Decision Tree Algorithm on Donors_Choose dataset

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import sqlite3
import pandas as pd
import numpy as np
import nltk
import gc
gc.enable()
gc.DEBUG_SAVEALL
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os
import math
from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
gc.set_threshold(2, 1, 1)
```

2.1 Loading Input Data

In [2]:
```
# %load_ext memory_profiler
# s=0
# We are taking samples of 0's and 1's and appending them to overcome memory erro
project_data = pd.read_csv('train_data.csv')
# project_data=project_data.dropna(how='any')
# project_data_1 = project_data[project_data['project_is_approved'] == s+1]
# project_data_0 = project_data[project_data['project_is_approved'] == s]
# project_data=project_data.fillna("")
project_data_1=project_data.head(25000)
project_data_0=project_data.tail(25000)
project_data_1=project_data_1.append(project_data_0)
project_data=project_data_1
resource_data = pd.read_csv('resources.csv')

#Sorting them by columns to spread the zeros and one's unevenly in the 'project_

project_data.sort_values(by=['teacher_number_of_previously_posted_projects'])

# project_data.sort_values(by=['project_essay_4'], ascending=False)
project_data_1=None
project_data_0=None
```

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' 'scho
ol_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 resource data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(1)
# project_data.head(2)
```

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

Out[4]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.0

In [5]:
```
y = project_data['project_is_approved'].values
X = project_data.drop(['project_is_approved'], axis=1)
X.head(1)
project_data=None
gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[5]: 32

2.2 Getting the Data Model Ready:Preprocessing and Vectorizing categorical features

2.2.1 Preprocessing:project_grade_category

In [6]:
```
sub_catogories = list(X['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.co

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science",
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trail
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
X['project_grade_category'] = sub_cat_list
```

In [7]:
```
sub_catogories=None
sub_cat_list=None
temp=None
i=None
j=None
catogories=None
cat_list=None
temp=None
my_counter=None
word=None
cat_dict=None
gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[7]: 32

2.2.2 Preprocessing:project_subject_categories

In [8]:
```
catogories = list(X['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science",
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trail
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

Out[8]:

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

2.2.3 Preprocessing:project_subject_subcategories

In [9]:
```
sub_catogories = list(X['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science",
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trail
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [10]:
```
X['clean_subcategories'] = sub_cat_list
X.drop(['project_subject_subcategories'], axis=1, inplace=True)
X.head(2)
```

Out[10]:

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

2.2.4 New Column:digits in summary

In [11]:
```
# Creating a new column 'digits_in_summary' which contains flags of 1 for /
# 'project_resource_summary' containing numeric specification in their requiremnt
project_resource_summary = []
new=[]

project_resource_summary = list(X['project_resource_summary'].values)

for i in project_resource_summary:
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(' '):
        if j.isdigit():
            new.append(1)
            break
        else:
            continue
    else:
        new.append(0)


X['digits_in_summary']=new

X.sort_values(by=['digits_in_summary'])
```

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	p
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
91629	8616	p220748	32c24ca620f89faba3107e4aafcdbf07	Ms.	CA	
91630	9156	p201859	0edc5fc97906c1fb271e1ffe50dfae24	Mrs.	OR	
91631	74015	p167529	701ea8412bb39370abb8ebec7ddd7ea2	Mrs.	NC	

In [12]:
```
#To make best use of the memory we are setting the variable names to 'None' and p
project_resource_summary=None
new=None
new1=None
i=None
j=None
a=None

gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[12]: 32

2.2.5 Preprocessing:Text features (Project Essay's)

In [13]:
```
# merge two column text dataframe:
X["essay"] = X["project_essay_1"].map(str) +\
                        X["project_essay_2"].map(str) + \
                        X["project_essay_3"].map(str) + \
                        X["project_essay_4"].map(str)
```

In [14]:
```
X = X.drop(['project_essay_1', 'project_essay_2','project_essay_3', 'project_essa
X.shape
```

Out[14]: (50000, 14)

2.2.6 Adding column Cost per project in dataset

In [15]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).r
price_data.head(2)
type(price_data)
```

Out[15]: pandas.core.frame.DataFrame

In [16]:
```
# join two dataframes in python:
X = pd.merge(X, price_data, on='id', how='left')
X.sort_values(by=['price'])
X.head(2)
```

Out[16]:

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

In [17]:
```
#To make best use of the memory we are setting the variable names to 'None' and p
resource_data=None
price_data=None
gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[17]: 32

2.2.7 Text Preprocessing:Essay Text

In [18]:
```
# 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 [19]:
```
sent = decontracted(X['essay'].values[99])
print(sent)
print("="*50)
```

```
My preschool students are children who are three to five years of age.  My scho
ol is in sunny San Pedro, California. The children from San Pedro come to schoo
l each morning ready to learn and grow.  There is never a dull moment in our cl
ass; my students are busy bees moving from one interest area to another.  They
are eager to learn, explore, and experiment with the instructional materials an
d centers I set up for them.  We need more materials for the children to engage
with, materials that will foster their interest in technology, literacy, math,
science, art, and engineering. \r\nMy student is will learn number recognition
and develop counting skills while engaging with the Learn to count picture puzz
les and number Sequencing puzzles. While building with the 3-D Magnet Builders
and Crystal Building Blocks, my student is mathematical skills will be supporte
d and strengthened in concepts such as measurement, comparison, number estimati
on, symmetry and balance. My student is will build number skills as the they si
ft and make exciting number shell discoveries with every scoop at the sand tabl
e. The sort a shape activity board will allow my youngest students to learn col
ors, shapes and sorting skills as they fit various shape pieces into place.nann
an
==================================================
```

In [20]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-bre
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My preschool students are children who are three to five years of age.  My scho
ol is in sunny San Pedro, California. The children from San Pedro come to schoo
l each morning ready to learn and grow.  There is never a dull moment in our cl
ass; my students are busy bees moving from one interest area to another.  They
are eager to learn, explore, and experiment with the instructional materials an
d centers I set up for them.  We need more materials for the children to engage
with, materials that will foster their interest in technology, literacy, math,
science, art, and engineering.   My student is will learn number recognition an
d develop counting skills while engaging with the Learn to count picture puzzle
s and number Sequencing puzzles. While building with the 3-D Magnet Builders an
d Crystal Building Blocks, my student is mathematical skills will be supported
and strengthened in concepts such as measurement, comparison, number estimatio
n, symmetry and balance. My student is will build number skills as the they sif
t and make exciting number shell discoveries with every scoop at the sand tabl
e. The sort a shape activity board will allow my youngest students to learn col
ors, shapes and sorting skills as they fit various shape pieces into place.nann
an
```

In [21]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My preschool students are children who are three to five years of age My school
is in sunny San Pedro California The children from San Pedro come to school eac
h morning ready to learn and grow There is never a dull moment in our class my
students are busy bees moving from one interest area to another They are eager
to learn explore and experiment with the instructional materials and centers I
set up for them We need more materials for the children to engage with material
s that will foster their interest in technology literacy math science art and e
ngineering My student is will learn number recognition and develop counting ski
lls while engaging with the Learn to count picture puzzles and number Sequencin
g puzzles While building with the 3 D Magnet Builders and Crystal Building Bloc
ks my student is mathematical skills will be supported and strengthened in conc
epts such as measurement comparison number estimation symmetry and balance My s
tudent is will build number skills as the they sift and make exciting number sh
ell discoveries with every scoop at the sand table The sort a shape activity bo
ard will allow my youngest students to learn colors shapes and sorting skills a
s they fit various shape pieces into place nannan
```

In [22]:
```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itse
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 't
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than'
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should'v
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "d
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn
            'won', "won't", 'wouldn', "wouldn't"]
```

In [23]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(X['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████| 50000/50000 [01:39<00:00, 500.76it/s]
```

In [24]:
```
# after preprocesing

# X['essay'] = None
X['essay'] = preprocessed_essays

X.head(2)
```

Out[24]:

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

In [25]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(X['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_project_title.append(sent.lower().strip())
```

100%|██████████| 50000/50000 [00:04<00:00, 10262.14it/s]

In [26]:
```
preprocessed_project_title[4999]
# after preprocesing

# X['project_title'] = None
X['project_title'] = preprocessed_project_title

X.head(2)
```

Out[26]:

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

2.2.8 Splitting the data into Train and Test

In [27]:
```
# train test split(67:33)
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, strati
# X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.
```

In [28]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X_train['clean_categories'].values:
    my_counter.update(word.split())
print(my_counter)
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict_train = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

```
Counter({'Literacy_Language': 16076, 'Math_Science': 12619, 'Health_Sports': 43
84, 'SpecialNeeds': 4165, 'AppliedLearning': 3769, 'Music_Arts': 3165, 'History
_Civics': 1785, 'Warmth': 410, 'Care_Hunger': 410})
```

2.2.9 Vectorizing Categorical data: clean_categories(Project subject categories)

In [29]:
```
print(X_train.shape, y_train.shape)
# print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

Bow_features_names1=[]

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict_train.keys()), lowe
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on t

# we use the fitted Countvectorizer to convert the text to vector
X_train_clean_cat_ohe = vectorizer.transform(X_train['clean_categories'].values)
# X_cv_clean_cat_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_clean_cat_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_clean_cat_ohe.shape, y_train.shape)
# print(X_cv_clean_cat_ohe.shape, y_cv.shape)
print(X_test_clean_cat_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(33500, 16) (33500,)
(16500, 16) (16500,)
====================================================================================
====================
After vectorizations
(33500, 9) (33500,)
(16500, 9) (16500,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'S
pecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
====================================================================================
====================
```

2.2.10 Vectorizing Categorical data: clean_subcategories(Project subject subcategories)

In [30]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X_train['clean_subcategories'].values:
    my_counter.update(word.split())
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict_train = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1
```

In [31]:
```
print(X_train.shape, y_train.shape)
# print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict_train.keys()), 
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only o

# we use the fitted Countvectorizer to convert the text to vector
X_train_clean_sub_cat_ohe = vectorizer.transform(X_train['clean_subcategories'].
# X_cv_clean_sub_cat_ohe = vectorizer.transform(X_cv['clean_subcategories'].value
X_test_clean_sub_cat_ohe = vectorizer.transform(X_test['clean_subcategories'].va

print("After vectorizations")
print(X_train_clean_sub_cat_ohe.shape, y_train.shape)
# print(X_cv_clean_sub_cat_ohe.shape, y_cv.shape)
print(X_test_clean_sub_cat_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(33500, 16) (33500,)
(16500, 16) (16500,)
====================================================================================
====================
After vectorizations
(33500, 30) (33500,)
(16500, 30) (16500,)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Ex
tracurricular', 'Civics_Government', 'ForeignLanguages', 'Warmth', 'Care_Hunge
r', 'NutritionEducation', 'SocialSciences', 'PerformingArts', 'CharacterEducati
on', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography',
'Health_LifeScience', 'EarlyDevelopment', 'Gym_Fitness', 'ESL', 'EnvironmentalS
cience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'L
iterature_Writing', 'Mathematics', 'Literacy']
====================================================================================
====================
```

2.2.11 Vectorizing Categorical data: school_state

In [32]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X_train['school_state'].values:
    my_counter.update(word.split())
school_state_dict = dict(my_counter)
sorted_school_state_dict_train = dict(sorted(school_state_dict.items(), key=lambd
# sorted_school_state_dict
```

In [33]:
```
print(X_train.shape, y_train.shape)
# print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)


vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict_train.keys
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train

# we use the fitted Countvectorizer to convert the text to vector
X_train_School_state_ohe = vectorizer.transform(X_train['school_state'].values)
# X_cv_School_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_School_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_School_state_ohe.shape, y_train.shape)
# print(X_cv_School_state_ohe.shape, y_cv.shape)
print(X_test_School_state_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(33500, 16) (33500,)
(16500, 16) (16500,)
====================================================================================
======================
After vectorizations
(33500, 51) (33500,)
(16500, 51) (16500,)
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'AK', 'NH', 'DE', 'ME', 'WV', 'NM',
'DC', 'HI', 'ID', 'IA', 'KS', 'AR', 'CO', 'MN', 'KY', 'OR', 'MS', 'NV', 'MD',
'CT', 'WI', 'UT', 'AL', 'TN', 'VA', 'AZ', 'OK', 'NJ', 'MA', 'WA', 'LA', 'OH',
'MO', 'IN', 'PA', 'MI', 'GA', 'SC', 'IL', 'NC', 'FL', 'NY', 'TX', 'CA']
====================================================================================
======================
```

2.2.12 Vectorizing Categorical data: project_grade_category

In [34]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X_train['project_grade_category'].values:
    my_counter.update(word.split())
project_grade_dict = dict(my_counter)
sorted_project_grade_dict_train = dict(sorted(project_grade_dict.items(), key=lar
```

In [35]:
```
print(X_train.shape, y_train.shape)
# print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vectorizer= CountVectorizer(vocabulary=list(sorted_project_grade_dict_train.keys
vectorizer.fit(X_train['project_grade_category'].values) # fit has to happen only

# we use the fitted Countvectorizer_pro_gradeto convert the text to vector
X_train_project_grade_category_ohe = vectorizer.transform(X_train['project_grade_
# X_cv_project_grade_category_ohe = vectorizer.transform(X_cv['project_grade_cate
X_test_project_grade_category_ohe = vectorizer.transform(X_test['project_grade_ca


print("After vectorizations")
print(X_train_project_grade_category_ohe.shape, y_train.shape)
# print(X_cv_project_grade_category_ohe.shape, y_cv.shape)
print(X_test_project_grade_category_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(33500, 16) (33500,)
(16500, 16) (16500,)
====================================================================================
====================
After vectorizations
(33500, 4) (33500,)
(16500, 4) (16500,)
['Grades9-12', 'Grades6-8', 'Grades3-5', 'GradesPreK-2']
====================================================================================
====================
```

2.2.13 Vectorizing Categorical data: teacher_prefix

In [36]:
```
#To overcome the blanks in the teacher_prefix categry the .fillna is used
X_train['teacher_prefix']=X_train['teacher_prefix'].fillna("")
# project_data1=project_data.dropna()
```

In [37]:
```
#To overcome the blanks in the teacher_prefix categry the .fillna is used
X_test['teacher_prefix']=X_test['teacher_prefix'].fillna("")
# project_data1=project_data.dropna()
```

In [38]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
my_counter1=[]
# project_data['teacher_prefix']=str(project_data['teacher_prefix'])
for word in X_train['teacher_prefix'].values:
    my_counter.update(word.split())
teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict_train = dict(sorted(teacher_prefix_dict.items(), key=l
# teacher_prefix_dict
```

In [39]:
```
print(X_train.shape, y_train.shape)
# print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict_train.key
vectorizer.fit(X_train['teacher_prefix'].values) # fit has to happen only on tra

# we use the fitted Countvectorizer to convert the text to vector
X_train_teacher_prefix_ohe = vectorizer.transform(X_train['teacher_prefix'].valu
# X_cv_teacher_prefix_ohe = vectorizer.transform(X_cv['teacher_prefix'].values)
X_test_teacher_prefix_ohe = vectorizer.transform(X_test['teacher_prefix'].values

print("After vectorizations")
print(X_train_teacher_prefix_ohe.shape, y_train.shape)
# print(X_cv_teacher_prefix_ohe.shape, y_cv.shape)
print(X_test_teacher_prefix_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(33500, 16) (33500,)
(16500, 16) (16500,)
====================================================================================
====================
After vectorizations
(33500, 5) (33500,)
(16500, 5) (16500,)
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
====================================================================================
====================
```

2.3 Make Data Model Ready: Vectorizing Numerical features

2.3.1 Vectorizing Numerical features--Price

In [40]:
```
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import StandardScaler
normalizer = StandardScaler() #Normalizer()
# normalizer_test = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(-1,1))
# normalizer_test.fit(X_test['price'].values.reshape(1,-1))

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


# X_train_price_norm=np.reshape(X_train_price_norm,(1,-1))
# X_test_price_norm=np.reshape(X_test_price_norm,(1,-1))

print("After vectorizations")

# np.reshape(X_train_price_norm,
print(X_train_price_norm.shape, y_train.shape)
# print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

2.3.2 Vectorizing Numerical features--teacher_number_of_previously_posted_projects

In [41]:
```
from sklearn.preprocessing import Normalizer
normalizer_train = StandardScaler() #Normalizer()
normalizer_test = StandardScaler() #Normalizer()
# normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.re

X_train_teacher_number_of_previously_posted_projects_norm = normalizer.transform
# X_cv_teacher_number_of_previously_posted_projects_norm = normalizer.transform(
X_test_teacher_number_of_previously_posted_projects_norm = normalizer.transform(



# X_train_teacher_number_of_previously_posted_projects_norm=np.reshape(X_train_te
# X_test_teacher_number_of_previously_posted_projects_norm=np.reshape(X_test_tea


print("After vectorizations")
print(X_train_teacher_number_of_previously_posted_projects_norm.shape, y_train.sh
# print(X_cv_teacher_number_of_previously_posted_projects_norm.shape, y_cv.shape)
print(X_test_teacher_number_of_previously_posted_projects_norm.shape, y_test.shap
print("="*100)
```

```
After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

2.3.3 Vectorizing Numerical features--digits_in_summary

In [42]:
```
X_train['digits_in_summary'].fillna(X_train['digits_in_summary'].mean(), inplace=
# X_cv['digits_in_summary'].fillna(X_cv['digits_in_summary'].mean(), inplace=True
X_test['digits_in_summary'].fillna(X_test['digits_in_summary'].mean(), inplace=Tr
```

In [43]:
```
from sklearn.preprocessing import Normalizer
normalizer_train = StandardScaler() #Normalizer()
normalizer_test = StandardScaler() #Normalizer()
# normalizer.fit(X_train['digits_in_summary'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['digits_in_summary'].values.reshape(-1,1))

X_train_digits_in_summary_norm = normalizer.transform(X_train['digits_in_summary
# X_cv_digits_in_summary_norm = normalizer.transform(X_cv['digits_in_summary'].va
X_test_digits_in_summary_norm = normalizer.transform(X_test['digits_in_summary']


# X_train_digits_in_summary_norm=np.reshape(X_train_digits_in_summary_norm,(1,-1)
# X_test_digits_in_summary_norm=np.reshape(X_test_digits_in_summary_norm,(1,-1))

print("After vectorizations")
print(X_train_digits_in_summary_norm.shape, y_train.shape)
# print(X_cv_digits_in_summary_norm.shape, y_cv.shape)
print(X_test_digits_in_summary_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================
====================
```

2.6 Make Data Model Ready: Vectorizing Essay and Project_title feature into BOW & TFIDF

Vectorizing Text data

2.6 TFIDF vectorizer

2.6.1 TFIDF vectorizer:Essay

In [44]:
```
print(X_train.shape, y_train.shape)
# print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
print("="*100)
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(2,2), max_features=5000)
vectorizer.fit(X_train['essay'].values) # fit has to happen only on train data
# we use the fitted Countvectorizer to convert the text to vector
X_train_essay_Tfidf = vectorizer.transform(X_train['essay'].values)
# X_cv_essay_Tfidf = vectorizer.transform(X_cv['essay'].values)
X_test_essay_Tfidf = vectorizer.transform(X_test['essay'].values)
print("After vectorizations")
print(X_train_essay_Tfidf.shape, y_train.shape)
# print(X_cv_essay_Tfidf.shape, y_cv.shape)
print(X_test_essay_Tfidf.shape, y_test.shape)
print("="*100)
```

```
(33500, 16) (33500,)
(16500, 16) (16500,)
====================================================================================================
After vectorizations
(33500, 5000) (33500,)
(16500, 5000) (16500,)
====================================================================================================
```

2.6.2 TFIDF vectorizer:Project Title

In [45]:
```
# print(X_train.shape, y_train.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)

# We are considering only the words which appeared in at least 10 documents(rows
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(2,2), max_features=5000)
vectorizer.fit(X_train['project_title'].values) # fit has to happen only on trai

# we use the fitted Countvectorizer to convert the text to vector
X_train_project_title_tfidf = vectorizer.transform(X_train['project_title'].valu
# X_cv_project_title_tfidf = vectorizer.transform(X_cv['project_title'].values)
X_test_project_title_tfidf = vectorizer.transform(X_test['project_title'].values

print("After vectorizations")
print(X_train_project_title_tfidf.shape, y_train.shape)
# print(X_cv_project_title_tfidf.shape, y_cv.shape)
print(X_test_project_title_tfidf.shape, y_test.shape)
print("="*100)
```

```
====================================================================================
====================
After vectorizations
(33500, 1054) (33500,)
(16500, 1054) (16500,)
====================================================================================
====================
```

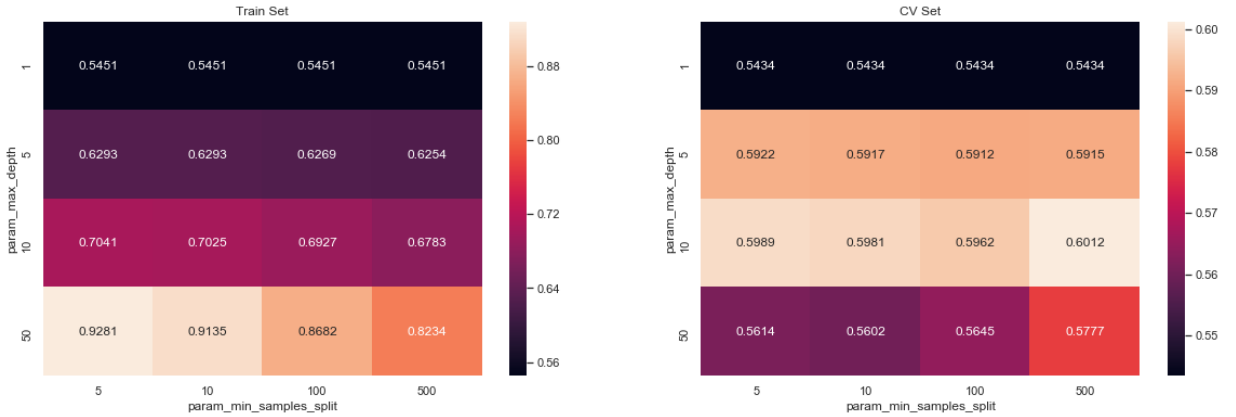
In [46]:
```
from scipy.sparse import hstack
X_Tfidf_train = hstack(( X_train_project_title_tfidf,X_train_essay_Tfidf,X_train_
X_Tfidf_train=X_Tfidf_train.todense()
X_Tfidf_train=np.array(X_Tfidf_train)

# X_Tfidf_cv = hstack(( X_cv_project_title_tfidf,X_cv_essay_Tfidf,X_cv_digits_in_
# X_Tfidf_cv=X_Tfidf_cv.todense()
# X_Tfidf_cv=np.array(X_Tfidf_cv)

X_Tfidf_test = hstack(( X_test_project_title_tfidf,X_test_essay_Tfidf,X_test_digi
X_Tfidf_test=X_Tfidf_test.todense()
X_Tfidf_test=np.array(X_Tfidf_test)
```

2.6.3 Applying Decision Tree on `TFIDF, SET 1`

Applying Decision Tree & GridSearchCV on Train data to obtain the best `C`

In [47]:
```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
# from sklearn.multioutput import MultiOutputClassifier
# from sklearn.datasets import make_multilabel_classification
from sklearn.tree import DecisionTreeClassifier


dt1 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 100,500]}
clf1 = GridSearchCV(dt1, parameters, cv=2, scoring='roc_auc',return_train_score=T
se1 = clf1.fit(X_Tfidf_train, y_train)


# train_auc = model.cv_results_['mean_train_score']
# cv_auc= model.cv_results_['mean_test_score']
# plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
# plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
# plt.legend()
# plt.xlabel("log")
# plt.xscale('log')
# plt.ylabel("ROC_AUC score")
# plt.title("ROC_AUC vs log plot")
# plt.grid()
# plt.show()
# model.fit(X_Tfidf_train, y_train)

# print(model.best_estimator_)
# print(model.score(X_Tfidf_test, y_test))
```

In [48]:
```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(clf1.cv_results_).groupby(['param_max_depth','param_m
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



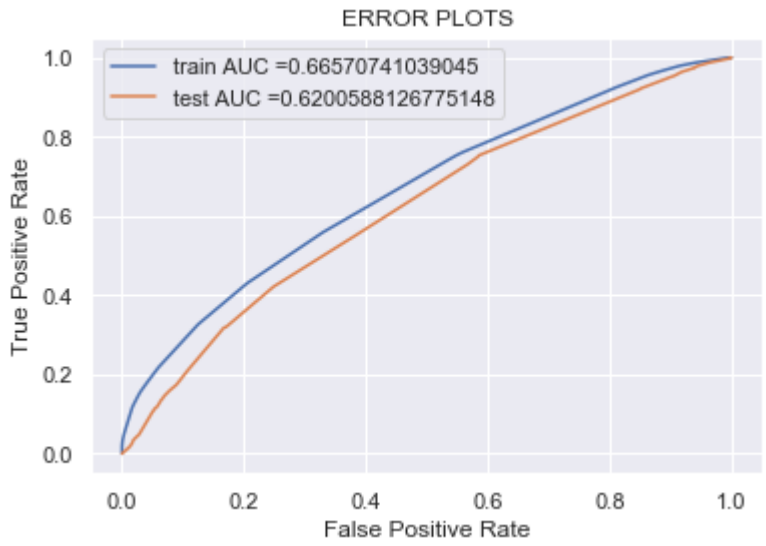
In [50]:
```
print(clf1.best_estimator_)
#Mean cross-validated score of the best_estimator
print(clf1.score(X_Tfidf_train,y_train))
print(clf1.score(X_Tfidf_test,y_test))
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight='balanced', criterion='gin
i',
                       max_depth=10, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
0.66570741039045
0.6200588126775148
```

In [51]:
```
best_tune_parameters=[{'max_depth':[10], 'min_samples_split':[500] } ]
```

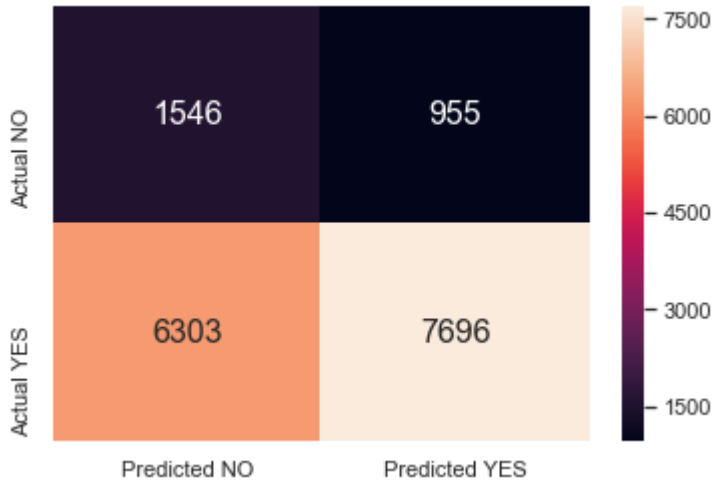
2.6.4 Receiver Operating Characteristic- (TFIDF)

In [65]:
```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, auc
clf11= DecisionTreeClassifier(class_weight = 'balanced',max_depth=10,min_samples_
# clfV1=DecisionTreeClassifier (class_weight = 'balanced',max_depth=10,min_sample
clf11.fit(X_Tfidf_train, y_train)
# for visulation
# clfV1.fit(X_Tfidf_train, y_train)
#https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDClassi
y_train_pred1 = clf11.predict_proba(X_Tfidf_train) [:,1]
y_test_pred1 = clf11.predict_proba(X_Tfidf_test) [:,1]
train_fpr1, train_tpr1, tr_thresholds1 = roc_curve(y_train, y_train_pred1)
test_fpr1, test_tpr1, te_thresholds1 = roc_curve(y_test, y_test_pred1)
plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr1, train_t
plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr1, test_tpr1))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



2.6.5 Confusion matrix- TFIDF

In [66]:
```
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#function to get heatmap confusion matrix
def get_confusion_matrix(clf,X_te,y_test):
    y_pred = clf.predict(X_te)
    df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred), index =['Actual NO','A

    sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')

# %%time
get_confusion_matrix(clf11,X_Tfidf_test,y_test)
```

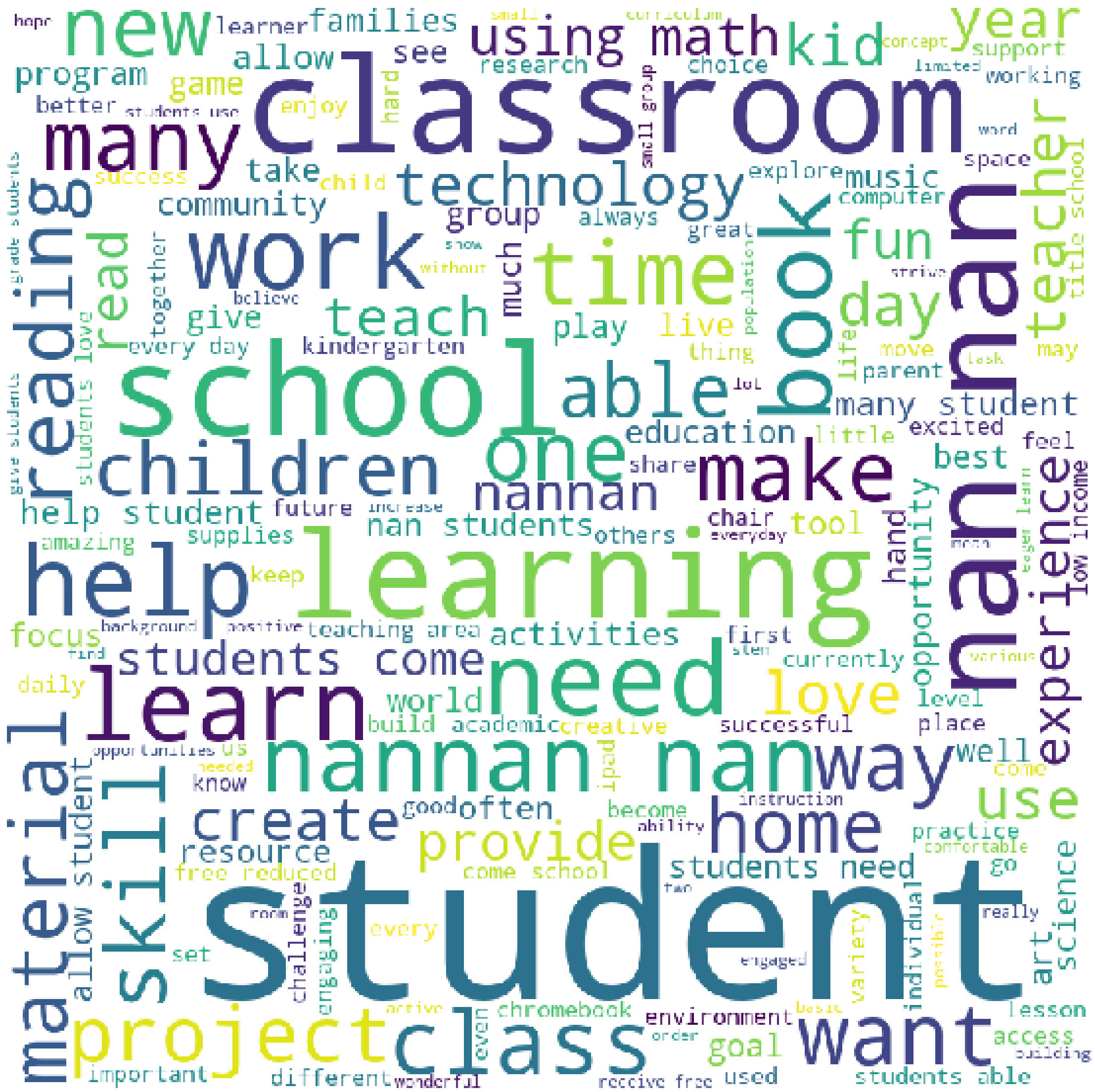


2.7.1 Selecting the FPR data points from Essay,Price and Teacher_number_of_previously_posted_projects test data set.

In [68]:
```
import numpy as np
import pandas as pd

y_pred = clf11.predict(X_Tfidf_test)
#Essay
df=X_test['essay']
df.to_frame()
#Price
df_=X_test['price']
df_.to_frame()
#Teacher_number_of_previously_posted_projects
df_1=X_test['teacher_number_of_previously_posted_projects']
df_1.to_frame()

df1=pd.DataFrame(y_test,columns=['y_test'])
df2=pd.DataFrame(y_pred,columns=['y_pred'])

df_row = pd.concat([df,df1, df2],axis=1)
df_row1 = pd.concat([df_,df1, df2],axis=1)
df_row2 = pd.concat([df_1,df1, df2],axis=1)

#Selecting the false positive data points
df_row = df_row[(df_row.y_pred == 1) & (df_row.y_test == 0) ]
df_row1 = df_row1[(df_row1.y_pred == 1) & (df_row1.y_test == 0) ]
df_row2 = df_row2[(df_row2.y_pred == 1) & (df_row2.y_test == 0) ]
df_row2=df_row2.fillna(0)
```

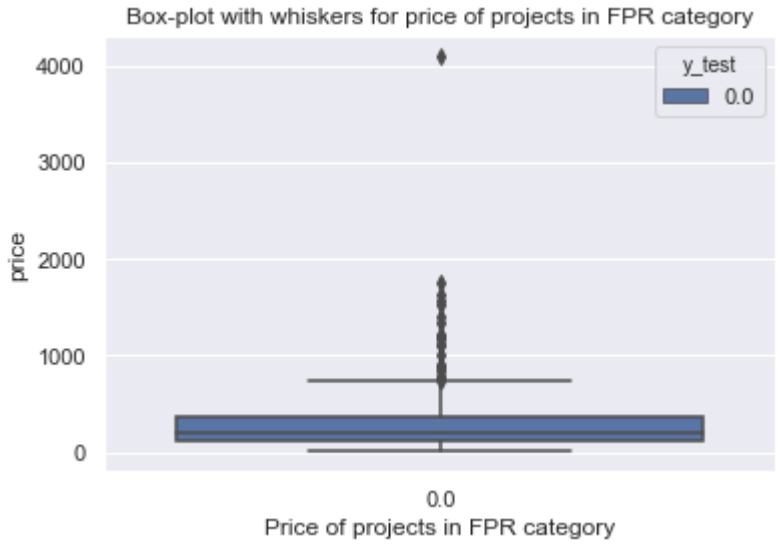
2.7.2 WordCloud of False positive data points- (TFIDF)

In [69]:
```
# https://www.geeksforgeeks.org/generating-word-cloud-python/

from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

# Reads 'Youtube04-Eminem.csv' file
# df = pd.read_csv(r"Youtube04-Eminem.csv", encoding ="latin-1")

comment_words = ' '
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in df_row['essay']:

    # 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()
```



2.7.3 Box plot of price of False positive data points

In [70]:
```
import seaborn as sns
import matplotlib.pyplot as plt
g=sns.boxplot(x='y_test',y='price',hue='y_test',data=df_row1)
plt.title("Box-plot with whiskers for price of projects in FPR category")
plt.xlabel("Price of projects in FPR category")
# plt.legend(loc=1)
plt.show()
```

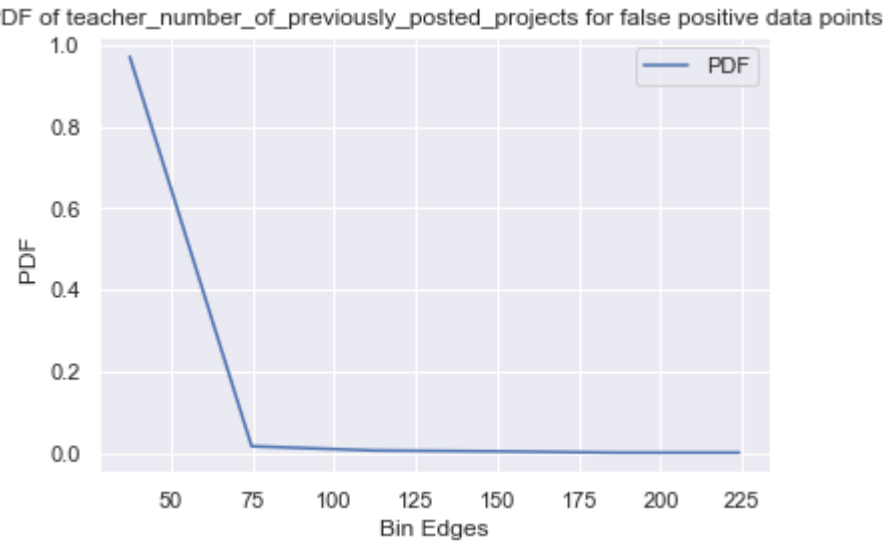


2.7.4 PDF of teacher_number_of_previously_posted_projects for false positive data points.

In [71]:
```
counts, bin_edges = np.histogram(df_row2['teacher_number_of_previously_posted_pro
                                 density = True)
pdf = counts/(sum(counts))
print("The pdf is :",pdf);
print("The bin edges of pdf are:",bin_edges);
plt.plot(bin_edges[1:],pdf,label='PDF');
plt.title('PDF of teacher_number_of_previously_posted_projects for false positive
plt.xlabel("Bin Edges")
plt.ylabel("PDF ")
plt.legend()
```

```
The pdf is : [0.97068063 0.01675393 0.00628272 0.00418848 0.00104712 0.0010471
2]
The bin edges of pdf are: [  0.          37.33333333  74.66666667 112.
149.33333333
 186.66666667 224.        ]
```

Out[71]:
```
<matplotlib.legend.Legend at 0x3229cb70>
```



In [72]:
```
# print("PDF1 of Patients who survived falling in >=5years from the graph-Blue l
print("The probability of finding project quote less than",round(bin_edges[1],2)
print("The probability of finding project quote less than ",round(bin_edges[2],2
print("The probability of finding project quote less than ",round(bin_edges[3],2
print("The probability of finding project quote less than ",round(bin_edges[4],2
print("The probability of finding project quote less than ",round(bin_edges[5],2
print("The probability of finding project quote less than ",round(bin_edges[6],2
```

```
The probability of finding project quote less than 37.33 that got approved and
falling in FPR category is 97.07 %
The probability of finding project quote less than  74.67 that got approved and
falling in FPR category is 1.68 %
The probability of finding project quote less than  112.0 that got approved and
falling in FPR category is 0.63 %
The probability of finding project quote less than  149.33 that got approved an
d falling in FPR category is 0.42 %
The probability of finding project quote less than  186.67 that got approved an
d falling in FPR category is 0.1 %
The probability of finding project quote less than  224.0 that got approved and
falling in FPR category is 0.1 %
```

2.8.1 Getting top features using `feature_importances_` from TFIDF model

In [76]:
```
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression

selector = SelectFromModel(estimator=DecisionTreeClassifier(max_depth=None,class_
# selector.estimator_.coef_
selector.threshold_
selector.get_support()
```

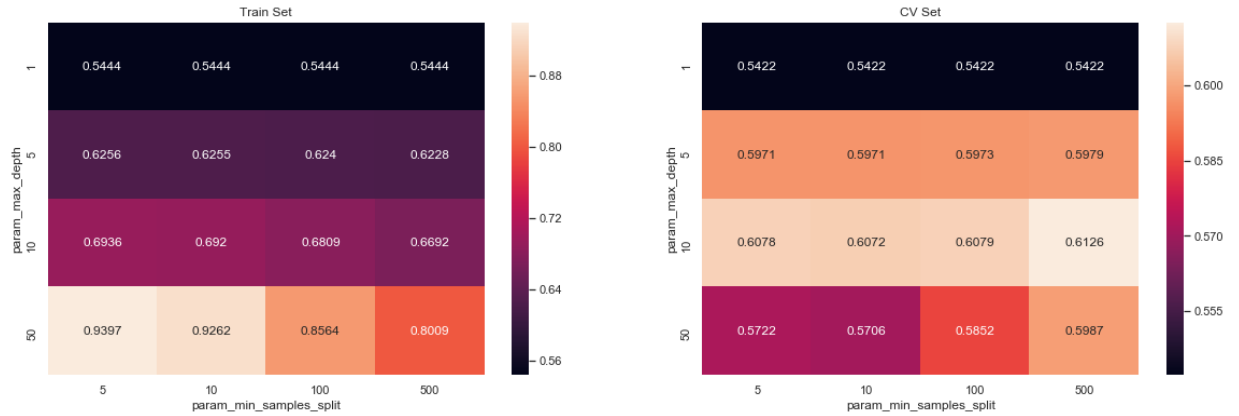
Out[76]:
```
array([False, False, False, ...,  True,  True,  True])
```

In [77]:
```
x=selector.transform(X_Tfidf_train)
x1=selector.transform(X_Tfidf_test)
# X_Tfidf_train.shape
# X_Tfidf_test.shape
```

2.8.2 Applying Decision Tree & GridSearchCV on Train data with best selected features to obtain the best `max_depth` , `min_samples_split`

In [61]:
```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dt4= DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50],'min_samples_split': [5, 10, 100,500]}
clf4 = GridSearchCV(dt4, parameters, cv=5, scoring='roc_auc',return_train_score=
set4= clf4.fit(x, y_train)
```

In [62]:
```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(clf4.cv_results_).groupby(['param_max_depth','param_m
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [63]:
```
#Best Estimator and Best tune parameters
print(clf4.best_estimator_)
#Mean cross-validated score of the best_estimator
print(clf4.score(x,y_train))
print(clf4.score(x1,y_test))
```

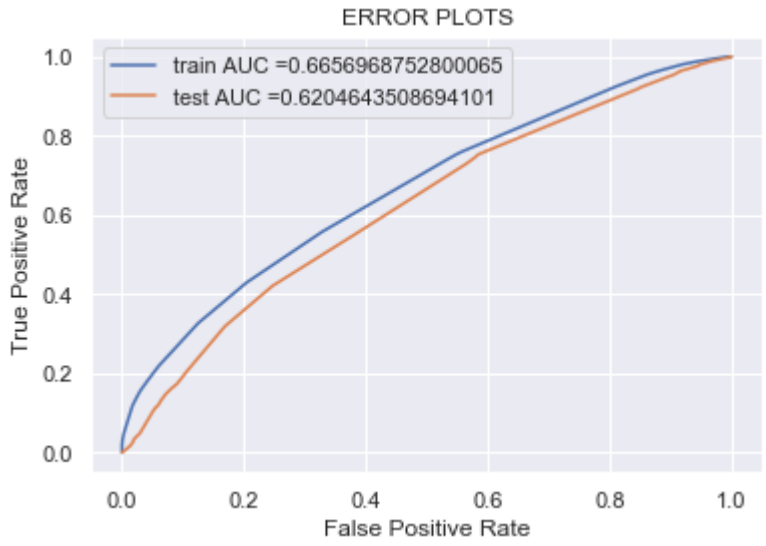
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight='balanced', criterion='gin
i',
                       max_depth=10, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
0.66570741039045
0.6200588126775148
```

In [ ]:
```
best_tune_parameters=[{'max_depth':[10], 'min_samples_split':[500] } ]
```

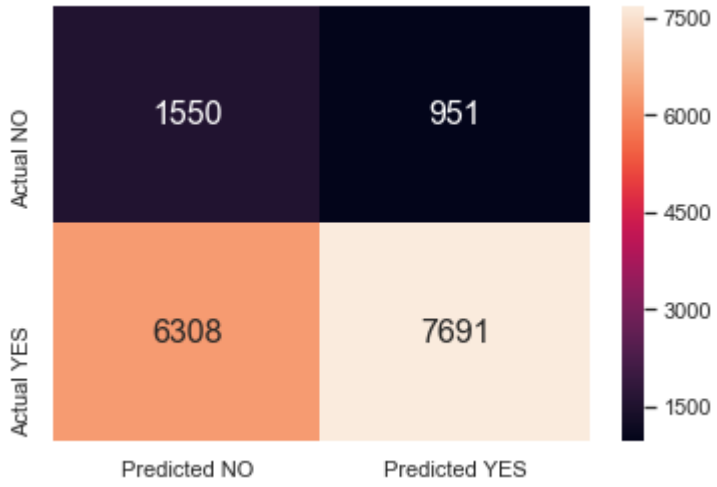
2.8.3 Receiver Operating Characteristic- (TFIDF)

In [81]:

```
#Fitting Model to Hyper-Parameter Curve
from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, auc
clf11= DecisionTreeClassifier(class_weight = 'balanced',max_depth=10,min_samples_
# clfV1=DecisionTreeClassifier (class_weight = 'balanced',max_depth=5,min_samples
clf11.fit(x, y_train)
# for visulation
# clfV1.fit(X_tfidf_w2v_train, y_train)
#https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDClassi
y_train_pred1 = clf11.predict_proba(x) [:,1]
y_test_pred1 = clf11.predict_proba(x1) [:,1]
train_fpr1, train_tpr1, tr_thresholds1 = roc_curve(y_train, y_train_pred1)
test_fpr1, test_tpr1, te_thresholds1 = roc_curve(y_test, y_test_pred1)
plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr1, train_t
plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr1, test_tpr1))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



In [82]:
```
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#function to get heatmap confusion matrix
def get_confusion_matrix(clf,X_te,y_test):
    y_pred = clf.predict(X_te)
    df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred), index =['Actual NO','A
    sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')

# %%time
get_confusion_matrix(clf11,x1,y_test)
```



In [84]:
```
#To make best use of the memory we are setting the variable names to 'None' and p
X_Tfidf_test=None
X_Tfidf_train=None
gc.collect()
```

Out[84]: 0

2.9 TFIDF weighted W2V-Essay

2.9.1 Using Pretrained Models: TFIDF weighted W2V-Essay

In [85]:
```
# train test split
from sklearn.model_selection import train_test_split

Xtfidf_w2v_vectors_train_, Xtfidf_w2v_vectors_test_, y_train, y_test = train_tes

# Xtfidf_w2v_vectors_train, Xtfidf_w2v_vectors_cv, y_train, y_cv = train_test_sp
```

In [86]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(Xtfidf_w2v_vectors_train_)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [87]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-t
# make sure you have the glove_vectors file
import pickle
with open('glove_vectors', 'rb') as f:
    model_ = pickle.load(f)
    glove_words =  set(model_.keys())
```

In [88]:
```
# average Word2Vec
# compute average word2vec for each review.

tfidf_w2v_vectors_essay_train = []; # the avg-w2v for each sentence/review is st
for sentence in tqdm(Xtfidf_w2v_vectors_train_): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model_[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf val
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_train.append(vector)

print(len(tfidf_w2v_vectors_essay_train))
print(len(tfidf_w2v_vectors_essay_train[0]))
```

```
100%|██████████████████████████████████| 33500/33500 [05:20<00:00, 104.46it/s]

33500
300
```

In [89]:
```
# average Word2Vec
# compute average word2vec for each review.

tfidf_w2v_vectors_essay_test = []; # the avg-w2v for each sentence/review is sto
for sentence in tqdm(Xtfidf_w2v_vectors_test_): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model_[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf valu
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_test.append(vector)

print(len(tfidf_w2v_vectors_essay_test))
print(len(tfidf_w2v_vectors_essay_test[0]))
```

```
100%|████████████████████████████████████| 16500/16500 [02:57<00:00, 92.76it/s]

16500
300
```

In [90]:
```
import scipy
tfidf_w2v_vectors_essay_train=scipy.sparse.csr_matrix(tfidf_w2v_vectors_essay_tra
type(tfidf_w2v_vectors_essay_train)

tfidf_w2v_vectors_essay_test=scipy.sparse.csr_matrix(tfidf_w2v_vectors_essay_test
type(tfidf_w2v_vectors_essay_test)

# Xtfidf_w2v_vectors_cv=scipy.sparse.csr_matrix(Xtfidf_w2v_vectors_cv)
# type(Xtfidf_w2v_vectors_cv)
```

Out[90]:
```
scipy.sparse.csr.csr_matrix
```

2.9.2 Using Pretrained Models: TFIDF weighted W2V on project_title

In [91]:
```
# train test split
from sklearn.model_selection import train_test_split

tfidf_w2v_vectors_Pro_title_train_, tfidf_w2v_vectors_Pro_title_test_, y_train, y

# tfidf_w2v_vectors_Pro_title_train, tfidf_w2v_vectors_Pro_title_cv, y_train, y_c
```

In [92]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(tfidf_w2v_vectors_Pro_title_train_)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [93]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_Pro_title_train = []; # the avg-w2v for each sentence/review is
for sentence in tqdm(tfidf_w2v_vectors_Pro_title_train_): # for each review/sente
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model_[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf valu
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_Pro_title_train.append(vector)

print(len(tfidf_w2v_vectors_Pro_title_train))
print(len(tfidf_w2v_vectors_Pro_title_train[0]))
```

```
100%|██████████████████████████████████| 33500/33500 [00:07<00:00, 4204.60it/s]

33500
300
```

In [94]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_Pro_title_test = []; # the avg-w2v for each sentence/review is
for sentence in tqdm(tfidf_w2v_vectors_Pro_title_test_): # for each review/sente
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model_[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf valu
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_Pro_title_test.append(vector)

print(len(tfidf_w2v_vectors_Pro_title_test))
print(len(tfidf_w2v_vectors_Pro_title_test[0]))
```

```
100%|██████████████████████████████████| 16500/16500 [00:03<00:00, 4892.35it/s]

16500
300
```

In [95]:
```
import scipy
tfidf_w2v_vectors_Pro_title_train=scipy.sparse.csr_matrix(tfidf_w2v_vectors_Pro_t
type(tfidf_w2v_vectors_Pro_title_train)

tfidf_w2v_vectors_Pro_title_test=scipy.sparse.csr_matrix(tfidf_w2v_vectors_Pro_ti
type(tfidf_w2v_vectors_Pro_title_test)

# tfidf_w2v_vectors_Pro_title_cv=scipy.sparse.csr_matrix(tfidf_w2v_vectors_Pro_t
# type(tfidf_w2v_vectors_Pro_title_cv)
```

Out[95]:
```
scipy.sparse.csr.csr_matrix
```

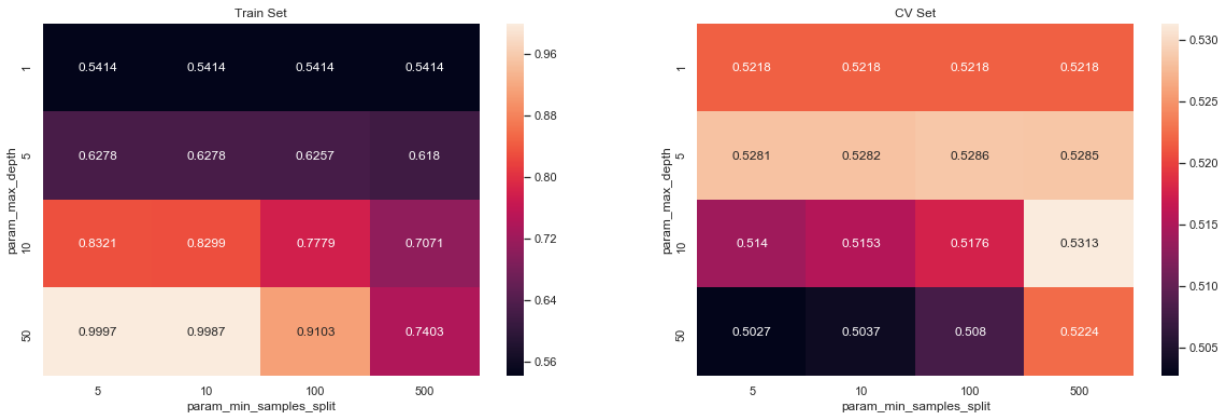
In [96]:
```
X_tfidf_w2v_train = hstack((tfidf_w2v_vectors_essay_train,tfidf_w2v_vectors_Pro_t
X_tfidf_w2v_train=X_tfidf_w2v_train.todense()
X_tfidf_w2v_train=np.array(X_tfidf_w2v_train)

# X_tfidf_w2v_cv = hstack((Xtfidf_w2v_vectors_cv,tfidf_w2v_vectors_Pro_title_cv ,
# X_tfidf_w2v_cv=X_tfidf_w2v_cv.todense()
# X_tfidf_w2v_cv=np.array(X_tfidf_w2v_cv)

X_tfidf_w2v_test = hstack((tfidf_w2v_vectors_essay_test,tfidf_w2v_vectors_Pro_ti
X_tfidf_w2v_test=X_tfidf_w2v_test.todense()
X_tfidf_w2v_test=np.array(X_tfidf_w2v_test)
# X_All = hstack((categories_one_hot,sub_categories_one_hot,school_state_one_hot,
```

2.9.3 Applying Decision Tree on `TFIDF W2V`, `SET 2`

Applying Decision Tree & GridSearchCV on Train data to obtain the best `max_depth`, `min_samples_split`

In [97]:
```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dt4= DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 100,500]}
clf4 = GridSearchCV(dt4, parameters, cv=2, scoring='roc_auc',return_train_score=T
set4= clf4.fit(X_tfidf_w2v_train, y_train)
```

In [98]:
```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(clf4.cv_results_).groupby(['param_max_depth','param_m
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [99]:
```
#Best Estimator and Best tune parameters
print(clf4.best_estimator_)
#Mean cross-validated score of the best_estimator
print(clf4.score(X_tfidf_w2v_train,y_train))
print(clf4.score(X_tfidf_w2v_test,y_test))
```

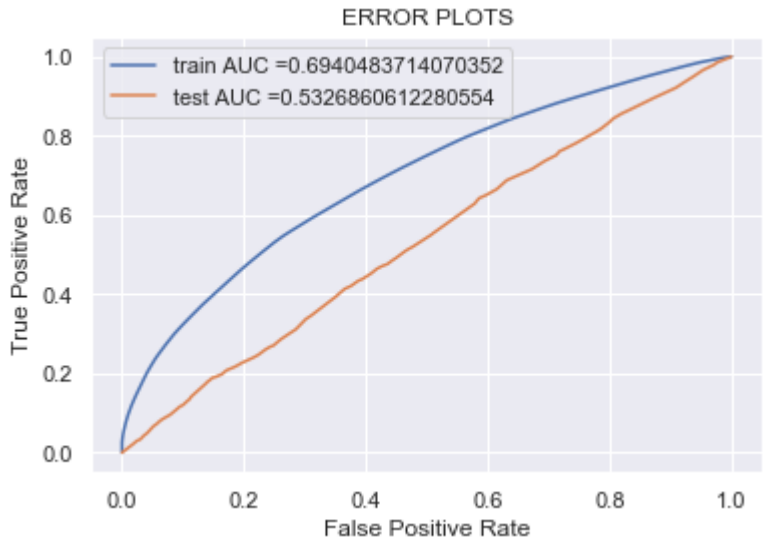
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight='balanced', criterion='gin
i',
                       max_depth=10, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
0.6940483714070352
0.532880611595636
```

In [100]:
```
best_tune_parameters= [{'max_depth': [10], 'min_samples_split':[500] }]
```

2.9.4 Receiver Operating Characteristic- TFIDF W2V

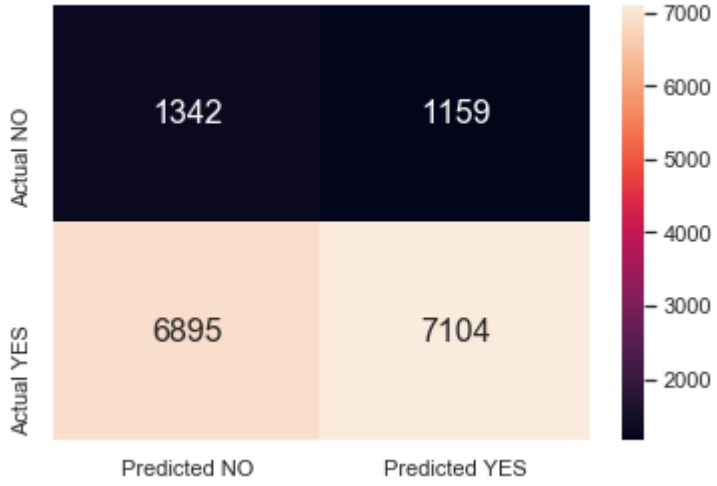
In [101]:

```
#Fitting Model to Hyper-Parameter Curve
from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, auc
clf11= DecisionTreeClassifier(class_weight = 'balanced',max_depth=10,min_samples_
# clfV1=DecisionTreeClassifier (class_weight = 'balanced',max_depth=5,min_samples
clf11.fit(X_tfidf_w2v_train, y_train)
# for visulation
# clfV1.fit(X_tfidf_w2v_train, y_train)
#https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDClassi
y_train_pred1 = clf11.predict_proba(X_tfidf_w2v_train) [:,1]
y_test_pred1 = clf11.predict_proba(X_tfidf_w2v_test) [:,1]
train_fpr1, train_tpr1, tr_thresholds1 = roc_curve(y_train, y_train_pred1)
test_fpr1, test_tpr1, te_thresholds1 = roc_curve(y_test, y_test_pred1)
plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr1, train_t
plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr1, test_tpr1))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



2.9.5 Confusion matrix- TFIDF W2V

In [102]:
```
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#function to get heatmap confusion matrix
def get_confusion_matrix(clf,X_te,y_test):
    y_pred = clf.predict(X_te)
    df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred), index =['Actual NO','A
    sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')

# %%time
get_confusion_matrix(clf11,X_tfidf_w2v_test,y_test)
```

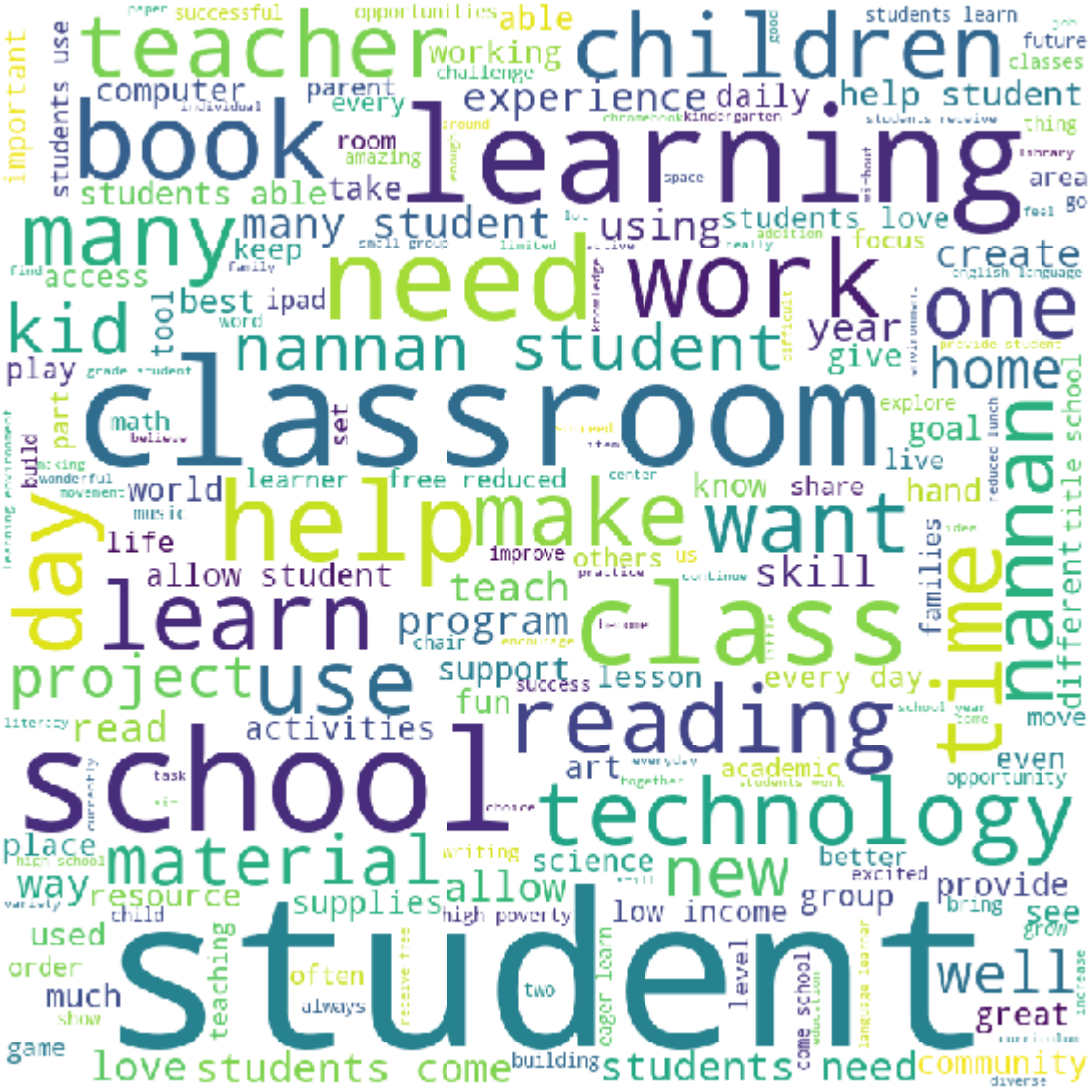


2.10 Selecting the FPR data points from Essay,Price and Teacher_number_of_previously_posted_projects test data set.

In [103]:
```
import numpy as np
import pandas as pd

y_pred = clf11.predict(X_tfidf_w2v_test)
#Essay
df=pd.DataFrame(Xtfidf_w2v_vectors_test_)
#Price
df_=X_test['price']
df_.to_frame()
#Teacher_number_of_previously_posted_projects
df_1=X_test['teacher_number_of_previously_posted_projects']
df_1.to_frame()

df1=pd.DataFrame(y_test,columns=['y_test'])
df2=pd.DataFrame(y_pred,columns=['y_pred'])

df_row = pd.concat([df,df1, df2],axis=1)
df_row1 = pd.concat([df_,df1, df2],axis=1)
df_row2 = pd.concat([df_1,df1, df2],axis=1)

#Selecting the false positive data points
df_row = df_row[(df_row.y_pred == 1) & (df_row.y_test == 0) ]
df_row1 = df_row1[(df_row1.y_pred == 1) & (df_row1.y_test == 0) ]
df_row2 = df_row2[(df_row2.y_pred == 1) & (df_row2.y_test == 0) ]
df_row2=df_row2.fillna(0)
```

2.11 WordCloud of False positive data points- `TFIDF W2V`

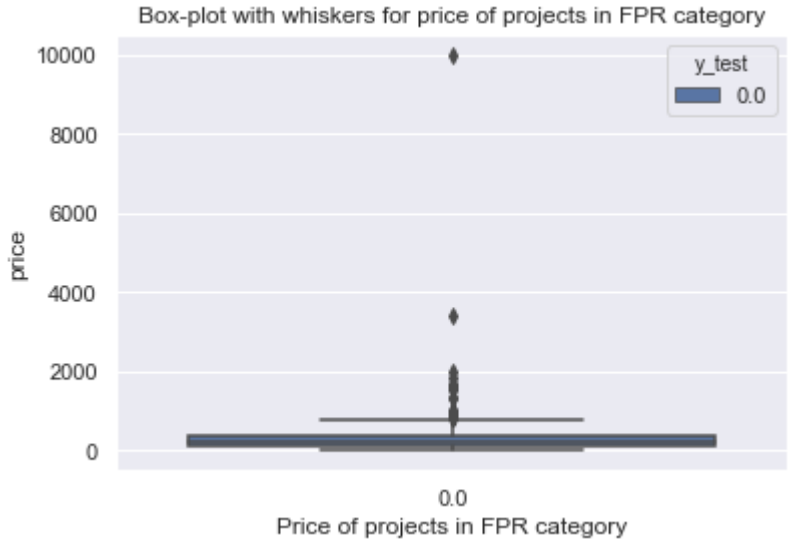
In [104]:
```
# https://www.geeksforgeeks.org/generating-word-cloud-python/

from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd

# Reads 'Youtube04-Eminem.csv' file
# df = pd.read_csv(r"Youtube04-Eminem.csv", encoding ="latin-1")

comment_words = ' '
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in df_row[0]:

    # 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()
```



2.12 Box plot of "price" of False positive data points

In [108]:

```
import seaborn as sns
import matplotlib.pyplot as plt
g=sns.boxplot(x='y_test',y='price',hue='y_test',data=df_row1)
plt.title("Box-plot with whiskers for price of projects in FPR category")
plt.xlabel("Price of projects in FPR category")
# plt.legend(loc=1)
plt.show()
```



2.13 PDF of teacher_number_of_previously_posted_projects for false positive data points.

In [106]:
```
counts, bin_edges1 = np.histogram(df_row2['teacher_number_of_previously_posted_pr
                                  density = True)
pdf1 = counts/(sum(counts))
print("The pdf is :",pdf1);
print("The bin edges of pdf are:",bin_edges1);
plt.plot(bin_edges1[1:],pdf1,label='PDF');
plt.title('PDF of teacher_number_of_previously_posted_projects for false positive
plt.xlabel("Bin Edges")
plt.ylabel("PDF ")
plt.legend()
```

```
The pdf is : [9.73252804e-01 1.72562554e-02 3.45125108e-03 8.62812770e-04
 2.58843831e-03 2.58843831e-03]
The bin edges of pdf are: [  0.   33.5  67.  100.5 134.  167.5 201. ]
```

Out[106]: <matplotlib.legend.Legend at 0x3b103f28>

In [109]:
```
# print("PDF1 of Patients who survived falling in >=5years from the graph-Blue l
print("The probability of finding project quote's less than",round(bin_edges1[1]
print("The probability of finding project quote's less than ",round(bin_edges1[2
print("The probability of finding project quote's less than ",round(bin_edges1[3
print("The probability of finding project quote's less than ",round(bin_edges1[4
print("The probability of finding project quote's less than ",round(bin_edges1[5
print("The probability of finding project quote's less than ",round(bin_edges1[6
```

```
The probability of finding project quote's less than 33.5 that got approved and
falling in FPR category are 97.07 %
The probability of finding project quote's less than  67.0 that got approved an
d falling in FPR category are 1.68 %
The probability of finding project quote's less than  100.5 that got approved a
nd falling in FPR category are 0.63 %
The probability of finding project quote's less than  134.0 that got approved a
nd falling in FPR category are 0.42 %
The probability of finding project quote's less than  167.5 that got approved a
nd falling in FPR category are 0.1 %
The probability of finding project quote's less than  201.0 that got approved a
nd falling in FPR category are 0.1 %
```

In [ ]:
```
#To make best use of the memory we are setting the variable names to 'None' and p
X_tfidf_w2v_test=None
X_tfidf_w2v_train=None
y_tfidf_w2v_train=None
gc.collect()
```

2.14 Pretty table summary

In [119]:
```
#http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer","Model", "Hyper Parameter-Alpha", "Test-AUC"]
# x.add_row(["BOW","Decision Tree", '0.0001', 0.64])
x.add_row(["Tfidf","Decision Tree" ,'max_depth:[10], min_samples_split:[500]', 0
# x.add_row(["avg_w2v","Decision Tree",'0.0001', 0.58])
x.add_row(["Tfidf- Non zero feature importance","Decision Tree",'max_depth: [10],
x.add_row(["Tfidf_w2v","Decision Tree",'max_depth: [10], min_samples_split:[500]
# x.add_row(["No_Text","Decision Tree",'0.0001', 0.52])
print(x)
```

```
+-----------------------------------+---------------+-------------------------
-----------------+----------+
|            Vectorizer             |     Model     |          Hyper Parameter
-Alpha           | Test-AUC |
+-----------------------------------+---------------+-------------------------
-----------------+----------+
|               Tfidf               | Decision Tree | max_depth:[10], min_samp
les_split:[500]  |   0.62   |
| Tfidf- Non zero feature importance | Decision Tree | max_depth: [10], min_sam
ples_split:[500] |   0.62   |
|            Tfidf_w2v              | Decision Tree | max_depth: [10], min_sam
ples_split:[500] |   0.53   |
+-----------------------------------+---------------+-------------------------
-----------------+----------+
```

In [ ]: