Gradient Boosting on Donor Choose Dataset

In [2]:

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

In [3]:
```
project_data = pd.read_csv('train_data.csv',nrows=35000)
resource_data = pd.read_csv('resources.csv')
project_data.reset_index()
resource_data.reset_index()
```

Out[3]:

	index	id	description	quantity	price
0	0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95
5	5	p069063	Last to Finish: A Story About the Smartest Boy...	1	16.99
6	6	p069063	Mrs. Gorski, I Think I Have the Wiggle Fidgets...	1	9.95
7	7	p069063	See-N-Read 1503905CQ Reading Tool - Book Size,...	2	10.11
8	8	p096795	Brewster WPD90218 Wall Pops Flirt Dot, Set of ...	2	9.95
9	9	p096795	Brewster Wall Pops WPE99065 Peel & Stick Calyp...	2	9.02
10	10	p096795	TIME For Kids - 3-4 PRINT Bundle - 24 issues /...	40	5.01
11	11	p149007	Ahora, Spanish, Grades 6 - 12, Level 2 (min. 1...	60	7.99
12	12	p149007	Scholastic News, Grades 5/6 (min. 10 subscript...	96	5.25
13	13	p149007	Science Spin Grades 3–6 - 8 Issues / Min. 10 S...	96	0.99
14	14	p236235	PP440X - Fairy Tales Problem Solving STEM Kits	2	149.00
15	15	p052460	DD165AT - Calming Colors® Easy-Clean Room...	1	129.00
16	16	p052460	DD165SB - Calming Colors® Easy-Clean Room...	1	129.00
17	17	p052460	DD165SE - Calming Colors® Easy-Clean Room...	1	129.00
18	18	p052460	DD165SG - Calming Colors® Easy-Clean Room...	1	129.00
19	19	p233680	AA758BU - Connect & Store Book Bin - Blue	4	4.99
20	20	p233680	AA758GR - Connect & Store Book Bin - Green	4	4.99
21	21	p233680	AA758RD - Connect & Store Book Bin - Red	4	4.99
22	22	p233680	AA758RG - Connect & Store Book Bin - Orange	4	4.99
23	23	p233680	AA758VT - Connect & Store Book Bin	5	4.99
24	24	p233680	AA758YE - Connect & Store Book Bin - Yellow	5	4.99
25	25	p233680	JJ302 - Books On Wheels Mobile Library - 6 Bins	1	149.00
26	26	p233680	LX468BU - Extra Storage Bin - Blue	2	8.99
27	27	p233680	LX468GR - Extra Storage Bin - Green	2	8.99
28	28	p233680	LX468RD - Extra Storage Bin - Red	2	8.99
29	29	p233680	LX468YE - Extra Storage Bin - Yellow	2	8.99
...	...	...	...	...	...
1541242	1541242	p187432	Samsung Chromebook, 11.6" Screen, 2 GB RAM, 16...	3	202.99

	index	id	description	quantity	price
1541243	1541243	p187432	Sentry Folding Headphones, Black	10	7.99
1541244	1541244	p187432	Sentry Folding Headphones, White	4	7.99
1541245	1541245	p149426	Piper Computer Kit | Educational Computer that...	1	299.00
1541246	1541246	p238803	CARPET MY FAVORITE COLORS-7FT6INX12FT	1	314.97
1541247	1541247	p087783	BALL STAY N PLACE SAND FILL	2	34.07
1541248	1541248	p087783	BR302BU - Comfy Floor Seat - Blue	1	49.99
1541249	1541249	p087783	BR302RD - Comfy Floor Seat - Red	1	49.99
1541250	1541250	p087783	CARDINAL (PP) - CLASSROOM SELECT	3	0.00
1541251	1541251	p087783	CF521GR - Giant Comfy Pillow - Green	1	69.99
1541252	1541252	p087783	OPTION CLASS - CS NEOCLASS/NEOMOVE SHELL COLOR...	3	0.00
1541253	1541253	p087783	STOOL - CS NEOROK - STOOL HEIGHT 12 - RUBBER B...	3	59.47
1541254	1541254	p086116	Apple iPad 2 2nd generation Tablet, 1 GHz proc...	1	124.99
1541255	1541255	p086116	Apple iPad with Retina Display MD513LL/A (16GB...	11	367.95
1541256	1541256	p086116	ProCase iPad Case 9.7" 2017 - Vintage Folio St...	3	11.99
1541257	1541257	p086116	iPad 2 Case, iPad 3 Case, iPad 4 Case, AiSMei ...	7	10.99
1541258	1541258	p086116	iPad 9.7 2017 Case (New 2017 Model), EasyAcc U...	2	9.90
1541259	1541259	p086116	iPad Mini Case, Apple iPad Mini 2 Case, iPad M...	1	14.99
1541260	1541260	p228679	AA162 - First 100 Sight-Words Talking Boards	1	59.99
1541261	1541261	p228679	EE809 - Magnetic Fishing Poles - Set of 2	2	12.99
1541262	1541262	p228679	FF468 - Magnetic Sight-Word Sentence Board	1	29.99
1541263	1541263	p228679	TT507 - Fishing for Sight-Words - Level 1	1	21.99
1541264	1541264	p183340	42 PC GRADESTUFF MID SCHOOL - PACK OF 42	1	219.10
1541265	1541265	p183340	Rubbermaid Commercial FG9S3100GRAY Brute Tote ...	1	27.49
1541266	1541266	p031981	5pcs DC3V/0.1A 1.5V/0.05A 10x2.7mm Coin Mobile...	2	6.46
1541267	1541267	p031981	AmazonBasics 9 Volt Everyday Alkaline Batterie...	1	9.99
1541268	1541268	p031981	AmazonBasics AAA Performance Alkaline Batterie...	1	6.99
1541269	1541269	p031981	Black Electrical Tape (GIANT 3 PACK) Each Roll...	6	8.99
1541270	1541270	p031981	Flormoon DC Motor Mini Electric Motor 0.5-3V 1...	2	8.14
1541271	1541271	p031981	WAYLLSHINE 6PCS 2 x 1.5V AAA Battery Spring Cl...	2	7.39

1541272 rows × 5 columns

In [4]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.project_is_approved.value_counts()
```

```
Number of data points in train data (35000, 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']
```

Out[4]:
```
1    29611
0     5389
Name: project_is_approved, dtype: int64
```

In [5]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[5]:

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

1.2 preprocessing of `project_subject_categories`

In [6]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflo

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-f
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-stri
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 o
            j=j.replace('The','') # if we have the words "The" we are going t
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(em
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tr
        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]))
```

In [7]:
```
sorted_cat_dict
```

Out[7]:
```
{'Warmth': 466,
 'Care_Hunger': 466,
 'History_Civics': 1891,
 'Music_Arts': 3280,
 'AppliedLearning': 3879,
 'SpecialNeeds': 4357,
 'Health_Sports': 4604,
 'Math_Science': 13186,
 'Literacy_Language': 16763}
```

1.3 preprocessing of project_subject_subcategories

In [8]:

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

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

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 o
            j=j.replace('The','') # if we have the words "The" we are going t
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(em
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tr
        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/228985
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])
```

In [9]:
```
1 sorted_sub_cat_dict
```

Out[9]:
```
{'Economics': 100,
 'CommunityService': 157,
 'FinancialLiteracy': 177,
 'ParentInvolvement': 214,
 'Extracurricular': 250,
 'Civics_Government': 266,
 'ForeignLanguages': 271,
 'NutritionEducation': 441,
 'Warmth': 466,
 'Care_Hunger': 466,
 'SocialSciences': 592,
 'PerformingArts': 611,
 'CharacterEducation': 659,
 'TeamSports': 712,
 'Other': 792,
 'College_CareerPrep': 809,
 'Music': 969,
 'History_Geography': 1020,
 'Health_LifeScience': 1295,
 'EarlyDevelopment': 1360,
 'ESL': 1393,
 'Gym_Fitness': 1483,
 'EnvironmentalScience': 1746,
 'VisualArts': 2039,
 'Health_Wellness': 3330,
 'AppliedSciences': 3497,
 'SpecialNeeds': 4357,
 'Literature_Writing': 7064,
 'Mathematics': 8941,
 'Literacy': 10896}
```

1.4 preprocessing of project grade categories

In [10]:
```
1 project_data['project_grade_category'].value_counts()
```

Out[10]:
```
Grades PreK-2    14199
Grades 3-5       11888
Grades 6-8        5415
Grades 9-12       3498
Name: project_grade_category, dtype: int64
```

In [11]:
```
1 preprocessed_project_grade_categories= []
2 for grade_cat in tqdm(project_data["project_grade_category"]):
3     grade_cat = grade_cat.replace('-', '_')  #Replacing(-) with(_)
4     grade_cat = grade_cat.replace('Grades', '') #Removing grades as it is rea
5     grad_cat = ' '.join(f for f in grade_cat.split())
6     preprocessed_project_grade_categories.append(grad_cat.strip())
```

```
100%|██████████| 35000/35000 [00:00<00:00, 584870.10it/s]
```

In [12]:
```
print(preprocessed_project_grade_categories[12])
print("="*50)
print(preprocessed_project_grade_categories[502])
print("="*50)
print(preprocessed_project_grade_categories[5002])
print("="*50)
# print(preprocessed_project_grade_categories[50002])
# print("="*50)
# print(preprocessed_project_grade_categories[100013])
# print("="*50)
```

```
6_8
==================================================
6_8
==================================================
6_8
==================================================
```

1.5 preprocessing of teacher prefix

In [13]:
```
project_data['teacher_prefix'].value_counts()
```

Out[13]:
```
Mrs.       18352
Ms.        12530
Mr.         3364
Teacher      752
Name: teacher_prefix, dtype: int64
```

Replacing Nan Values with maximum frequencies values i.e Mrs.

In [14]:
```
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('Mrs.'
```

In [15]:
```
def replace_cate(lst):          # Removing (.) in Mrs.
    return lst.replace('.','')
project_data['teacher_prefix']= project_data['teacher_prefix'].apply(replace_
```

In [16]:
```
preprocessed_teacher_prefix = []
for teach_prefix in tqdm(project_data["teacher_prefix"]):
    preprocessed_teacher_prefix.append(teach_prefix.strip())
```

```
100%|██████████| 35000/35000 [00:00<00:00, 2064764.69it/s]
```

In [17]:
```
print(preprocessed_teacher_prefix[1])
print("="*50)
print(preprocessed_teacher_prefix[50])
print("="*50)
project_data.teacher_prefix.value_counts()
```

```
Mr
==================================================
Mrs
==================================================
```

Out[17]:
```
Mrs        18354
Ms         12530
Mr          3364
Teacher      752
Name: teacher_prefix, dtype: int64
```

1.6 Adding a new feature

Introducing New Features Consider these set of features for Set 5 in Assignment:

categorical dataschool_state
clean_categories....clean_subcategories....project_grade_category....teacher_prefix

numerical data quantity....teacher_number_of_previously_posted_projects....price

New Features

sentiment score's of each of the essay : numerical data number of words in the title : numerical data number of words in the combine essays : numerical data

In [18]:
```
title_word_count = []
```

In [19]:
```
for a in project_data["project_title"] :
    b = len(a.split())
    title_word_count.append(b)
```

In [20]:
```
project_data["title_word_count"] = title_word_count
```

In [21]:
```
project_data.head(5)
```

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sub
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr	FL	20
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms	AZ	20
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs	KY	20
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs	TX	20

combining 4 essays into 1 essay

In [22]:
```
# 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)
```

Adding a new feature Number of words in essay

In [23]:
```
essay_word_count=[]
```

In [24]:
```
for ess in project_data["essay"] :
    c = len(ess.split())
    essay_word_count.append(c)
```

In [25]:
```
project_data["essay_word_count"] = essay_word_count
```

Computing Sentiment Score

In [26]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')
# import nltk
#nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach th
ss = sid.polarity_scores(for_sentiment)

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

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

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\honey\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
neg: 0.109, neu: 0.693, pos: 0.198, compound: 0.2023,
```

In [27]:
```
SID = SentimentIntensityAnalyzer()
#There is NEGITIVE and POSITIVE and NEUTRAL and COMPUND SCORES
#http://www.nltk.org/howto/sentiment.html

negitive = []
positive = []
neutral = []
compound = []
for i in tqdm(project_data['essay']):
    j = SID.polarity_scores(i)['neg']
    k = SID.polarity_scores(i)['neu']
    l = SID.polarity_scores(i)['pos']
    m = SID.polarity_scores(i)['compound']
    negitive.append(j)
    positive.append(k)
    neutral.append(l)
    compound.append(m)
```

```
100%|██████████| 35000/35000 [05:30<00:00, 105.75it/s]
```

In [28]:
```
project_data['negitive'] = negitive
project_data['positive'] = positive
project_data['neutral'] = neutral
project_data['compound'] = compound
```

In [29]:
```
1 project_data.head(2)
```

Out[29]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr	FL	20

2 rows × 24 columns

In [30]:
```
1 project_data.columns.values
```

Out[30]:
```
array(['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',
       'title_word_count', 'essay', 'essay_word_count', 'negitive',
       'positive', 'neutral', 'compound'], dtype=object)
```

Train Test split

In [31]:
```
1 # train test split using sklearn.model selection
2 from sklearn.model_selection import train_test_split
3
4 X_train, X_test, y_train, y_test = train_test_split(project_data, project_dat
5 #X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=
```

In [32]:
```
1 project_data.head()
```

Out[32]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sub
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr	FL	20
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms	AZ	20
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs	KY	20
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs	TX	20

5 rows × 24 columns

In [33]:
```
1 X_train.shape
```

Out[33]: (23450, 24)

In [34]:
```
1 X_train.drop(['project_is_approved'], axis=1, inplace=True)
2 X_test.drop(['project_is_approved'], axis=1, inplace=True)
3 #X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

In [35]:
```
1 X_train.head(2)
```

Out[35]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	projec
2205	61258	p110928	da47958d7d9e9ab5550166aa5f42d500	Mrs	OH	
31234	133706	p159437	d5ee4cd2ab601b71382ac1cf7094cbff	Ms	CA	

2 rows × 23 columns

1.8 Text preprocessing

In [36]:
```
# printing some random reviews

print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[500])
print("="*50)
```

```
This project will benefit 25, 4-5 year olds, who are first timers to the public
school setting!  They are excited about being in school and learning!  They wil
l be filled with lots of energy, enthusiasm, and a true love of learning!  This
will be a mix of both boys and girls.  This group of students will include stud
ents who achieve and grow at a very rapid pace.  There may be students with spe
cial needs included in this group as well.With this project, I would like to pr
ovide my students with Boogie Boards with which to practice their writing.  The
se boards can be used to write on with a stylus or finger and then, with the to
uch of a button, can be erased to practice again!  They will provide the studen
ts with a form of technology that is both fun and effective!  They are durable
and will hold the students attention.  Boogie Boards will provide the students
with a fun way to practice their writing for years to come!  I am hoping to rec
eive a set of 4 of these Boogie Boards so that several students can work indivi
dually or in pairs.nannan
==================================================
My kiddos will experience a 21st century learning environment where they get to
make choices about how hey learn. We are not a sit and get classroom. Tradition
al classroom seating is boring. I want to encourage collaboration and creative
thinking, one size fits all chairs are yesterday's classroom.My students are of
lower socioeconomic status and we have an extremely high percentage of free and
reduced lunches. Many of them come from truly broken homes with many having a s
ingle parent to support them  and many are being raised by grandparents. To giv
e my kiddos these choices in class gives them a safe homelike feeling that they
may not otherwise have.My kiddos will be using these as a seating option anytim
e they need to sit and complete a task, wether it be independent, partners or s
mall groups. It will be one of the many flexible seating options that I hope to
offer to my Kinders.  The will have he freedoms to use these at a low table or
to use them to lounge with a clipboard. They need to be free to determine what
seating options work for them!Seeing a child truly engaged in their learning an
d making choices about the way they learn best. This helps them to learn more a
bout who they are at a young age. Many behavior problems are virtually eliminat
ed and they are allowed to move and learn. They are able to retain more and tak
e an active role in their learning at a very young age. I hope to be able to ge
t many more seating options for my kiddos to encourage and inspire them to do a
nd be their very best in everything they do!
==================================================
```

In [37]:
```
# 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 [38]:
```
sent = decontracted(X_train['essay'].values[20000])
print(sent)
print("="*50)
```

```
In my class I normally read a shared text with the students, we work on various
writing pieces, and they are given time to read and write independently. When r
eading a shared text, we often have class discussions and Socratic Seminars. Wh
en writing, we have workshops and conferences in small groups.My students are i
n the 7th and 8th grade. I have students from many different races and religion
s. As a result, I promote and model tolerance. We often read about history, and
learn about novels in context, because I believe it is very important for stude
nts gain deeper awareness of cultural context whenever we read a new text. For
example, when we recently read Night by Elie Wiesel, the students also gained a
deeper understanding of WWII and the Holocaust, so that they might be able to b
etter understand and empathize with Wiesel is memoir. We also recently read To
Kill a Mockingbird, by Harper Lee,  and the students learned more about the Gre
at Depression and Jim Crowe South in order to better understand the plot and th
emes within the text. I believe that students need to be made aware of history
in order to better understand not only what they read, but the world around the
m today, and even themselves.Most of my students do enjoy reading independently
whether they are in my Integrated Co-Teaching class, or Honors class. However,
I am in need of a greater variety of books for them to choose from. I am also i
n need of a document camera so that I might aid my students with their writing,
through modeling. This will also allow students to share their work with one an
other, and receive feedback.I think rejuvenating my classroom library with new
books will give my students a greater interest and choice in what they read. I
think that acquiring the document camera will allow me to better help my studen
ts with their writing because I will be able to model writing for them, show th
em how to revise and edit more easily, and allow them to provide feedback to an
other by sharing their work on the board.
==================================================
```

In [39]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
In my class I normally read a shared text with the students, we work on various
writing pieces, and they are given time to read and write independently. When r
eading a shared text, we often have class discussions and Socratic Seminars. Wh
en writing, we have workshops and conferences in small groups.My students are i
n the 7th and 8th grade. I have students from many different races and religion
s. As a result, I promote and model tolerance. We often read about history, and
learn about novels in context, because I believe it is very important for stude
nts gain deeper awareness of cultural context whenever we read a new text. For
example, when we recently read Night by Elie Wiesel, the students also gained a
deeper understanding of WWII and the Holocaust, so that they might be able to b
etter understand and empathize with Wiesel is memoir. We also recently read To
Kill a Mockingbird, by Harper Lee,  and the students learned more about the Gre
at Depression and Jim Crowe South in order to better understand the plot and th
emes within the text. I believe that students need to be made aware of history
in order to better understand not only what they read, but the world around the
m today, and even themselves.Most of my students do enjoy reading independently
whether they are in my Integrated Co-Teaching class, or Honors class. However,
I am in need of a greater variety of books for them to choose from. I am also i
n need of a document camera so that I might aid my students with their writing,
through modeling. This will also allow students to share their work with one an
other, and receive feedback.I think rejuvenating my classroom library with new
books will give my students a greater interest and choice in what they read. I
think that acquiring the document camera will allow me to better help my studen
ts with their writing because I will be able to model writing for them, show th
em how to revise and edit more easily, and allow them to provide feedback to an
other by sharing their work on the board.
```

In [40]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
In my class I normally read a shared text with the students we work on various
writing pieces and they are given time to read and write independently When rea
ding a shared text we often have class discussions and Socratic Seminars When w
riting we have workshops and conferences in small groups My students are in the
7th and 8th grade I have students from many different races and religions As a
result I promote and model tolerance We often read about history and learn abou
t novels in context because I believe it is very important for students gain de
eper awareness of cultural context whenever we read a new text For example when
we recently read Night by Elie Wiesel the students also gained a deeper underst
anding of WWII and the Holocaust so that they might be able to better understan
d and empathize with Wiesel is memoir We also recently read To Kill a Mockingbi
rd by Harper Lee and the students learned more about the Great Depression and J
im Crowe South in order to better understand the plot and themes within the tex
t I believe that students need to be made aware of history in order to better u
nderstand not only what they read but the world around them today and even them
selves Most of my students do enjoy reading independently whether they are in m
y Integrated Co Teaching class or Honors class However I am in need of a greate
r variety of books for them to choose from I am also in need of a document came
ra so that I might aid my students with their writing through modeling This wil
l also allow students to share their work with one another and receive feedback
I think rejuvenating my classroom library with new books will give my students
a greater interest and choice in what they read I think that acquiring the docu
ment camera will allow me to better help my students with their writing because
I will be able to model writing for them show them how to revise and edit more
easily and allow them to provide feedback to another by sharing their work on t
he board
```

In [41]:
```
# 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', 'i
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', '
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'beca
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'th
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shou
            'won', "won't", 'wouldn', "wouldn't"]
```

1.8.1 Preprocessesd training data - Text

In [42]:
```
from tqdm import tqdm
def preprocess_textual(row_value):
    preprocessed_train = []
    for sentences in tqdm(row_value):
        sent = decontracted(sentences)
        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_train.append(sent.lower().strip())
    return preprocessed_train
```

In [43]:
```
preprocessed_essays_train=preprocess_textual(X_train['essay'].values)
```

```
100%|██████████| 23450/23450 [00:11<00:00, 2073.20it/s]
```

In [44]:
```
preprocessed_essays_test=preprocess_textual(X_test['essay'].values)
```

```
100%|██████████| 11550/11550 [00:05<00:00, 2078.18it/s]
```

In [45]:
```
preprocessed_essays_test
```

Out[45]:
```
['want introduce students world beyond know give opportunity dream big create
better future students difficult time paying attention class concerned sleepi
ng night next meal coming sometimes difficult pay attention kinds issues desp
ite high mobility extreme poverty many issues school kids active participants
learning excited classroom strive make learning environment one sanctuary kid
s forget problems focus education using donors choose get students great lite
rature students love things received lead great classroom book discussions ph
enomenal new vocabulary adventures make believe land far away places built gr
eat comprehension skills however classroom library lacking books students wan
t read say success breeds success want hep students successful possible stude
nts started book wish list classrom white board writing book titles books wan
t see classroom library kids get check books library school complain turn fin
ished reading book books want already checked project allow students read boo
k cover cover without return library chance complete books students asked nan
nan',
 'students hard working bright young individuals attend school every day read
y learn meet high expectations students love challenges difficulties face eve
ry day smiles faces no less students come title school high population englis
h learners come low income household limited resources experiences many stude
```

1.8.3 Preprocessed cross validation data

In [46]:
```
#preprocessed_essays_cv=preprocess_textual(X_cv['essay'].values)
```

1.9 preprocessing of project title

In [47]:
```
# printing some randomproject titles.
print(project_data['project_title'].values[1])
print("="*50)
print(project_data['project_title'].values[1501])
print("="*50)
print(project_data['project_title'].values[10001])
print("="*50)
print(project_data['project_title'].values[20001])
print("="*50)
```

```
Wanted: Projector for Hungry Learners
==================================================
Making Every Day at School Count
==================================================
Becoming 'Readerly' Readers
==================================================
The Beautiful Life of a Butterfly
==================================================
```

In [48]:
```
title = decontracted(X_train['project_title'].values[2000])
```

1.9.1 Preprocessing of Project Title(Train)

In [49]:
```
preprocessed_titles_train=preprocess_textual(X_train["project_title"].values)
```

```
100%|██████████| 23450/23450 [00:00<00:00, 49289.76it/s]
```

1.9.2 Preprocessing of Project Title(Test)

In [50]:
```
preprocessed_titles_test=preprocess_textual(X_test["project_title"])
```

```
100%|██████████| 11550/11550 [00:00<00:00, 45842.64it/s]
```

In [51]:
```
preprocessed_titles_test[10]
```

Out[51]:
```
'leap reading new leapfrog tag reader pens'
```

1.9.2 Preprocessing of Project Title(CV)

In [52]:
```
#preprocessed_titles_cv=preprocess_textual(X_cv["project_title"])
```

In [53]:
```
#preprocessed_titles_cv[10]
```

1.5 Preparing data for models

In [54]:
```
1 X_train.shape
```

Out[54]:
```
(23450, 23)
```

In [55]:
```
1 project_data.columns
```

Out[55]:
```
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', 'title_word_count', 'essay',
       'essay_word_count', 'negitive', 'positive', 'neutral', 'compound'],
      dtype='object')
```

we are going to consider

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

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

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

In [56]:
```
1 project_data.head(10)
```

Out[56]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_su
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN	2(
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr	FL	2(
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms	AZ	2(
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs	KY	2(
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs	TX	2(
5	141660	p154343	a50a390e8327a95b77b9e495b58b9a6e	Mrs	FL	2(
6	21147	p099819	9b40170bfa65e399981717ee8731efc3	Mrs	CT	2(
7	94142	p092424	5bfd3d12fae3d2fe88684bbac570c9d2	Ms	GA	2(
8	112489	p045029	487448f5226005d08d36bdd75f095b31	Mrs	SC	2(

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_su
9	158561	p001713	140eeac1885c820ad5592a409a3a8994	Ms	NC	2(

10 rows × 24 columns

1.5.1 Vectorizing Categorical data

In [57]:
```
def Responsetable(table, col) :
    cat = table[col].unique()
    freq_Pos = []
    for i in cat :
        freq_Pos.append(len(table.loc[(table[col] == i) & (table['project_is_

    freq_Neg = []
    for i in cat :
        freq_Neg.append(len(table.loc[(table[col] == i) & (table['project_is_

    encoded_Pos = []
    for i in range(len(cat)) :
        encoded_Pos.append(freq_Pos[i]/(freq_Pos[i] + freq_Neg[i]))

    encoded_Neg = []
    encoded_Neg[:] = [1 - x for x in encoded_Pos]

    encoded_Pos_val = dict(zip(cat, encoded_Pos))
    encoded_Neg_val = dict(zip(cat, encoded_Neg))

    return encoded_Pos_val, encoded_Neg_val
```

In [58]:
```
def Responsecode(train,test) :
    pos_cleancat, neg_cleancat = Responsetable(train,'clean_categories')
    pos_cleansubcat, neg_cleansubcat = Responsetable(train,'clean_subcategori
    pos_schoolstate, neg_schoolstate = Responsetable(train, 'school_state')
    pos_teacherprefix, neg_teacherprefix = Responsetable(train, 'teacher_pref
    pos_projgradecat, neg_projgradecat = Responsetable(train, 'project_grade_
    dftrain = pd.DataFrame()
    dftrain['clean_cat_pos'] = train['clean_categories'].map(pos_cleancat)
    dftrain['clean_cat_neg'] = train['clean_categories'].map(neg_cleancat)
    dftrain['clean_subcat_pos'] = train['clean_subcategories'].map(pos_cleans
    dftrain['clean_subcat_neg'] = train['clean_subcategories'].map(neg_cleans
    dftrain['school_state_pos'] = train['school_state'].map(pos_schoolstate)
    dftrain['school_state_neg'] = train['school_state'].map(neg_schoolstate)
    dftrain['teacher_prefix_pos'] = train['teacher_prefix'].map(pos_teacherpr
    dftrain['teacher_prefix_neg'] = train['teacher_prefix'].map(neg_teacherpr
    dftrain['proj_grade_cat_pos'] = train['project_grade_category'].map(pos_p
    dftrain['proj_grade_cat_neg'] = train['project_grade_category'].map(neg_p
    dftest = pd.DataFrame()
    dftest['clean_cat_pos'] = test['clean_categories'].map(pos_cleancat).fill
    dftest['clean_cat_neg'] = test['clean_categories'].map(neg_cleancat).fill
    dftest['clean_subcat_pos'] = test['clean_subcategories'].map(pos_cleansub
    dftest['clean_subcat_neg'] = test['clean_subcategories'].map(neg_cleansub
    dftest['school_state_pos'] = test['school_state'].map(pos_schoolstate).fi
    dftest['school_state_neg'] = test['school_state'].map(neg_schoolstate).fi
    dftest['teacher_prefix_pos'] = test['teacher_prefix'].map(pos_teacherpref
    dftest['teacher_prefix_neg'] = test['teacher_prefix'].map(neg_teacherpref
    dftest['proj_grade_cat_pos'] = test['project_grade_category'].map(pos_pro
    dftest['proj_grade_cat_neg'] = test['project_grade_category'].map(neg_pro
    return dftrain,dftest
```

In [59]:
```
X_new_train=X_train
X_new_train['project_is_approved']=y_train.to_frame()['project_is_approved']
X_new_test=X_test
X_new_test['project_is_approved']=y_test.to_frame()['project_is_approved']
newTrain,newTest = Responsecode(X_new_train,X_new_test)
```

In [60]:
```
print(newTrain.shape,newTest.shape)
```

```
(23450, 10) (11550, 10)
```

In [61]:
```
def mergeEncoding(table, p, n) :
    lstPos = table[p].values.tolist()
    lstNeg = table[n].values.tolist()
    frame = pd.DataFrame(list(zip(lstNeg, lstPos)))
    return frame
```

In [62]:
```
#Clean Categories
X_train_clean_cat_ohe = mergeEncoding(newTrain, 'clean_cat_pos', 'clean_cat_n
X_test_clean_cat_ohe = mergeEncoding(newTest, 'clean_cat_pos', 'clean_cat_neg
print(X_train_clean_cat_ohe.shape)
X_test_clean_cat_ohe.shape
```

```
(23450, 2)
```

Out[62]:
```
(11550, 2)
```

In [63]:
```
#Clean SUB Categories
X_train_clean_subcat_ohe = mergeEncoding(newTrain, 'clean_subcat_pos', 'clean
X_test_clean_subcat_ohe = mergeEncoding(newTest, 'clean_subcat_pos', 'clean_s
print(X_train_clean_subcat_ohe.shape)
print(X_test_clean_subcat_ohe.shape)
```

```
(23450, 2)
(11550, 2)
```

In [64]:
```
#Project Grade Category
X_train_grade_ohe = mergeEncoding(newTrain, 'proj_grade_cat_pos', 'proj_grade
X_test_grade_ohe = mergeEncoding(newTest, 'proj_grade_cat_pos', 'proj_grade_c
print(X_train_grade_ohe.shape)
print(X_test_grade_ohe.shape)
```

```
(23450, 2)
(11550, 2)
```

In [65]:
```
#School State
X_train_state_ohe = mergeEncoding(newTrain, 'school_state_pos', 'school_state
X_test_state_ohe = mergeEncoding(newTest, 'school_state_pos', 'school_state_n
print(X_train_state_ohe.shape)
print(X_test_state_ohe.shape)
```

```
(23450, 2)
(11550, 2)
```

In [66]:
```
#Teacher Prefix
X_train_teacher_ohe = mergeEncoding(newTrain, 'teacher_prefix_pos', 'teacher_
X_test_teacher_ohe = mergeEncoding(newTest, 'teacher_prefix_pos', 'teacher_pr
print(X_train_teacher_ohe.shape)
print(X_test_teacher_ohe.shape)
```

```
(23450, 2)
(11550, 2)
```

TFIDF

tfidf(train essays)

In [67]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay = TfidfVectorizer(min_df=10,max_features=5000) #Consia
vectorizer_tfidf_essay.fit(preprocessed_essays_train)

text_tfidf_train = vectorizer_tfidf_essay.transform(preprocessed_essays_train
print("Shape of matrix after TfidfVectorizer ",text_tfidf_train.shape)
```

```
Shape of matrix after TfidfVectorizer  (23450, 5000)
```

tfidf(test essays)

In [68]:
```
text_tfidf_test = vectorizer_tfidf_essay.transform(preprocessed_essays_test)
print("Shape of matrix after TfidfVectorizer ",text_tfidf_test.shape)
```

```
Shape of matrix after TfidfVectorizer  (11550, 5000)
```

tfidf(cv essays)

In [69]:
```
#text_tfidf_cv = vectorizer_tfidf_essay.transform(preprocessed_essays_cv)
#print("Shape of matrix after TfidfVectorizer ",text_tfidf_cv.shape)
```

tfidf(train Titles)

In [70]:
```
vectorizer_tfidf_titles = TfidfVectorizer(min_df=10)

vectorizer_tfidf_titles.fit(preprocessed_titles_train)
title_tfidf_train = vectorizer_tfidf_titles.transform(preprocessed_titles_tra
print("Shape of matrix after TfidfVectorizer ",title_tfidf_train.shape)
```

```
Shape of matrix after TfidfVectorizer  (23450, 1217)
```

tfidf(test titles)

In [71]:
```
title_tfidf_test = vectorizer_tfidf_titles.transform(preprocessed_titles_test
print("Shape of matrix after TfidfVectorizer ",title_tfidf_test.shape)
```

```
Shape of matrix after TfidfVectorizer  (11550, 1217)
```

tfidf(cv titles)

In [72]:
```
#title_tfidf_cv = vectorizer_tfidf_titles.transform(preprocessed_titles_cv)
#print("Shape of matrix after TfidfVectorizer ",title_tfidf_cv.shape)
```

1.5.2.3 Using Pretrained Models: TFIDF Weighted W2V

In [73]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/ho
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

train essays

In [74]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays_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 [75]:
```
def tfidfWV(preprocessed_data):
    # average Word2Vec
    # compute average word2vec for each review.
    tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored
    for sentence in tqdm(preprocessed_data): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        tf_idf_weight =0; # num of words with a valid vector in the sentence/
        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_idf = dictionary[word]*(sentence.count(word)/len(sentence.
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors.append(vector)
    print("The length of TFIDF word to vec is ", len(tfidf_w2v_vectors))
    print("The length of TFIDF word to vec of index 0 is ",len(tfidf_w2v_vect
    return tfidf_w2v_vectors
```

In [76]:
```
tfidf_w2v_vectors_train=tfidfWV(preprocessed_essays_train)
```

```
100%|██████████| 23450/23450 [00:35<00:00, 656.44it/s]

The length of TFIDF word to vec is  23450
The length of TFIDF word to vec of index 0 is  300
```

Test essays

In [77]:
```
tfidf_w2v_vectors_test=tfidfWV(preprocessed_essays_test)
```

```
100%|██████████| 11550/11550 [00:26<00:00, 431.10it/s]

The length of TFIDF word to vec is  11550
The length of TFIDF word to vec of index 0 is  300
```

cv essays

In [78]:
```
#tfidf_w2v_vectors_cv = tfidfWV(preprocessed_essays_cv)
```

train titles

In [79]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles_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 [80]:
```
tfidf_w2v_vectors_titles_train = tfidfWV(preprocessed_titles_train)
```

```
100%|██████████| 23450/23450 [00:00<00:00, 28506.97it/s]

The length of TFIDF word to vec is  23450
The length of TFIDF word to vec of index 0 is  300
```

test titles

In [81]:
```
tfidf_w2v_vectors_titles_test = tfidfWV(preprocessed_titles_test)
```

```
100%|██████████| 11550/11550 [00:00<00:00, 27768.46it/s]

The length of TFIDF word to vec is  11550
The length of TFIDF word to vec of index 0 is  300
```

cv titles

In [82]:
```
#tfidf_w2v_vectors_titles_cv = tfidfWV(preprocessed_titles_cv)
```

1.12 Vectorizing Numerical features

Various numerical feautures are :

1.Price

2.Quantity

3.Number of Projects previously proposed by Teacher

4.Title word Count (introduced by us)

5.Essay word Count (introduced by us)

6.Negative Intensity

7.Positive Intensity

8.Neutral Intensity

In [83]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-inde
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'
price_data.head(4)
```

Out[83]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98

In [84]:
```
price_data.head(4)
```

Out[84]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98

In [85]:
```
1 X_train
```

Out[85]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pro
2205	61258	p110928	da47958d7d9e9ab5550166aa5f42d500	Mrs	OH	
31234	133706	p159437	d5ee4cd2ab601b71382ac1cf7094cbff	Ms	CA	
7957	89555	p090465	5c0c521aa9aa0cf31e32e4fa1fbb3deb	Ms	CA	
9891	49379	p038756	17561728716402d8cf0c380c3c358813	Mrs	WI	

In [86]:
```
1 # join two dataframes in python:
2 X_train = pd.merge(X_train, price_data, on='id')
3 X_test = pd.merge(X_test, price_data, on='id')
4 #X_cv = pd.merge(X_cv, price_data, on='id', how='left')
```

In [87]:
```
1 X_train
```

Out[87]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pro
0	61258	p110928	da47958d7d9e9ab5550166aa5f42d500	Mrs	OH	
1	133706	p159437	d5ee4cd2ab601b71382ac1cf7094cbff	Ms	CA	
2	89555	p090465	5c0c521aa9aa0cf31e32e4fa1fbb3deb	Ms	CA	
3	49379	p038756	17561728716402d8cf0c380c3c358813	Mrs	WI	

In [88]:
```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

price_for_test=X_test['price']    #Box Plot Purpose

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

price_train = normalizer.transform(X_train['price'].values.reshape(-1,1))
#price_cv = normalizer.transform(X_cv['price'].values.reshape(-1,1))
price_test = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(price_train.shape, y_train.shape)
#print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
====================================================================================================
```

2) Quantity

In [89]:
```
normalizer = Normalizer()

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

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

quantity_train = normalizer.transform(X_train['quantity'].values.reshape(-1,1
#quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(quantity_train.shape, y_train.shape)
#print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
====================================================================================================
```

3) Number of Projects previously proposed by Teacher

In [90]:
```
normalizer = Normalizer()

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

normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values

prev_projects_train = normalizer.transform(X_train['teacher_number_of_previou
#prev_projects_cv = normalizer.transform(X_cv['teacher_number_of_previously_p
number_of_previously_posted_projects_dtrees=X_test['teacher_number_of_previou
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previousl

print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
#print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
====================================================================================================
```

4) title word count

In [91]:
```
normalizer = Normalizer()

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

title_word_count_train = normalizer.transform(X_train['title_word_count'].val
#title_word_count_cv = normalizer.transform(X_cv['title_word_count'].values.r
title_word_count_test = normalizer.transform(X_test['title_word_count'].value

print("After vectorizations")
print(title_word_count_train.shape, y_train.shape)
#print(title_word_count_cv.shape, y_cv.shape)
print(title_word_count_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
====================================================================================================
```

5) essay word count

In [92]:
```
normalizer = Normalizer()

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

essay_word_count_train = normalizer.transform(X_train['essay_word_count'].val
#essay_word_count_cv = normalizer.transform(X_cv['essay_word_count'].values.r
essay_word_count_test = normalizer.transform(X_test['essay_word_count'].value

print("After vectorizations")
print(essay_word_count_train.shape, y_train.shape)
#print(essay_word_count_cv.shape, y_cv.shape)
print(essay_word_count_test.shape, y_test.shape)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
```

6) Positive Intensity

In [93]:
```
normalizer = Normalizer()

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

positive_intensity_train = normalizer.transform(X_train['positive'].values.re
#positive_intensity_cv = normalizer.transform(X_cv['positive'].values.reshape
positive_intensity_test = normalizer.transform(X_test['positive'].values.resh

print("After vectorizations")
print(positive_intensity_train.shape, y_train.shape)
#print(positive_intensity_cv.shape, y_cv.shape)
print(positive_intensity_test.shape, y_test.shape)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
```

7) Negative Intensity

In [94]:
```
normalizer = Normalizer()

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

negative_intensity_train = normalizer.transform(X_train['negitive'].values.re
#negative_intensity_cv = normalizer.transform(X_cv['negitive'].values.reshape
negative_intensity_test = normalizer.transform(X_test['negitive'].values.resh

print("After vectorizations")
print(negative_intensity_train.shape, y_train.shape)
#print(negative_intensity_cv.shape, y_cv.shape)
print(negative_intensity_test.shape, y_test.shape)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
```

8) Neutral Intensity

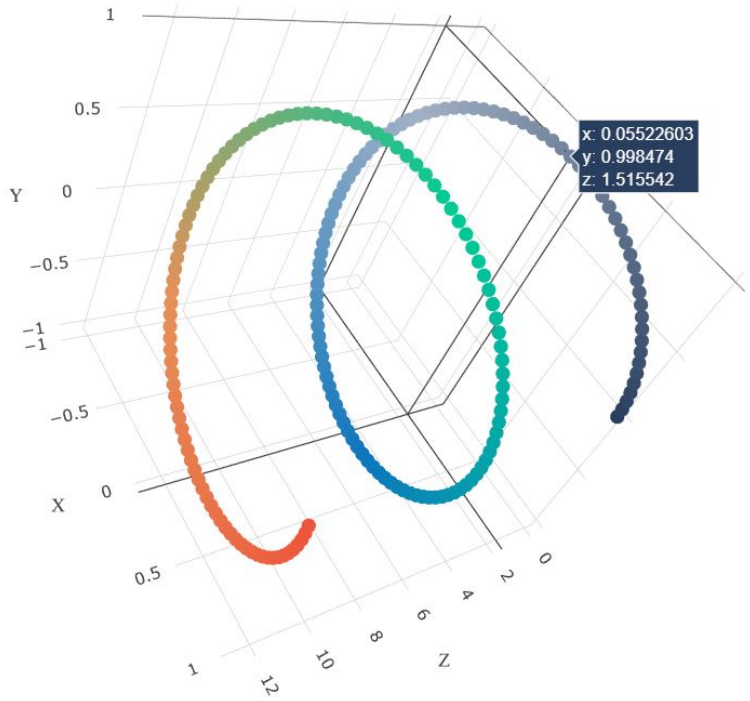
In [95]:
```
normalizer = Normalizer()

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

neutral_intensity_train = normalizer.transform(X_train['neutral'].values.resh
#neutral_intensity_cv = normalizer.transform(X_cv['neutral'].values.reshape(-
neutral_intensity_test = normalizer.transform(X_test['neutral'].values.reshap

print("After vectorizations")
print(neutral_intensity_train.shape, y_train.shape)
#print(neutral_intensity_cv.shape, y_cv.shape)
print(neutral_intensity_test.shape, y_test.shape)
```

```
After vectorizations
(23450, 1) (23450,)
(11550, 1) (11550,)
```

1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
    - Set 1: categorical, numerical features + preprocessed_essay (TFIDF) + Sentiment scores(preprocessed_essay)
    - Set 2: categorical, numerical features + preprocessed_essay (TFIDF W2V) + Sentiment scores(preprocessed_essay)
2. **The hyper paramter tuning (best `depth` in range [1, 3, 10, 30], and the best `min_samples_split` in range [5, 10, 100, 500])**
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - find the best hyper paramter using k-fold cross validation(use gridsearch cv or randomsearch cv)/simple cross validation data(you can write your own for loops refer sample solution)

3. **Representation of results**

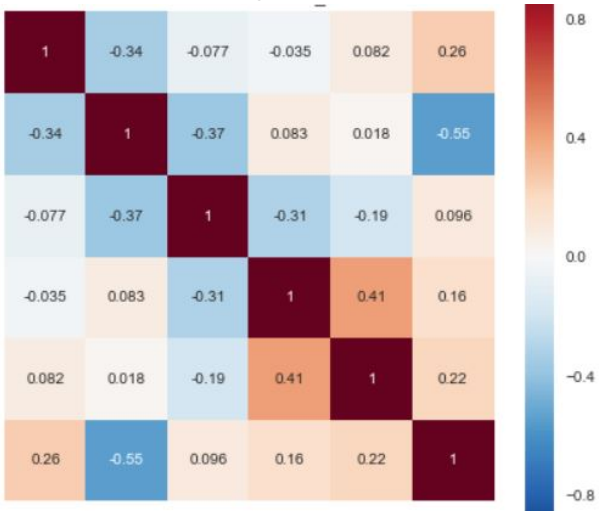
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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

or

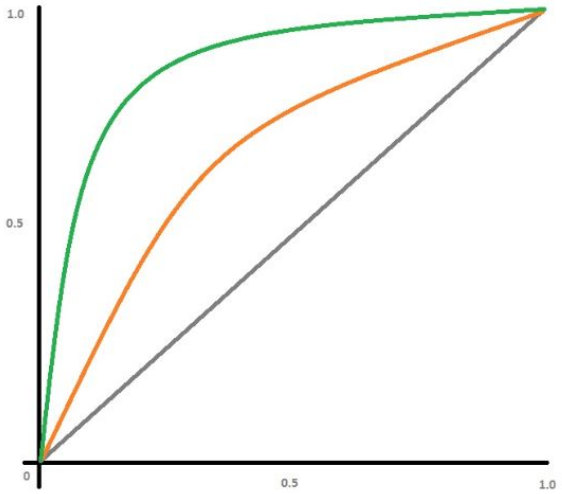
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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

- You choose either of the plotting techniques out of 3d plot or heat map

- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test. Make sure that you are using predict_proba method to calculate AUC curves, because AUC is calcualted on class probabilities and not on class labels.



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

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

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

In [96]:
```
from scipy.sparse import hstack
X_train = hstack((X_train_clean_cat_ohe , X_train_clean_subcat_ohe, X_train_s
                  X_train_teacher_ohe ,text_tfidf_train,title_tfidf_train, qu
                  prev_projects_train, price_train,title_word_count_train,
                  essay_word_count_train, positive_intensity_train, negative_
print(X_train.shape, y_train.shape)
print(type(X_train))
```

```
(23450, 6235) (23450,)
<class 'scipy.sparse.csr.csr_matrix'>
```

In [97]:
```
print(X_train_clean_cat_ohe.shape)
print(neutral_intensity_train.shape)
```

```
(23450, 2)
(23450, 1)
```

In [98]:
```
X_test = hstack((X_test_clean_cat_ohe  ,X_test_clean_subcat_ohe ,X_test_state
                 X_test_teacher_ohe ,text_tfidf_test, title_tfidf_test, quant
                  prev_projects_test, price_test,title_word_count_test,
                  essay_word_count_test, title_word_count_test, negative_inte
                  neutral_intensity_test)).tocsr()
print(X_test.shape, y_test.shape)
print(type(X_test))
```

```
(11550, 6235) (11550,)
<class 'scipy.sparse.csr.csr_matrix'>
```

We don't need CV data since it is automatically done by the optimization function of python

In [99]:
```
# X_cv = hstack((cv_categories_one_hot ,sub_categories_one_hot_cv,school_stat
#                  teacher_prefix_categories_one_hot_cv,text_tfidf_cv, title_
#                  prev_projects_cv, price_cv, title_word_count_cv,
#                   essay_word_count_cv ,positive_intensity_cv, negative_inte
#                   neutral_intensity_cv)).tocsr()
# print(X_cv.shape, y_cv.shape)
# print(type(X_cv))
```

Hyperparameter Tuning

In [100]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
GBDT = GradientBoostingClassifier()

parameters = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 100]}

classifier_5 = GridSearchCV(GBDT, parameters, cv=3, scoring='roc_auc',verbose
classifier_5.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits

[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:   27.4s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   50.5s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done  25 out of  36 | elapsed:  6.5min remaining:  2.9mi
n
[Parallel(n_jobs=-1)]: Done  29 out of  36 | elapsed: 29.0min remaining:  7.0mi
n
[Parallel(n_jobs=-1)]: Done  33 out of  36 | elapsed: 30.0min remaining:  2.7mi
n
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed: 42.1min finished
```

Out[100]:
```
GridSearchCV(cv=3, error_score='raise',
       estimator=GradientBoostingClassifier(criterion='friedman_mse', init=Non
e,
              learning_rate=0.1, loss='deviance', max_depth=3,
              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, n_estimators=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False),
       fit_params=None, iid=True, n_jobs=-1,
       param_grid={'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 10
0]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [101]:
```
max_depth_best_param=classifier_5.best_params_['max_depth']
min_samples_split_best_param=classifier_5.best_params_['min_samples_split']
print(classifier_5.best_params_)
```

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

In [102]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [103]:
```
x1 = [5, 10, 100, 500] #min_sample_split
y1 = [1, 3, 10, 30]#max_depth
z1 = classifier_5.cv_results_['mean_test_score']

x2 = [5, 10, 100, 500] #min_sample_split
y2 = [1, 3, 10, 30] #max_depth
z2 = classifier_5.cv_results_['mean_train_score']

```

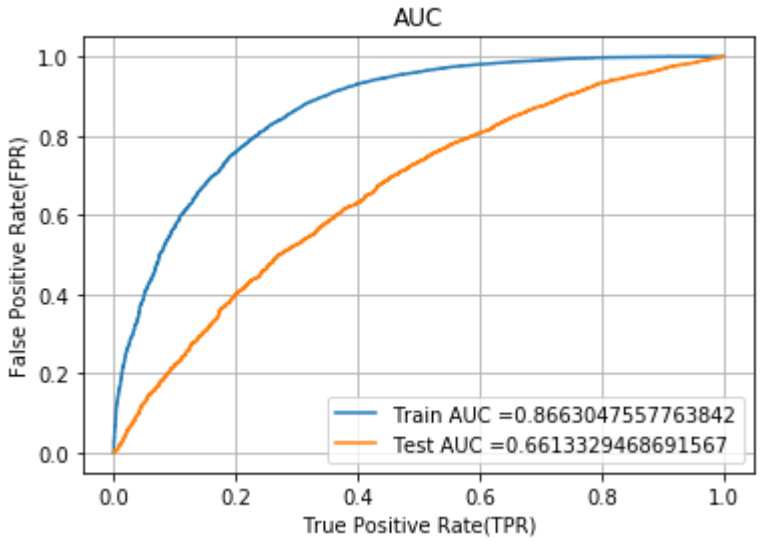
In [104]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross Validation')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Train')
data = [trace1, trace2]

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

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

In [105]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 490
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [106]:
```
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(fpr*(1-tpr))]
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshol
    predictions = []
    global predictions1
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    predictions1 = predictions
    return predictions
```

In [107]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.ensemble import GradientBoostingClassifier

classifier = GradientBoostingClassifier(max_depth = max_depth_best_param, min
classifier.fit(X_train, y_train)

y_train_pred = batch_predict(classifier,X_train)
y_test_pred = batch_predict(classifier,X_test)

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```

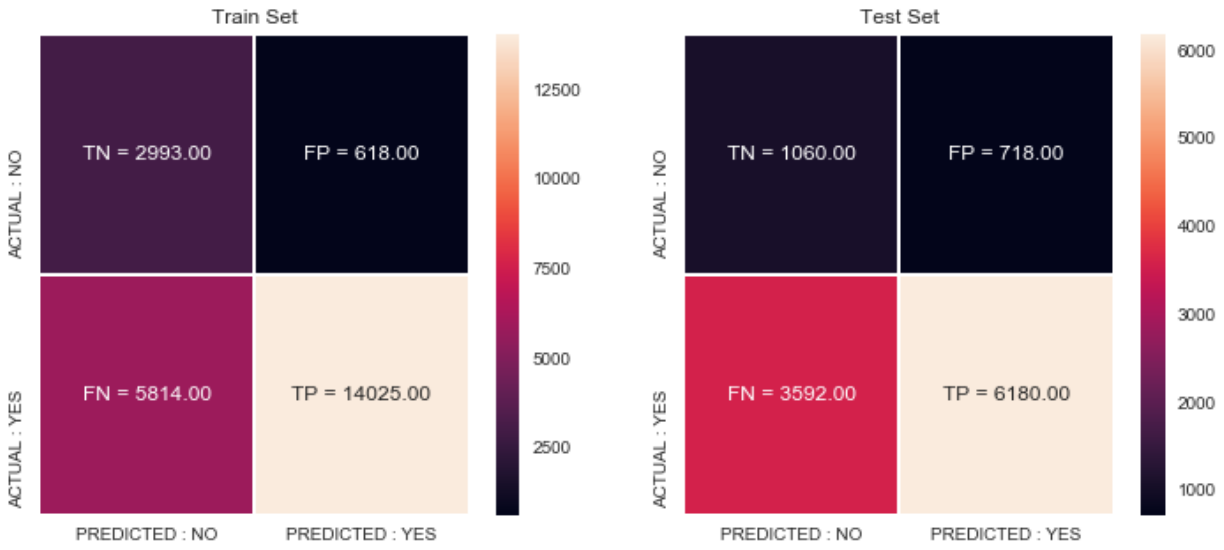


Confusion Matrix For Train & Test Data

In [108]:

```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
print("For the Train Data")
con_m_train = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
print("For the Test Data")
con_m_test = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, tes

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(12,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, valu
labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value

sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDI
sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')

plt.show()

```

```
For the Train Data
the maximum value of tpr*(1-fpr) 0.615496647765272 for threshold 0.851
For the Test Data
the maximum value of tpr*(1-fpr) 0.3831900515096276 for threshold 0.851
```



Set2 TFIDF W2V

In [109]:
```
from scipy.sparse import hstack
X_train = np.hstack((X_train_clean_cat_ohe,X_train_clean_subcat_ohe,X_train_s
                     X_train_teacher_ohe,tfidf_w2v_vectors_train,tfidf_w2v_vecto
                     prev_projects_train, price_train,title_word_count_train,
                     essay_word_count_train,
                     positive_intensity_train,
                     negative_intensity_train,
                     neutral_intensity_train))
print(X_train.shape, y_train.shape)
print(type(X_train))
# X_train=X_train.tocsr()
```

```
(23450, 618) (23450,)
<class 'numpy.ndarray'>
```

In [110]:
```
X_test = np.hstack((X_test_clean_cat_ohe  ,X_test_clean_subcat_ohe ,X_test_st
                    X_test_teacher_ohe ,tfidf_w2v_vectors_test, tfidf_w2v_vector
                     prev_projects_test, price_test,title_word_count_test,
                     essay_word_count_test, title_word_count_test, negative_inte
                     neutral_intensity_test))
print(X_test.shape, y_test.shape)
print(type(X_test))
```

```
(11550, 618) (11550,)
<class 'numpy.ndarray'>
```

In [111]:
```
# X_cv = hstack((cv_categories_one_hot ,sub_categories_one_hot_cv,school_stat
#                  teacher_prefix_categories_one_hot_cv,tfidf_w2v_vectors_cv,
#                  prev_projects_cv, price_cv, title_word_count_cv,
#                   essay_word_count_cv ,positive_intensity_cv, negative_inte
#                   neutral_intensity_cv)).tocsr()
# print(X_cv.shape, y_cv.shape)
# print(type(X_cv))
```

In [112]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
GBDT = GradientBoostingClassifier()

parameters = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 100]}

classifier_5 = GridSearchCV(GBDT, parameters, cv=3, scoring='roc_auc',verbose
classifier_5.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits

[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:  3.0min
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:  9.7min
[Parallel(n_jobs=-1)]: Done  25 out of  36 | elapsed: 31.6min remaining: 13.9mi
n
[Parallel(n_jobs=-1)]: Done  29 out of  36 | elapsed: 96.8min remaining: 23.4mi
n
[Parallel(n_jobs=-1)]: Done  33 out of  36 | elapsed: 101.4min remaining:  9.2m
in
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed: 120.2min finished
```

Out[112]:
```
GridSearchCV(cv=3, error_score='raise',
       estimator=GradientBoostingClassifier(criterion='friedman_mse', init=Non
e,
              learning_rate=0.1, loss='deviance', max_depth=3,
              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, n_estimators=100,
              presort='auto', random_state=None, subsample=1.0, verbose=0,
              warm_start=False),
       fit_params=None, iid=True, n_jobs=-1,
       param_grid={'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10, 10
0]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [113]:
```
max_depth_best_param=classifier_5.best_params_['max_depth']
min_samples_split_best_param=classifier_5.best_params_['min_samples_split']
print(classifier_5.best_params_)
```

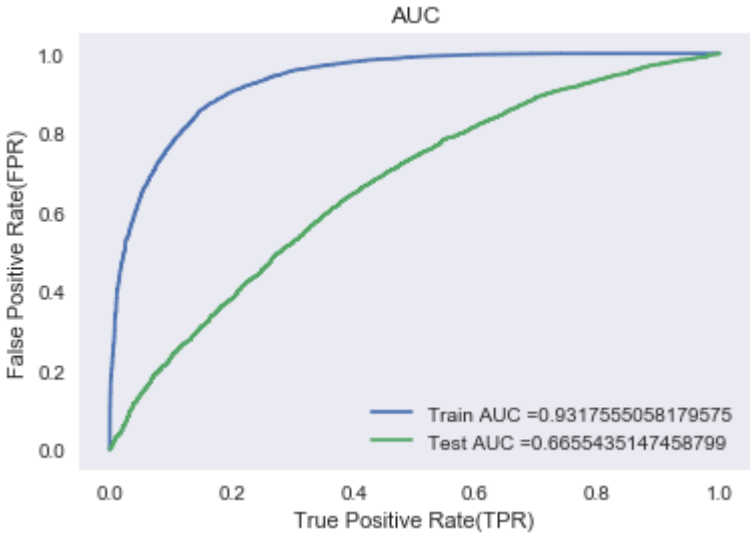
```
{'max_depth': 5, 'min_samples_split': 10}
```

In [114]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [115]:
```
x1 =  [1, 3, 10, 30]#min_sample_split
y1 = [5, 10, 100, 500]#max_depth
z1 = classifier_5.cv_results_['mean_test_score']

x2 = [1, 3, 10, 30] #min_sample_split
y2 = [5, 10, 100, 500] #max_depth
z2 = classifier_5.cv_results_['mean_train_score']

```

In [116]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross Validation')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Train')
data = [trace1, trace2]

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

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

In [117]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.ensemble import GradientBoostingClassifier

classifier = GradientBoostingClassifier(max_depth = max_depth_best_param, min
classifier.fit(X_train, y_train)

y_train_pred = batch_predict(classifier,X_train)
y_test_pred = batch_predict(classifier,X_test)

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```

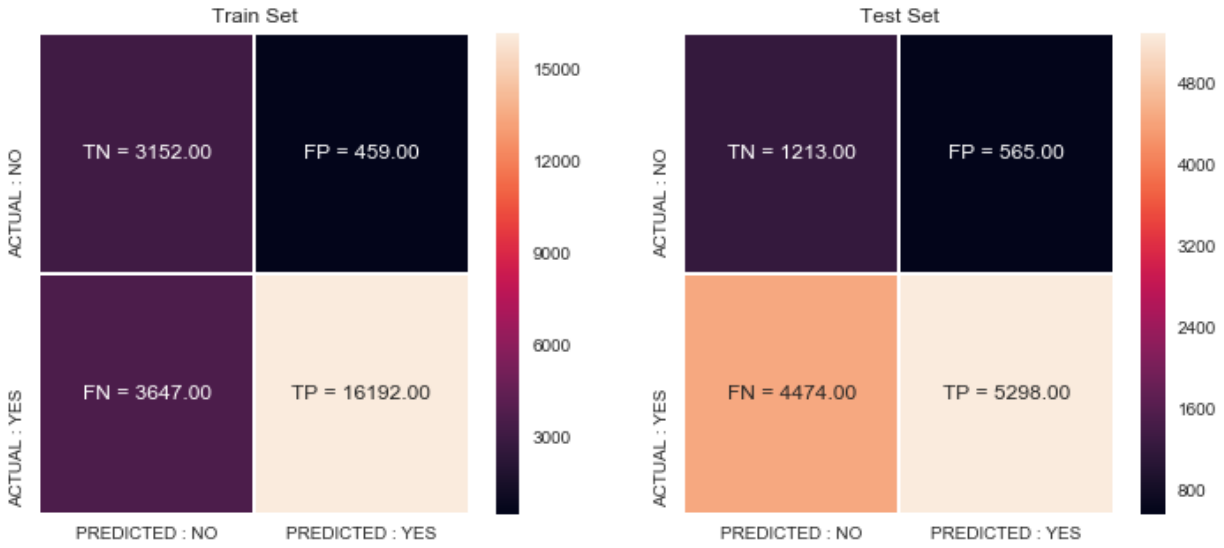


Confusion Matrix For Train & Test Data

In [118]:

```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
print("For the Train Data")
con_m_train = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
print("For the Test Data")
con_m_test = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, tes

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(12,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, valu
labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value

sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDI
sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')

plt.show()
```

```
For the Train Data
the maximum value of tpr*(1-fpr) 0.7302713875219472 for threshold 0.836
For the Test Data
the maximum value of tpr*(1-fpr) 0.3893065607896025 for threshold 0.876
```



Conclusion

In [120]:
```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
TB = PrettyTable()
TB.field_names = ["Model", "Hyperparameter", "Train_AUC", "Test_Auc"]
TB.title = "Decision Tree"
TB.add_row(["TFIDF", "Depth:5 | Samp_Split:100", 0.8663,0.6613])
TB.add_row(["TFIDF-W2V", "Depth:5 | Samp_Split:10", 0.93175, 0.665543])
print(TB)
```

```
+-------------------------------------------------------------+
|                        Decision Tree                        |
+-----------+--------------------------+-----------+----------+
|   Model   |      Hyperparameter      | Train_AUC | Test_Auc |
+-----------+--------------------------+-----------+----------+
|   TFIDF   | Depth:5 | Samp_Split:100 |   0.8663  |  0.6613  |
| TFIDF-W2V | Depth:5 | Samp_Split:10  |  0.93175  | 0.665543 |
+-----------+--------------------------+-----------+----------+
```

In [ ]:
```
1
```