced')
DTC_set4.fit(x_train_avg_w2v, y_train)

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

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

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

plt.show()
```



In [181]:

```
# we are writing our own function for predict, with defined threshould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
nd(t,3))
    return t

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

In [182]:

```
#Confusion matrix

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

the maximum value of tpr*(1-fpr) 0.4061738217657156 for threshold 0.49

Train confusion matrix

```
[[ 3484 1684]
 [11262 17070]]
```

Test confusion matrix

```
[[1625  921]
 [5677 8277]]
```

In [183]:

```
import seaborn as sns

#Train Confusion matrix

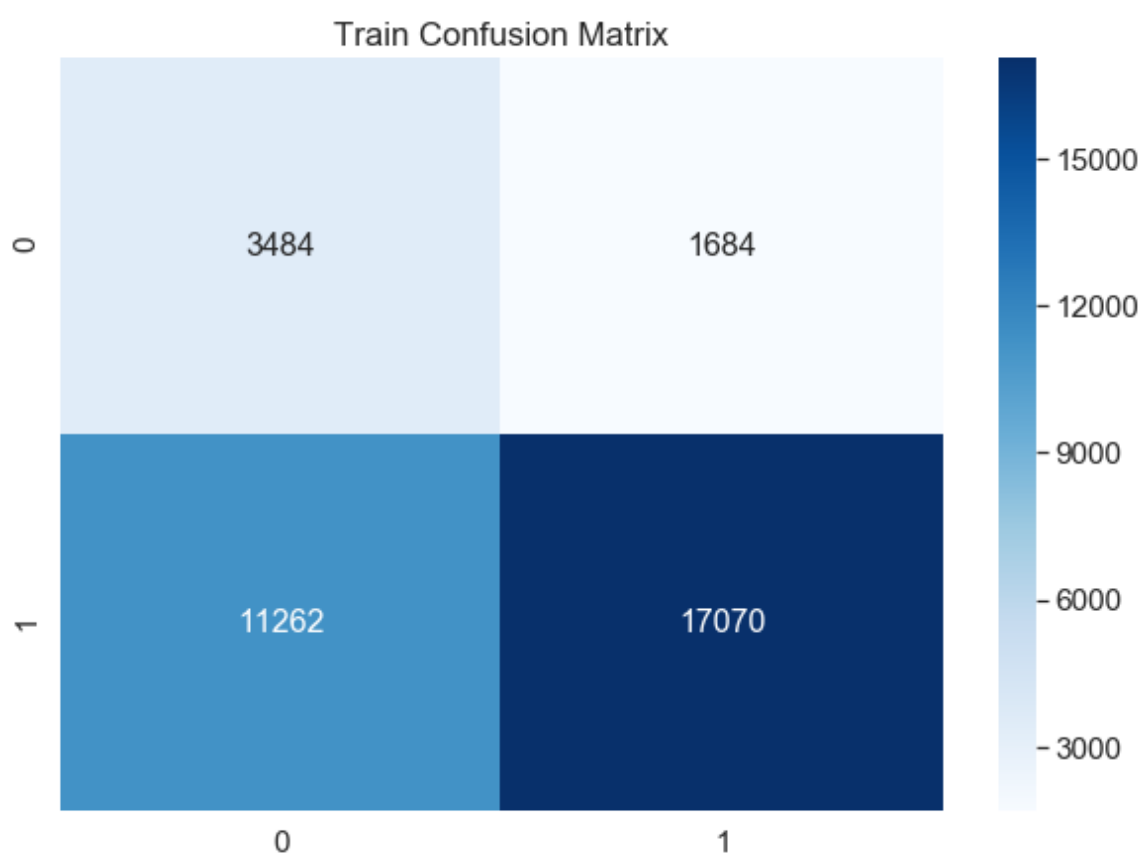
#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t
)), columns=np.unique(y_train), index = np.unique(y_train))

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

Out[183]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b524c5f898>



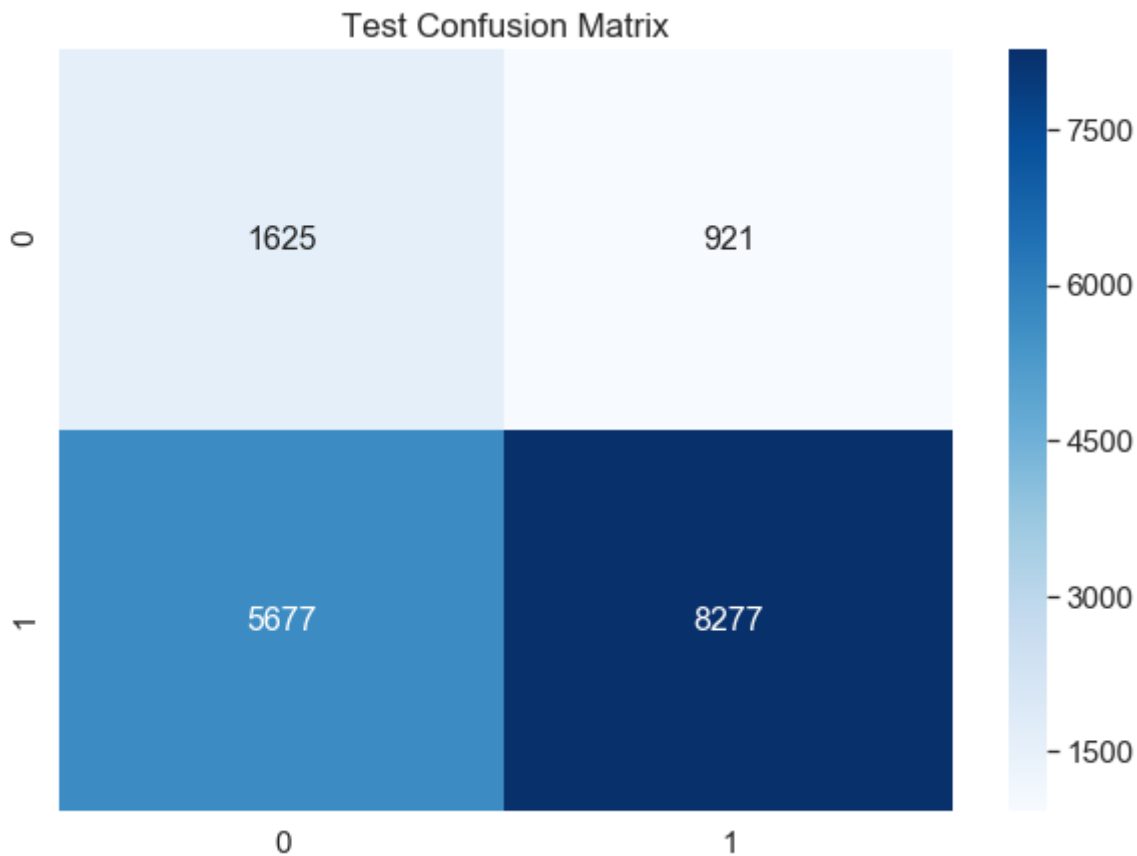
In [184]:

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

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

Out[184]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b57a208a58>



In [185]:

```
#Getting all the false positive points
```

In [186]:

```
test_fpr = []
for i in range(len(y_test)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        test_fpr.append(i)
```

In [187]:

```
#Plotting the word cloud
```

In [188]:

```
values = x_test['essay'].values
essay_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        essay_fpr.append(values[i])

print(len(essay_fpr))
```

891

In [189]:

```
word=[]
for i in essay_fpr:
    word.append(i.split())
```

In [190]:

```
import nltk
from wordcloud import WordCloud, STOPWORDS

comment_words = ' '
stopwords = set(STOPWORDS)

for val in essay_fpr:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

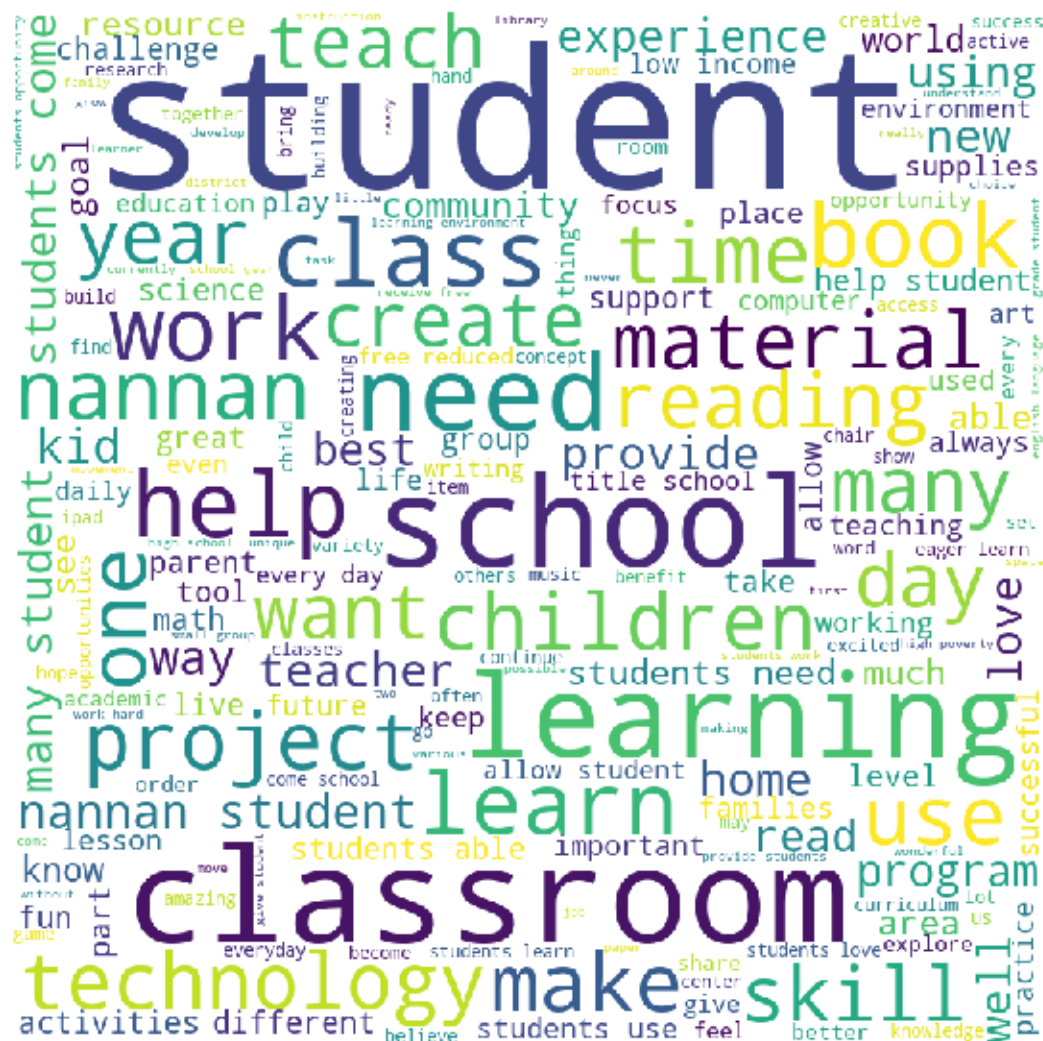
    # Converts each token into Lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800,
                       background_color = 'white',
                       stopwords = stopwords,
                       min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

```
#Plotting the boxplot with price

values = x_test['price'].values
price_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        price_fpr.append(values[i])

print(len(price_fpr))
```

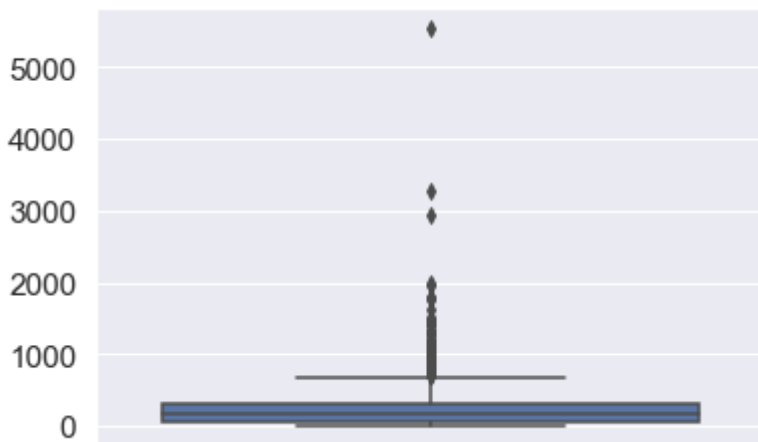
file:///C:/Users/hp/Downloads/8 DonorsChoose DT.html

In [192]:

```
sns.boxplot(y=price_fpr)
```

Out[192]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b526a24080>



In [193]:

```
#plotting the pdf with teacher_number_of_previously_posted_projects
```

In [194]:

```
values = x_test['teacher_number_of_previously_posted_projects'].values
teacher_previous_projects_fpr = []
for i in range(len(values)):
    if y_test[i]==0 and y_test_predicted[i]==1:
        teacher_previous_projects_fpr.append(values[i])

print(len(teacher_previous_projects_fpr))
```

891

In [195]:

```
#Reference https://stackoverflow.com/questions/15415455/plotting-probability-density-function-by-sample-with-matplotlib
from scipy.stats.kde import gaussian_kde
from numpy import linspace

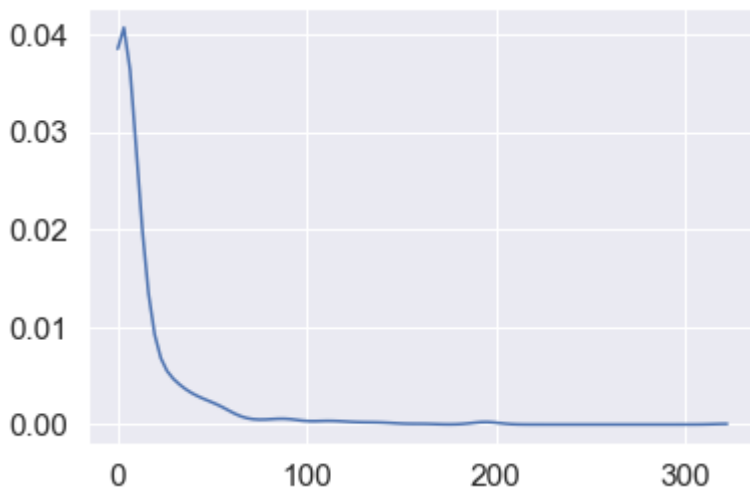
kde = gaussian_kde(teacher_previous_projects_fpr)

dist_space = linspace( min(teacher_previous_projects_fpr), max(teacher_previous_projects_fpr), 100 )

plt.plot( dist_space, kde(dist_space) )
```

Out[195]:

[<matplotlib.lines.Line2D at 0x1b5250b4c50>]



2.5 [Task-2]Getting top 5k features using `feature_importances_`

In [196]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [197]:

```
x_train_set5 = DTC_set2.feature_importances_  
x_train_set5 = x_train_set5[:5000]
```

In [198]:

```
x_train_set5 = x_train_set5.reshape(-1, 1)  
x_train_set5.shape
```

Out[198]:

(5000, 1)

In [209]:

```
x_test_set5 = DTC_set2.feature_importances_  
x_test_set5 = x_test_set5[:5000]  
x_test_set5 = x_test_set5.reshape(-1, 1)  
x_test_set5.shape
```

Out[209]:

(5000, 1)

In [199]:

```
y_train_set5 = y_train[:5000]  
y_train_set5.shape
```

Out[199]:

(5000,)

In [212]:

```
y_test_set5 = y_test[:5000]  
y_test_set5.shape
```

Out[212]:

(5000,)

In []:

In [229]:

```

from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

tuned_parameters = {'max_depth':[1, 5, 10, 50, 100, 500], 'min_samples_split':[5, 10, 50, 100]}
DTC = DecisionTreeClassifier(class_weight='balanced')
clf = GridSearchCV(DTC, tuned_parameters, cv=10, scoring='roc_auc')
clf.fit(x_train_set5, y_train_set5)

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

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

#For min_samples_split
train_auc_min_samples = results['mean_train_score']
train_auc_std_min_samples = results['std_train_score']
cv_auc_min_samples = results['mean_test_score']
cv_auc_std_min_samples = results['std_test_score']
min_samples_split = results['param_min_samples_split']

plt.plot(max_depth, train_auc_max_depth, label='Train AUC max_depth')
# 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(min_samples_split, train_auc_min_samples, label='Train AUC min_samples') #Plot
ting _min_samples-split curve

plt.plot(max_depth, cv_auc_max_depth, 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='da
rkorange')
plt.plot(min_samples_split, cv_auc_min_samples, label='CV AUC min_samples')

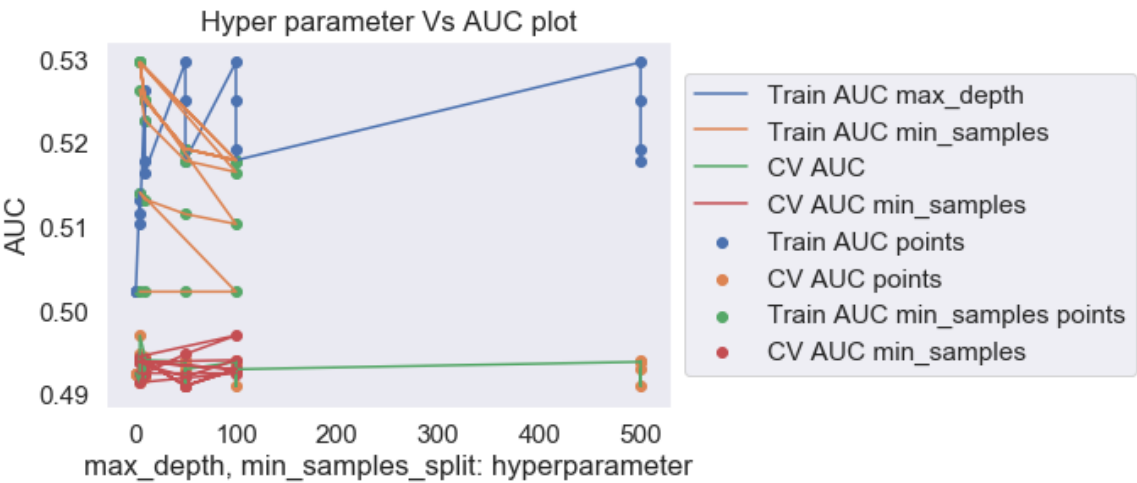
plt.scatter(max_depth, train_auc_max_depth, label='Train AUC points')
plt.scatter(max_depth, cv_auc_max_depth, label='CV AUC points')

plt.scatter(min_samples_split, train_auc_min_samples, label=('Train AUC min_samples poi
nts'))
plt.scatter(min_samples_split, cv_auc_min_samples, label=('CV AUC min_samples'))

plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.xlabel("max_depth, min_samples_split: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()

results.head()

```



Out[229]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	param_n
0	0.006802	0.009814	0.001601	0.001744	1	
1	0.005054	0.001665	0.002901	0.003477	1	
2	0.003520	0.005680	0.002163	0.004663	1	
3	0.000201	0.000602	0.001563	0.004688	1	
4	0.003327	0.006668	0.000150	0.000451	5	

5 rows × 32 columns

In [230]:

```
clf.best_params_
```

Out[230]:

```
{'max_depth': 5, 'min_samples_split': 100}
```

In [231]:

```
from sklearn.metrics import roc_curve, auc

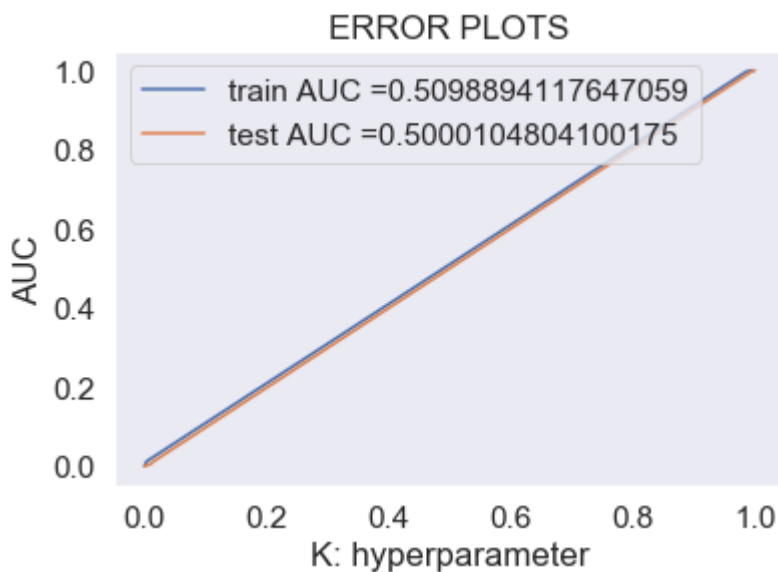
DTC_set5 = DecisionTreeClassifier(max_depth=5, min_samples_split=100, class_weight='balanced')
DTC_set5.fit(x_train_set5, y_train_set5)

y_train_pred = clf.predict_proba(x_train_set5)[:, 1]
y_test_pred = clf.predict_proba(x_test_set5)[:, 1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_set5, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_set5, y_test_pred)

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

plt.show()
```



In [232]:

```
#Confusion matrix
```

```
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train_set5, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test_set5, predict_with_best_t(y_test_pred, best_t)))
```

the maximum value of $tpr \cdot (1 - fpr)$ 0.013826823529411765 for threshold 0.688

Train confusion matrix

```
[[ 747    3]
 [4191   59]]
```

Test confusion matrix

```
[[ 783   11]
 [4155   51]]
```


In [234]:

```
import seaborn as sns

#Train Confusion matrix

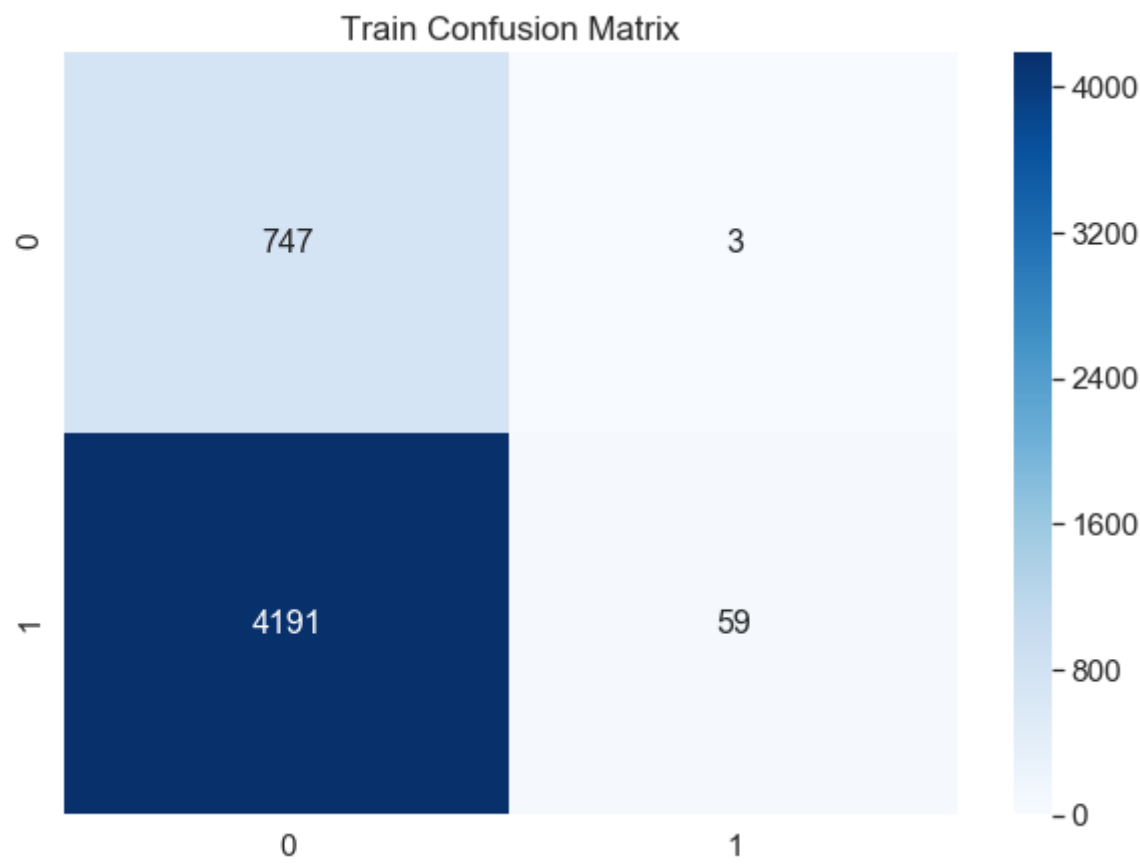
#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

df_cm = pd.DataFrame(confusion_matrix(y_train_set5, predict_with_best_t(y_train_pred, best_t)), columns=np.unique(y_train_set5), index = np.unique(y_train_set5))

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

Out[234]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b5264a9588>



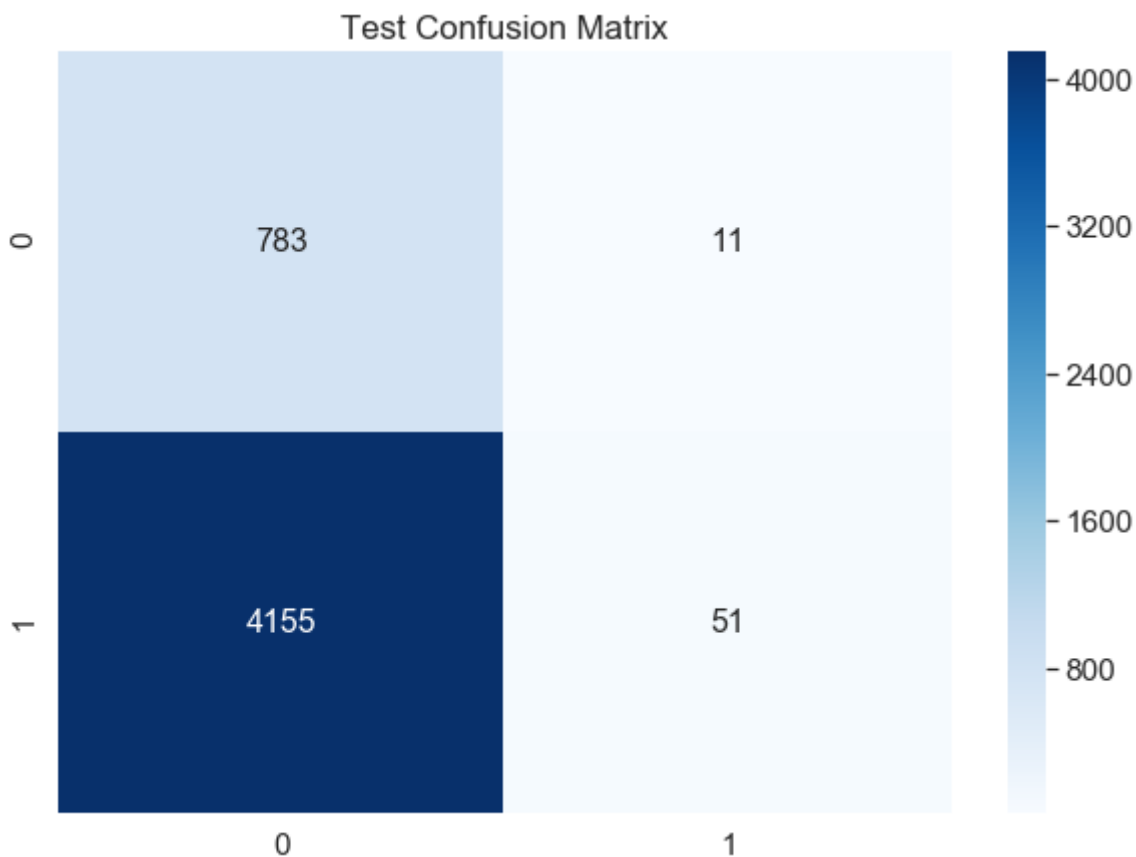
In [236]:

```
#Test Confusion matrix
df_cm = pd.DataFrame(confusion_matrix(y_test_set5, predict_with_best_t(y_test_pred, best_t)), columns=np.unique(y_test_set5), index = np.unique(y_test_set5))

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

Out[236]:

<matplotlib.axes._subplots.AxesSubplot at 0x1b525cf87b8>



In []:

In []:

3. Conclusion

In [0]:

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

In [237]:

```

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ['Vectorizer', 'Model', 'max_depth', 'min_samples_split', 'Test AUC']

x.add_row(['BOW', 'Decision Tree', '10', '100', '0.65'])
x.add_row(['TFIDF', 'Decision Tree', '10', '5', '0.65'])
x.add_row(['AVG W2V', 'Decision Tree', '5', '5', '0.66'])
x.add_row(['TFIDF W2V', 'Decision Tree', '5', '5', '0.66'])
x.add_row(['TFIDF', 'Decision Tree', '5', '100', '0.5'])

print(x)

```

Vectorizer	Model	max_depth	min_samples_split	Test AUC
BOW	Decision Tree	10	100	0.65
TFIDF	Decision Tree	10	5	0.65
AVG W2V	Decision Tree	5	5	0.66
TFIDF W2V	Decision Tree	5	5	0.66
TFIDF	Decision Tree	5	100	0.5

In []: