DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

Feature	Description
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

# References: (1)Applied Ai course(https://www.appliedaicourse.com/) (2)Stackoverflow(https://stackoverflow.com/) (3)StackExchange(https://stackexchange.com/) (4)Google (https://www.google.com/)

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

Here I am taking 50000 dataset because my laptop doesnot have much computing power

1.1 Reading Data

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

In [3]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

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

In [4]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

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

In [5]:
```
project_data=project_data.head(50000)
project_data.shape
```

Out[5]:
```
(50000, 17)
```

In [6]:
```
resource_data=resource_data.head(50000)
resource_data.shape
```

Out[6]:
```
(50000, 4)
```

In [7]:
```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ["Date" if x=="project_submitted_datetime" else x for x in list(project_data.columns)]

#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data["Date"] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)

# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data=project_data[cols]
project_data.head(2)
```

Out[7]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_su
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Grades PreK-2	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5	Lit

In [8]:
```
print("Number of datapoints in train value", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of datapoints in train value (50000, 4)
['id' 'description' 'quantity' 'price']
```

Out[8]:

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

In [9]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [10]:
```
#https://stackoverflow.com/questions/18689823/pandas-dataframe-replace-nan-values-with-average-of-columns/186
91949
#Filling the nan value of price and quantity with mean if any
project_data['price'] = project_data['price'].fillna((project_data['price'].mean()))
project_data['quantity'] = project_data['quantity'].fillna((project_data['quantity'].mean()))
```

1.2 preprocessing of `project_subject_categories`

In [11]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [12]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

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

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

1.4 preprocessing of `teacher prefix

In [13]:
```
#project_data.teacher_prefix.replace(-1, np.nan)  #https://stackoverflow.com/questions/41882011/pandas-handli
ng-nans-in-categorical-data
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-attribute-split
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna("") #fill all NaN value with ""
prefix_teacher = list(project_data['teacher_prefix'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

project_data['teach_pref'] = teacher_prefix_list #create new column having name teach_pref with preprocessed
 data
project_data.drop(['teacher_prefix'], axis=1, inplace=True) #delete the teacher_prefix column

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
teach_pref_dict = dict(my_counter)
sorted_teach_pref_dict = dict(sorted(teach_pref_dict.items(), key=lambda kv: kv[1]))
```

1.5 preprocessing of `project_grade_category

In [14]:
```
#project_data.project_grade_category.replace(-1, np.nan)  #https://stackoverflow.com/questions/41882011/panda
s-handling-nans-in-categorical-data
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-attribute-split
project_data['project_grade_category']=project_data['project_grade_category'].fillna("") #fill all NaN value
 with ""
project_grad_cat = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

project_data['project_grad_cat'] = project_grad_list    #create new column having name project_grad_cat with
 preprocessed data
project_data.drop(['project_grade_category'], axis=1, inplace=True) #delete the project_grade_category column

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grad_dict = dict(my_counter)  #this will make a dictionary with keys and values of words and its coun
ts
sorted_project_grad_dict = dict(sorted(project_grad_dict.items(), key=lambda kv: kv[1])) #result a sorted dic
tionary by number of counts
```

In [15]:
```
#project_data.project_grade_category.replace(-1, np.nan)  #https://stackoverflow.com/questions/41882011/panda
s-handling-nans-in-categorical-data
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-attribute-split
project_data['school_state']=project_data['school_state'].fillna("") #fill all NaN value with ""
project_school_state = list(project_data['school_state'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

project_data['project_school_state'] = project_school_state_list    #create new column having name project_gr
ad_cat with preprocessed data
project_data.drop(['school_state'], axis=1, inplace=True) #delete the project_grade_category column

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_school_state_dict = dict(my_counter)  #this will make a dictionary with keys and values of words and
 its counts
sorted_project_school_state_dict = dict(sorted(project_school_state_dict.items(), key=lambda kv: kv[1])) #res
ult a sorted dictionary by number of counts
```

1.3 Text preprocessing

In [16]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [17]:
```
project_data.head(2)
```

Out[17]:

	Unnamed: 0	id	teacher_id	Date	project_title	project_essay_1	project_essay_2	project_essay_3	prc
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	re

In [18]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [19]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
```

```
I recently read an article about giving students a choice about how they learn. We already set goals; why not
let them choose where to sit, and give them options of what to sit on?I teach at a low-income (Title 1) schoo
l. Every year, I have a class with a range of abilities, yet they are all the same age. They learn differentl
y, and they have different interests. Some have ADHD, and some are fast learners. Yet they are eager and acti
ve learners that want and need to be able to move around the room, yet have a place that they can be comforta
ble to complete their work.We need a classroom rug that we can use as a class for reading time, and students 
can use during other learning times. I have also requested four Kore Kids wobble chairs and four Back Jack pa
dded portable chairs so that students can still move during whole group lessons without disrupting the class.
Having these areas will provide these little ones with a way to wiggle while working.Benjamin Franklin once s
aid, \"Tell me and I forget, teach me and I may remember, involve me and I learn.\" I want these children to 
be involved in their learning by having a choice on where to sit and how to learn, all by giving them options
for comfortable flexible seating.
==================================================
At the beginning of every class we start out with a Math Application problem to help students see the relevan
ce of topics in math. We are always in groups and do a lot of cooperative activities. We also use lots of tec
hnology in our class. I love seeing my students grow and love math!I have a very diverse population of studen
ts from all different races, SES, and experiences. My students love school and are starting to embrace the ha
rd work it takes to be a fifth grader.  My school is a 5th/6th grade school only and is considered a school f
or the middle grades. It is located in a suburban area. It is now more diverse than it has been in many year
s.I am in an inclusion setting and many of my students have disabilities. It is hard for them to see the boar
d because our resources are old and outdated.  A new document camera for our classroom will allow our student
s to see the board more clearly during instructional times and will create a classroom environment where lots
of movement isn't necessary just because my students cannot see the board.It's frustrating to teach a lesson 
when many of my students can't see the board because the resources I have are old and outdated. Oftentimes st
udents will tell me to wait before moving on because it takes them forever to write notes because they cannot
see the materials.  I want students to enjoy coming to my class to learn math and not feel frustrated because
they cannot see the board.
==================================================
My students love coming to school and they love learning. I strive daily to make our classroom a relaxed, com
fortable and welcoming environment where all learners will excel and grow in their learning. And a new rug wi
ll make our days even brighter!My 2nd grade classroom is filled with 20 amazing young learners. These student
s fill my heart everyday with their passion for learning new things. Working with these students and how enga
ged they are in each subject matter is so much fun. We are small elementary school in mid-Missouri and we hav
e an 80 percent free and reduced lunch rate. I have a wide range of learners in my classroom, and all of my s
tudents learn in different ways. So it is important to provide a learning environment that meets all student
s.A beautiful new carpet will be the focal point of our classroom. The carpet will be full of students all da
y long. It will be a clean and comfortable place where my students will find comfort in learning. Students wi
ll be sitting in small groups, laying and reading a book or even dancing on the carpet for brain breaks durin
g the day. A carpet in an elementary classroom is the heart of where learning takes place!Thank you for donat
ing or considering a donation to this project. I want to make my 2nd grade classroom as comfortable and invit
ing as Starbucks or as cozy as a grandma's living room! This beautiful carpet will be a perfect addition to a
classroom the is filled with so much excitement and enthusiasm!
==================================================
```

In [20]:
```
# 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 [21]:
```
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

```
I teach in an elementary school that is a 4th / 5th grade building in a small town in central Illinois. Next
year I will be teaching three different classes of students reading and language / writing / spelling. In my
classroom, my students enjoy a variety of activities including hands-on and collaborative learning in order t
o help make the information real and interesting to them while giving them a reason to practice.\r\n\r\nOur s
tudents are a wide-variety of students at our school with over 60% of our students receiving free lunch. Beca
use of this low-income percentage, our students often require additional help and support to help make their
learning valuable and real-world to them. Our teachers work hard to collaborate in order to help all of our s
tudents achieve at their highest level.\r\n\r\nOur community is very supportive of our schools, but lately be
cause of lower levels of state support many local businesses have cut back on individual assistance for class
rooms.  In order to continue some of our learning projects, we have had to look to other support to help us o
ut.Next year we will be focusing a great deal of our ELA (English Language Arts) time in 5th grade to improvi
ng our writing across all the curriculum:  math, science, reading, language, and social studies.  These indiv
idual marker boards will give my students the ability to practice writing skills individually while giving me
the ability to check individual is work as they practice.  They will also allow them to add creativity to the
ir writing and vocabulary practice. \r\n\r\nThese boards are an amazing tool in the classroom, and all kids e
njoy them! They offer them the benefit of working by themselves and making errors that they can then learn to
fix - such an important step in the learning process.nannan
==================================================
```

In [22]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
I teach in an elementary school that is a 4th / 5th grade building in a small town in central Illinois. Next
year I will be teaching three different classes of students reading and language / writing / spelling. In my
classroom, my students enjoy a variety of activities including hands-on and collaborative learning in order t
o help make the information real and interesting to them while giving them a reason to practice.    Our stude
nts are a wide-variety of students at our school with over 60% of our students receiving free lunch. Because
of this low-income percentage, our students often require additional help and support to help make their lear
ning valuable and real-world to them. Our teachers work hard to collaborate in order to help all of our stude
nts achieve at their highest level.    Our community is very supportive of our schools, but lately because of
lower levels of state support many local businesses have cut back on individual assistance for classrooms.  I
n order to continue some of our learning projects, we have had to look to other support to help us out.Next y
ear we will be focusing a great deal of our ELA (English Language Arts) time in 5th grade to improving our wr
iting across all the curriculum:  math, science, reading, language, and social studies.  These individual mar
ker boards will give my students the ability to practice writing skills individually while giving me the abil
ity to check individual is work as they practice.  They will also allow them to add creativity to their writi
ng and vocabulary practice.     These boards are an amazing tool in the classroom, and all kids enjoy them! T
hey offer them the benefit of working by themselves and making errors that they can then learn to fix - such
an important step in the learning process.nannan
```

In [23]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
I teach in an elementary school that is a 4th 5th grade building in a small town in central Illinois Next yea
r I will be teaching three different classes of students reading and language writing spelling In my classroo
m my students enjoy a variety of activities including hands on and collaborative learning in order to help ma
ke the information real and interesting to them while giving them a reason to practice Our students are a wid
e variety of students at our school with over 60 of our students receiving free lunch Because of this low inc
ome percentage our students often require additional help and support to help make their learning valuable an
d real world to them Our teachers work hard to collaborate in order to help all of our students achieve at th
eir highest level Our community is very supportive of our schools but lately because of lower levels of state
support many local businesses have cut back on individual assistance for classrooms In order to continue some
of our learning projects we have had to look to other support to help us out Next year we will be focusing a 
great deal of our ELA English Language Arts time in 5th grade to improving our writing across all the curricu
lum math science reading language and social studies These individual marker boards will give my students the
ability to practice writing skills individually while giving me the ability to check individual is work as th
ey practice They will also allow them to add creativity to their writing and vocabulary practice These boards
are an amazing tool in the classroom and all kids enjoy them They offer them the benefit of working by themse
lves and making errors that they can then learn to fix such an important step in the learning process nannan
```

In [24]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'thos
e', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'd
oes', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'o
f', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before',
'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again',
'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'fe
w', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm',
'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'must
n',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'were
n', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [25]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['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 [00:32<00:00, 1552.88it/s]
```

In [26]:
```
# after preprocesing
preprocessed_essays[2000]
```

Out[26]:
```
'i teach elementary school 4th 5th grade building small town central illinois next year i teaching three diff
erent classes students reading language writing spelling in classroom students enjoy variety activities inclu
ding hands collaborative learning order help make information real interesting giving reason practice our stu
dents wide variety students school 60 students receiving free lunch because low income percentage students of
ten require additional help support help make learning valuable real world our teachers work hard collaborate
order help students achieve highest level our community supportive schools lately lower levels state support
many local businesses cut back individual assistance classrooms in order continue learning projects look supp
ort help us next year focusing great deal ela english language arts time 5th grade improving writing across c
urriculum math science reading language social studies these individual marker boards give students ability p
ractice writing skills individually giving ability check individual work practice they also allow add creativ
ity writing vocabulary practice these boards amazing tool classroom kids enjoy they offer benefit working mak
ing errors learn fix important step learning process nannan'
```

In [27]:
```
project_data['preprocessed_essay'] = preprocessed_essays    #create new column having name project_grad_cat w
ith preprocessed data
project_data.drop(['essay'], axis=1, inplace=True) #delete the project_grade_category column
project_data.head(2)
```

Out[27]:

	Unnamed: 0	id	teacher_id	Date	project_title	project_essay_1	project_essay_2	project_essay_3	pro
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	re

1.4 Preprocessing of `project_title`

In [28]:
```
# similarly you can preprocess the titles also
```

In [29]:
```
#Printing some random titles
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
```

```
Flexible Seating for Flexible Learning
==================================================
Elmo for Math Instruction
==================================================
There's Only One You in This Great Big World
==================================================
```

In [30]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    title_sent = decontracted(sentance)
    title_sent = title_sent.replace('\\r', ' ')
    title_sent = title_sent.replace('\\"', ' ')
    title_sent = title_sent.replace('\\n', ' ')
    title_sent = re.sub('[^A-Za-z0-9]+', ' ', title_sent)
    # https://gist.github.com/sebleier/554280
    title_sent = ' '.join(e for e in title_sent.split() if e not in stopwords)
    preprocessed_titles.append(title_sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:01<00:00, 35082.36it/s]
```

In [31]:
```
project_data['preprocessed_title'] = preprocessed_titles    #create new column having name project_grad_cat w
ith preprocessed data
project_data.drop(['project_title'], axis=1, inplace=True) #delete the project_grade_category column
project_data.head()
```

Out[31]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_essay_2	project_essay_3	project_essay_4
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	Benjamin Franklin once said, \"Tell me and I f...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	These remarkable gifts will provide students w...
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My students desire challenges, movement, and c...	I will design different clues using specific c...	Donations to this project will immediately imp...
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Language Arts and Social Justice Magnet Sc...	\"Is it my turn, Ms. K? When am I going to be ...	By donating to this project, you will give my ...
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I have the privilege of teaching an incredible...	Ideally, I would love to delve right into \"fl...	This project will be so beneficial for my stud...

In [32]:
```
project_data.head(5)
```

Out[32]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_essay_2	project_essay_3	project_essay_4
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	Benjamin Franklin once said, \"Tell me and I f...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	These remarkable gifts will provide students w...
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My students desire challenges, movement, and c...	I will design different clues using specific c...	Donations to this project will immediately imp...
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Language Arts and Social Justice Magnet Sc...	\"Is it my turn, Ms. K? When am I going to be ...	By donating to this project, you will give my ...
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I have the privilege of teaching an incredible...	Ideally, I would love to delve right into \"fl...	This project will be so beneficial for my stud...

In [33]:
```
#https://stackoverflow.com/questions/9288169/python-word-length-function-example-needed
def count(line):
    num_text=[]
    for words in line:
        splitted = words.split()
        length = len(splitted)
        num_text.append(length)
    return num_text
```

In [34]:
```
project_data['num_title'] = count(project_data['preprocessed_title'])    #create new column having name proje
ct_grad_cat with preprocessed data
project_data.head(5)
```

Out[34]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_essay_2	project_essay_3	project_essay_4
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	Benjamin Franklin once said, \"Tell me and I f...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	These remarkable gifts will provide students w...
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My students desire challenges, movement, and c...	I will design different clues using specific c...	Donations to this project will immediately imp...
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Language Arts and Social Justice Magnet Sc...	\"Is it my turn, Ms. K? When am I going to be ...	By donating to this project, you will give my ...
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I have the privilege of teaching an incredible...	Ideally, I would love to delve right into \"fl...	This project will be so beneficial for my stud...

5 rows × 21 columns

In [35]:
```
project_data['num_essay'] = count(project_data['preprocessed_essay'])    #create new column having name proje
ct_grad_cat with preprocessed data
project_data.head(5)
```

Out[35]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_essay_2	project_essay_3	project_essay_4
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	Benjamin Franklin once said, \"Tell me and I f...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	These remarkable gifts will provide students w...
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My students desire challenges, movement, and c...	I will design different clues using specific c...	Donations to this project will immediately imp...
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Language Arts and Social Justice Magnet Sc...	\"Is it my turn, Ms. K? When am I going to be ...	By donating to this project, you will give my ...
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I have the privilege of teaching an incredible...	Ideally, I would love to delve right into \"fl...	This project will be so beneficial for my stud...

5 rows × 22 columns

1.5 Preparing data for models

In [36]:
```
project_data.columns
```

Out[36]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'Date', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'quantity', 'price', 'clean_categories', 'clean_subcategories',
       'teach_pref', 'project_grad_cat', 'project_school_state',
       'preprocessed_essay', 'preprocessed_title', 'num_title', 'num_essay'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [37]:
```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [38]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

1.5.2.3 Using Pretrained Models: Avg W2V

In [39]:
```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
Output:

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

# ============================

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

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

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

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


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

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


'''
```

Out[39]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef loadGloveModel(gloveF
ile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    
for line in tqdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n        embedding = np.a
rray([float(val) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model),"
words loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ======================
======\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfo
r i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))
\nwords = set(words)\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(model.keys()).
intersection(words)\nprint("The number of words that are present in both glove vectors and our coupus",
len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = s
et(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word
2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python: http://www.jessicayun
g.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\',
\'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

1.5.4 Merging all the above features

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

Computing Sentiment Scores

In [40]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

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

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students with the big
gest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelligences i us
e a wide range\
of techniques to help all my students succeed students in my class come from a variety of different backgroun
ds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a caring community
of successful \
learners which can be seen through collaborative student project based learning in and out of the classroom k
indergarteners \
in my class love to work with hands on materials and have many different opportunities to practice a skill be
fore it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten
 curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role play in our pr
etend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food i will take
their idea \
and create common core cooking lessons where we learn important math and writing concepts while cooking delic
ious healthy \
food for snack time my students will have a grounded appreciation for the work that went into making the food
and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project would expand ou
r learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce make
 our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create our own cookbooks to be
printed and \
shared with families students will gain math and literature skills as well as a life long enjoyment for healt
hy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

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

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

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

In [41]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.downloader.download('vader_lexicon')

# import nltk
# nltk.download('vader_lexicon')
#https://www.programcreek.com/python/example/100005/nltk.sentiment.vader.SentimentIntensityAnalyzer

def analyze_sentiment(project_data):
    sentiments = []
    sid = SentimentIntensityAnalyzer()
    for i in range(project_data.shape[0]):
        line = project_data['preprocessed_essay'].iloc[i]
        sentiment = sid.polarity_scores(line)
        sentiments.append([sentiment['neg'], sentiment['pos'],
                           sentiment['neu'], sentiment['compound']])
    project_data[['neg', 'pos', 'neu', 'compound']] = pd.DataFrame(sentiments)
    return project_data
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/navneetkumarbitsindri/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [42]:
```
project_data=analyze_sentiment(project_data)
project_data.head(5)
```

Out[42]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_essay_2	project_essay_3	project_essay_4
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	Benjamin Franklin once said, \"Tell me and I f...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	These remarkable gifts will provide students w...
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My students desire challenges, movement, and c...	I will design different clues using specific c...	Donations to this project will immediately imp...
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Language Arts and Social Justice Magnet Sc...	\"Is it my turn, Ms. K? When am I going to be ...	By donating to this project, you will give my ...
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I have the privilege of teaching an incredible...	Ideally, I would love to delve right into \"fl...	This project will be so beneficial for my stud...

5 rows × 26 columns

Assignment 9: RF and GBDT

Response Coding: Example

The response tabel is built only on train dataset. For a category which is not there in train data and present in test data, we will encode them with default values Ex: in our test data if have State: D then we encode it as [0.5, 0.05]

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

    - Set 1: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

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

    - 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/simple cross validation data
    - use gridsearch cv or randomsearch cv or you can write your own for loops to do this task

3. **Representation of results**

    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure with X-axis as **n_estimators**, Y-axis as **max_depth**, and Z-axis as **AUC Score** , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

or

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

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

4. **Conclusion**

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

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. Random Forest and GBDT

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

In [43]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [44]:
```
y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
project_data.head(3)
```

Out[44]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_essay_2	project_essay_3	project_essay_4
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We need a classroom rug that we can use as a c...	Benjamin Franklin once said, \"Tell me and I f...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the new common core standards that have b...	These remarkable gifts will provide students w...
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My students desire challenges, movement, and c...	I will design different clues using specific c...	Donations to this project will immediately imp...

3 rows × 25 columns

In [45]:
```
#train_test_split
from sklearn.model_selection import train_test_split
project_data_train, project_data_test, project_data_y_train, project_data_y_test = train_test_split(project_d
ata, y, test_size=0.33, stratify=y)
project_data_train, project_data_cv, project_data_y_train, project_data_y_cv = train_test_split(project_data_
train, project_data_y_train, test_size=0.33, stratify=project_data_y_train)
```

In [46]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

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

In [47]:
```
project_data_train.head(2)
```

Out[47]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_essay_2	project_essay_3	project_ess
27763	60154	p138036	7e1f3b0fed7bf310b17e2b5312fd36b1	2016-10-19 18:07:11	My students love coming to school. They love ...	Every day my students go on Lexia for reading ...	NaN	
49872	32991	p009150	64611fb9b3482e2117e5114a89c8659f	2017-04-28 16:56:44	John F. Kennedy once said: \"Ask not what your...	As a social studies teacher I am looking to en...	NaN	

2 rows × 25 columns

2.2 Make Data Model Ready: categorical features

Here I am doing target encoding in categorical data

In [48]:
```
from category_encoders import *
```

In [49]:
```
#https://github.com/scikit-learn-contrib/categorical-encoding
#http://contrib.scikit-learn.org/categorical-encoding/targetencoder.html
#https://maxhalford.github.io/blog/target-encoding-done-the-right-way/
```

# use target encoding to encode two categorical features enc = TargetEncoder(cols=['CHAS', 'RAD']).fit(X_train, y_train) # transform the datasets training_numeric_dataset = enc.transform(X_train, y_train) testing_numeric_dataset = enc.transform(X_test)

In [50]:
```
vectorizer3 = TargetEncoder().fit(project_data_train['clean_categories'].values, project_data_y_train)
#print(vectorizer3.get_params())

categories_one_hot_train = vectorizer3.transform(project_data_train['clean_categories'].values)
categories_one_hot_cv = vectorizer3.transform(project_data_cv['clean_categories'].values)
categories_one_hot_test = vectorizer3.transform(project_data_test['clean_categories'].values)

print("After vectorizations")
print(categories_one_hot_train.shape, project_data_y_train.shape)
print(categories_one_hot_cv.shape, project_data_y_cv.shape)
print(categories_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

In [51]:
```
vectorizer1 = TargetEncoder().fit(project_data_train['teach_pref'].values, project_data_y_train)
#print(vectorizer1.get_feature_names())

teach_pref_one_hot_train = vectorizer1.transform(project_data_train['teach_pref'].values)
teach_pref_one_hot_cv = vectorizer1.transform(project_data_cv['teach_pref'].values)
teach_pref_one_hot_test = vectorizer1.transform(project_data_test['teach_pref'].values)

print("After vectorizations")
print(teach_pref_one_hot_train.shape, project_data_y_train.shape)
print(teach_pref_one_hot_cv.shape, project_data_y_cv.shape)
print(teach_pref_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

In [52]:
```
vectorizer = TargetEncoder().fit(project_data_train['project_grad_cat'].values, project_data_y_train)
#print(vectorizer.get_feature_names())

project_grad_one_hot_train = vectorizer.transform(project_data_train['project_grad_cat'].values)  #this will
 change categorical data into binary form
project_grad_one_hot_cv = vectorizer.transform(project_data_cv['project_grad_cat'].values)  #this will change
categorical data into binary form
project_grad_one_hot_test = vectorizer.transform(project_data_test['project_grad_cat'].values)  #this will ch
ange categorical data into binary form

print("After vectorizations")
print(project_grad_one_hot_train.shape, project_data_y_train.shape)
print(project_grad_one_hot_cv.shape, project_data_y_cv.shape)
print(project_grad_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

In [53]:
```
vectorizer2 = TargetEncoder().fit(project_data_train['project_school_state'].values, project_data_y_train)
#print(vectorizer2.get_feature_names())

project_school_state_one_hot_train = vectorizer2.transform(project_data_train['project_school_state'].values)
#this will change categorical data into binary form
project_school_state_one_hot_cv = vectorizer2.transform(project_data_cv['project_school_state'].values)  #thi
s will change categorical data into binary form
project_school_state_one_hot_test = vectorizer2.transform(project_data_test['project_school_state'].values)
#this will change categorical data into binary form

print("After vectorizations")
print(project_school_state_one_hot_train.shape, project_data_y_train.shape)
print(project_school_state_one_hot_cv.shape, project_data_y_cv.shape)
print(project_school_state_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

In [54]:
```
vectorizer5 = TargetEncoder().fit(project_data_train['project_school_state'].values, project_data_y_train)
#print(vectorizer2.get_feature_names())

sub_categories_one_hot_train = vectorizer5.transform(project_data_train['project_school_state'].values)  #thi
s will change categorical data into binary form
sub_categories_one_hot_cv = vectorizer5.transform(project_data_cv['project_school_state'].values)  #this will
change categorical data into binary form
sub_categories_one_hot_test = vectorizer5.transform(project_data_test['project_school_state'].values)  #this
 will change categorical data into binary form

print("After vectorizations")
print(sub_categories_one_hot_train.shape, project_data_y_train.shape)
print(sub_categories_one_hot_cv.shape, project_data_y_cv.shape)
print(sub_categories_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

Encoding numerical features

In [55]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
num_title_standardized_train = num_title_scalar.transform(project_data_train['num_title'].values.reshape(-1,
1))
num_title_standardized_cv = num_title_scalar.transform(project_data_cv['num_title'].values.reshape(-1, 1))
num_title_standardized_test = num_title_scalar.transform(project_data_test['num_title'].values.reshape(-1, 1
))
```

```
/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

In [56]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
num_essay_standardized_train = num_title_scalar.transform(project_data_train['num_essay'].values.reshape(-1,
1))
num_essay_standardized_cv = num_title_scalar.transform(project_data_cv['num_essay'].values.reshape(-1, 1))
num_essay_standardized_test = num_title_scalar.transform(project_data_test['num_essay'].values.reshape(-1, 1
))
```

```
/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

In [57]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
quantity_standardized_train = quantity_scalar.transform(project_data_train['quantity'].values.reshape(-1, 1))
quantity_standardized_cv = quantity_scalar.transform(project_data_cv['quantity'].values.reshape(-1, 1))
quantity_standardized_test = quantity_scalar.transform(project_data_test['quantity'].values.reshape(-1, 1))
```

In [58]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(project_data_train['price'].values.reshape(-1, 1))
price_standardized_cv = price_scalar.transform(project_data_cv['price'].values.reshape(-1, 1))
price_standardized_test = price_scalar.transform(project_data_test['price'].values.reshape(-1, 1))
```

In [59]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
previous_standardized_train = previous_scalar.transform(project_data_train['teacher_number_of_previously_post
ed_projects'].values.reshape(-1, 1))
previous_standardized_cv = previous_scalar.transform(project_data_cv['teacher_number_of_previously_posted_pro
jects'].values.reshape(-1, 1))
previous_standardized_test = previous_scalar.transform(project_data_test['teacher_number_of_previously_posted
_projects'].values.reshape(-1, 1))
```

```
/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/utils/validation.py:595: DataConversio
nWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

In [60]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

neg_scalar = StandardScaler()
neg_scalar.fit(project_data_train['neg'].values.reshape(-1,1)) # finding the mean and standard deviation of t
his data
#print(f"Mean : {neg_scalar.mean_[0]}, Standard deviation : {np.sqrt(neg_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
neg_standardized_train = neg_scalar.transform(project_data_train['neg'].values.reshape(-1, 1))
neg_standardized_cv = neg_scalar.transform(project_data_cv['neg'].values.reshape(-1, 1))
neg_standardized_test = neg_scalar.transform(project_data_test['neg'].values.reshape(-1, 1))
```

In [61]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

pos_scalar = StandardScaler()
pos_scalar.fit(project_data_train['pos'].values.reshape(-1,1)) # finding the mean and standard deviation of t
his data
#print(f"Mean : {pos_scalar.mean_[0]}, Standard deviation : {np.sqrt(pos_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
pos_standardized_train = pos_scalar.transform(project_data_train['pos'].values.reshape(-1, 1))
pos_standardized_cv = pos_scalar.transform(project_data_cv['pos'].values.reshape(-1, 1))
pos_standardized_test = pos_scalar.transform(project_data_test['pos'].values.reshape(-1, 1))
```

In [62]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

neu_scalar = StandardScaler()
neu_scalar.fit(project_data_train['neg'].values.reshape(-1,1)) # finding the mean and standard deviation of t
his data
#print(f"Mean : {neu_scalar.mean_[0]}, Standard deviation : {np.sqrt(neu_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
neu_standardized_train = neu_scalar.transform(project_data_train['neu'].values.reshape(-1, 1))
neu_standardized_cv = neu_scalar.transform(project_data_cv['neu'].values.reshape(-1, 1))
neu_standardized_test = neu_scalar.transform(project_data_test['neu'].values.reshape(-1, 1))
```

In [63]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
aler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
compound_standardized_train = compound_scalar.transform(project_data_train['compound'].values.reshape(-1, 1))
compound_standardized_cv = compound_scalar.transform(project_data_cv['compound'].values.reshape(-1, 1))
compound_standardized_test = compound_scalar.transform(project_data_test['compound'].values.reshape(-1, 1))
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [64]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

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

Bow on title and essay

In [65]:
```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
vectorizer_bow = CountVectorizer(min_df=10)
vectorizer_bow = vectorizer_bow.fit(project_data_train['preprocessed_title'])  #this will convert word into n
dimensional vectors

title_bow_train = vectorizer_bow.transform(project_data_train['preprocessed_title'].values)
title_bow_cv = vectorizer_bow.transform(project_data_cv['preprocessed_title'].values)
title_bow_test = vectorizer_bow.transform(project_data_test['preprocessed_title'].values)

print("Shape of train matrix after one hot encodig ",title_bow_train.shape)
print("Shape of cv matrix after one hot encodig ",title_bow_cv.shape)
print("Shape of test matrix after one hot encodig ",title_bow_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 1217)
Shape of cv matrix after one hot encodig  (11055, 1217)
Shape of test matrix after one hot encodig  (16500, 1217)
```

In [66]:
```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
vectorizer_bow1 = CountVectorizer(min_df=10, ngram_range=(1,2), max_features=5000)
vectorizer_bow1 = vectorizer_bow1.fit(project_data_train['preprocessed_essay'])  #this will convert word into
n dimensional vectors

essay_bow_train = vectorizer_bow1.transform(project_data_train['preprocessed_essay'].values)
essay_bow_cv = vectorizer_bow1.transform(project_data_cv['preprocessed_essay'].values)
essay_bow_test = vectorizer_bow1.transform(project_data_test['preprocessed_essay'].values)

print("Shape of train matrix after one hot encodig ",essay_bow_train.shape)
print("Shape of cv matrix after one hot encodig ",essay_bow_cv.shape)
print("Shape of test matrix after one hot encodig ",essay_bow_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 5000)
Shape of cv matrix after one hot encodig  (11055, 5000)
Shape of test matrix after one hot encodig  (16500, 5000)
```

tdidf on essay and title

In [67]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf = TfidfVectorizer(min_df=10)
vectorizer_tfidf = vectorizer_tfidf.fit(project_data_train['preprocessed_title'])

title_tfidf_train = vectorizer_tfidf.transform(project_data_train['preprocessed_title'].values)
title_tfidf_cv = vectorizer_tfidf.transform(project_data_cv['preprocessed_title'].values)
title_tfidf_test = vectorizer_tfidf.transform(project_data_test['preprocessed_title'].values)

print("Shape of train matrix after one hot encodig ",title_tfidf_train.shape)
print("Shape of cv matrix after one hot encodig ",title_tfidf_cv.shape)
print("Shape of test matrix after one hot encodig ",title_tfidf_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 1217)
Shape of cv matrix after one hot encodig  (11055, 1217)
Shape of test matrix after one hot encodig  (16500, 1217)
```

In [68]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf1 = TfidfVectorizer(min_df=10, ngram_range=(1,2), max_features=5000)
vectorizer_tfidf1 = vectorizer_tfidf1.fit(project_data_train['preprocessed_essay'])


essay_tfidf_train = vectorizer_tfidf1.transform(project_data_train['preprocessed_essay'].values)
essay_tfidf_cv = vectorizer_tfidf1.transform(project_data_cv['preprocessed_essay'].values)
essay_tfidf_test = vectorizer_tfidf1.transform(project_data_test['preprocessed_essay'].values)

print("Shape of train matrix after one hot encodig ",essay_bow_train.shape)
print("Shape of cv matrix after one hot encodig ",essay_bow_cv.shape)
print("Shape of test matrix after one hot encodig ",essay_bow_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 5000)
Shape of cv matrix after one hot encodig  (11055, 5000)
Shape of test matrix after one hot encodig  (16500, 5000)
```

Below codes are taken from Applied Ai course and modified according to my use

In [69]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load
-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [70]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
100%|██████████| 22445/22445 [00:07<00:00, 2995.93it/s]
  3%|▎         | 277/11055 [00:00<00:03, 2769.32it/s]

22445
300

100%|██████████| 11055/11055 [00:03<00:00, 2972.85it/s]
  2%|▏         | 289/16500 [00:00<00:05, 2883.28it/s]

11055
300

100%|██████████| 16500/16500 [00:05<00:00, 3204.88it/s]

16500
300
```

In [71]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train1.append(vector)

print(len(avg_w2v_vectors_train1))
print(len(avg_w2v_vectors_train1[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_cv1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv1.append(vector)

print(len(avg_w2v_vectors_cv1))
print(len(avg_w2v_vectors_cv1[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_test1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test1.append(vector)

print(len(avg_w2v_vectors_test1))
print(len(avg_w2v_vectors_test1[0]))
```

```
100%|██████████| 22445/22445 [00:00<00:00, 58182.68it/s]
100%|██████████| 11055/11055 [00:00<00:00, 58046.60it/s]
  0%|          | 0/16500 [00:00<?, ?it/s]

22445
300
11055
300

100%|██████████| 16500/16500 [00:00<00:00, 57822.11it/s]

16500
300
```

In [72]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_essay'])
# 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())

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_essay']): # 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 value((sentence.count(word)/len
(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
 for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essay']): # 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 value((sentence.count(word)/len
(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
 for each word
```

```
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test= []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essay']): # 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 value((sentence.count(word)/len
(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
 for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
100%|██████████| 22445/22445 [00:45<00:00, 494.56it/s]
  4%|▍         | 490/11055 [00:01<00:21, 484.31it/s]

22445
300

100%|██████████| 11055/11055 [00:22<00:00, 488.80it/s]
  1%|          | 103/16500 [00:00<00:31, 517.10it/s]

11055
300

100%|██████████| 16500/16500 [00:33<00:00, 492.81it/s]

16500
300
```

In [73]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model_1 = TfidfVectorizer()
tfidf_model_1.fit(project_data_train['preprocessed_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_1.get_feature_names(), list(tfidf_model_1.idf_)))
tfidf_words = set(tfidf_model_1.get_feature_names())

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_title']): # 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 value((sentence.count(word)/len
(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
 for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_train1.append(vector)

print(len(tfidf_w2v_vectors_train1))
print(len(tfidf_w2v_vectors_train1[0]))

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_title']): # 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 value((sentence.count(word)/len
(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
 for each word
```

```
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
                tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_cv1.append(vector)

print(len(tfidf_w2v_vectors_cv1))
print(len(tfidf_w2v_vectors_cv1[0]))

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test1 = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_title']): # 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 value((sentence.count(word)/len
(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
 for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_test1.append(vector)

print(len(tfidf_w2v_vectors_test1))
print(len(tfidf_w2v_vectors_test1[0]))
```

```
100%|██████████| 22445/22445 [00:00<00:00, 23741.27it/s]
 22%|██        | 2404/11055 [00:00<00:00, 24035.61it/s]

22445
300

100%|██████████| 11055/11055 [00:00<00:00, 24684.96it/s]
 30%|███       | 4941/16500 [00:00<00:00, 24701.56it/s]

11055
300

100%|██████████| 16500/16500 [00:00<00:00, 24595.94it/s]

16500
300
```

In [74]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
lr_train_1=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, title_bow_train, essay_bow_train)).tocsr()
lr_cv_1=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, title_bow_cv, essay_bow_cv)).tocsr()
lr_test_1=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, title_bow_test,essay_bow_test)).tocsr()
```

In [75]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
lr_train_2=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, title_tfidf_train, essay_tfidf_train)).tocsr()
lr_cv_2=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, title_tfidf_cv, essay_tfidf_cv)).tocsr()
lr_test_2=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, title_tfidf_test,essay_tfidf_test)).tocsr()
```

In [76]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
lr_train_3=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, avg_w2v_vectors_train1, avg_w2v_vectors_train)).tocsr()
lr_cv_3=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, avg_w2v_vectors_cv1, avg_w2v_vectors_cv)).tocsr()
lr_test_3=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, avg_w2v_vectors_test1,avg_w2v_vectors_test)).tocsr()
```

In [77]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
lr_train_4=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, tfidf_w2v_vectors_train1, tfidf_w2v_vectors_train)).tocsr()
lr_cv_4=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, tfidf_w2v_vectors_cv1, tfidf_w2v_vectors_cv)).tocsr()
lr_test_4=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, tfidf_w2v_vectors_test1,tfidf_w2v_vectors_test)).tocsr()
```

In [ ]:

2.4 Applying Random Forest

Apply Random Forest on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

2.4.1 Applying Random Forests on BOW, SET 1

In [78]:
```
# Please write all the code with proper documentation
```

Plot_confusion_matrix is taken from Facebook friend recommendatation https://www.appliedaicourse.com/

In [84]:
```
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)

    A =(((C.T)/(C.sum(axis=1))).T)

    B =(C/C.sum(axis=0))
    plt.figure(figsize=(20,4))

    labels = [0,1]
    # representing A in heatmap format
    cmap=sns.light_palette("Navy", as_cmap=True)#https://stackoverflow.com/questions/37902459/seaborn-color-p
alette-as-matplotlib-colormap
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")
    plt.show()
```

In [85]:
```
def Zero1(prediction):
    predicted=[]
    for i in prediction:
        if i<0.5:
            predicted.append(0)
        else:
            predicted.append(1)
    return predicted
```

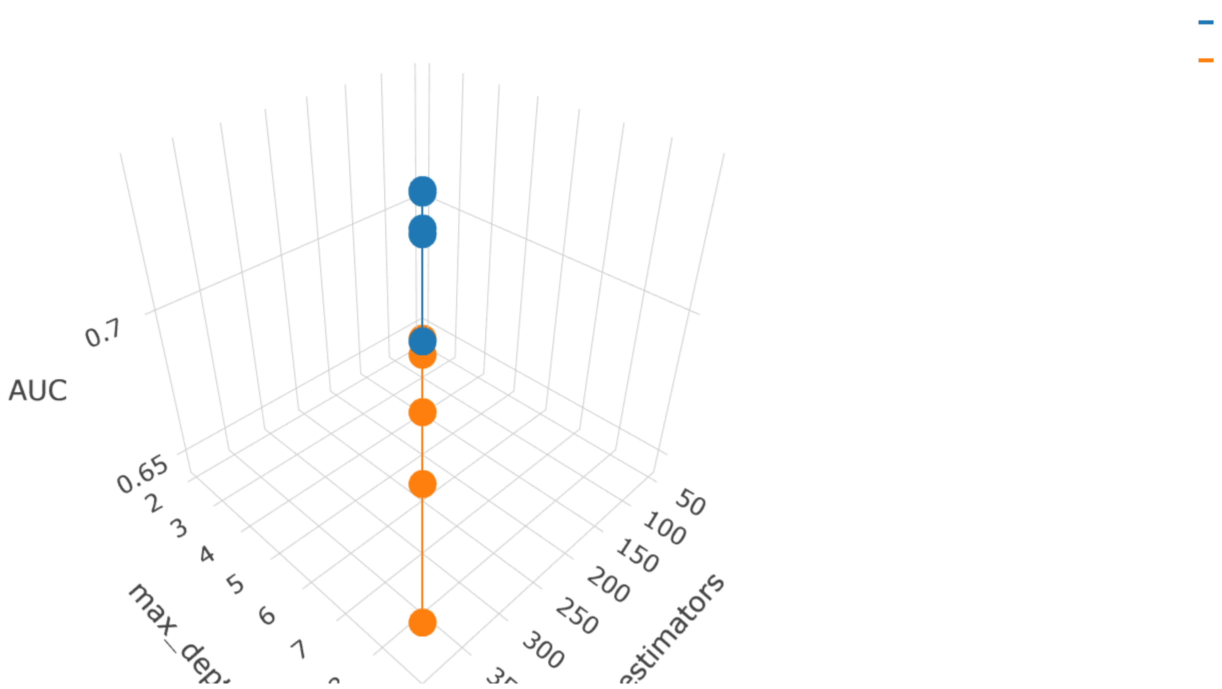
In [81]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

RandomF = RandomForestClassifier(max_features='sqrt', n_jobs=-1)
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(RandomF, parameters, cv=3, scoring='roc_auc')
clf.fit(lr_train_1, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=9, max_features='sqrt', 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=300, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

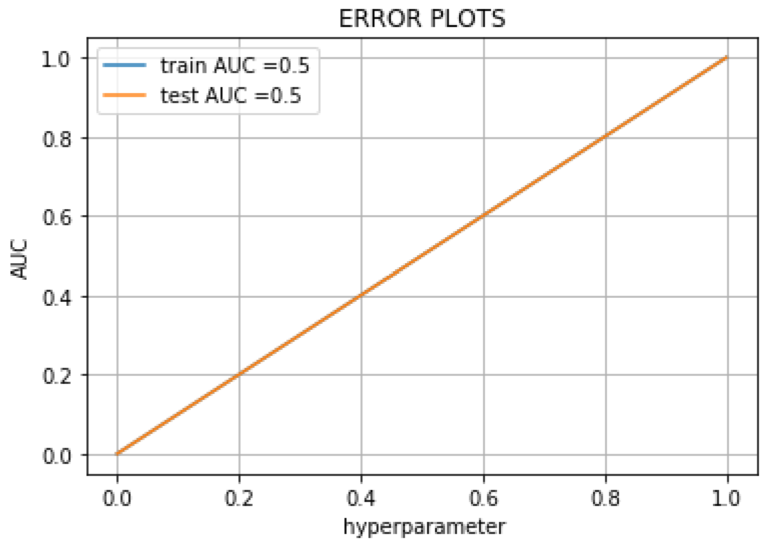
In [82]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [83]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [84]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

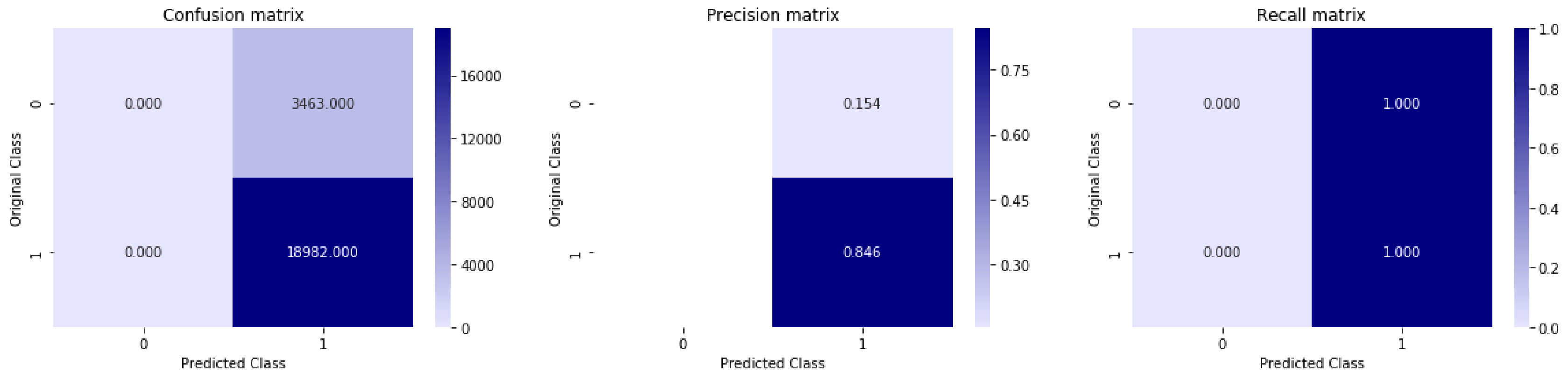

AUC
0.7
0.65
2
3
4
5
6
7
max_dept
50
100
150
200
250
300
estimators

In [85]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

RandomF = RandomForestClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_depth, n_jo
bs=-1)
RandomF.fit(lr_train_1, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = RandomF.predict(lr_train_1)
y_test_pred = RandomF.predict(lr_test_1)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

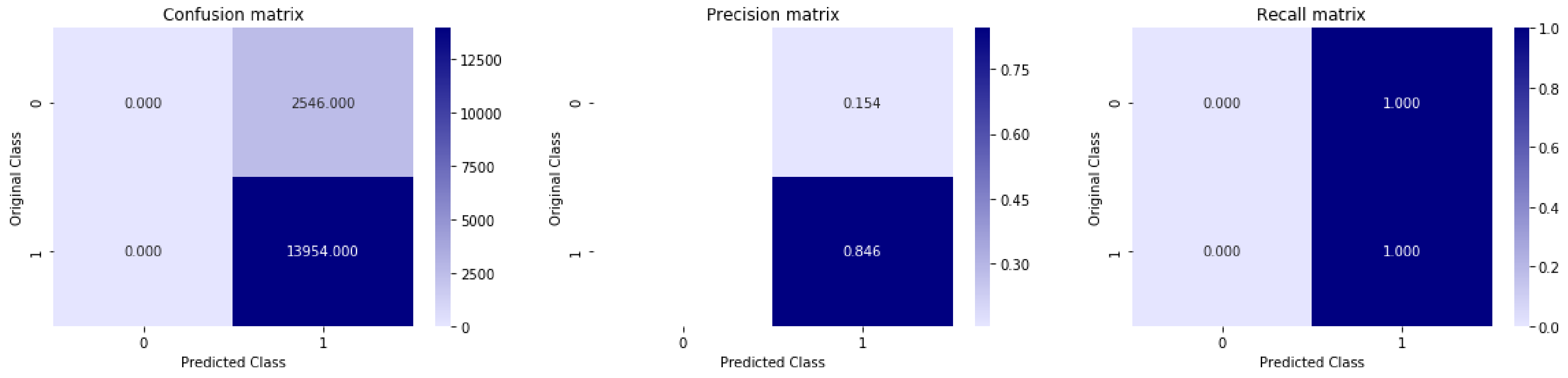


In [86]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



In [ ]:

2.4.2 Applying Random Forests on TFIDF, SET 2

In [87]:
```
# Please write all the code with proper documentation
```

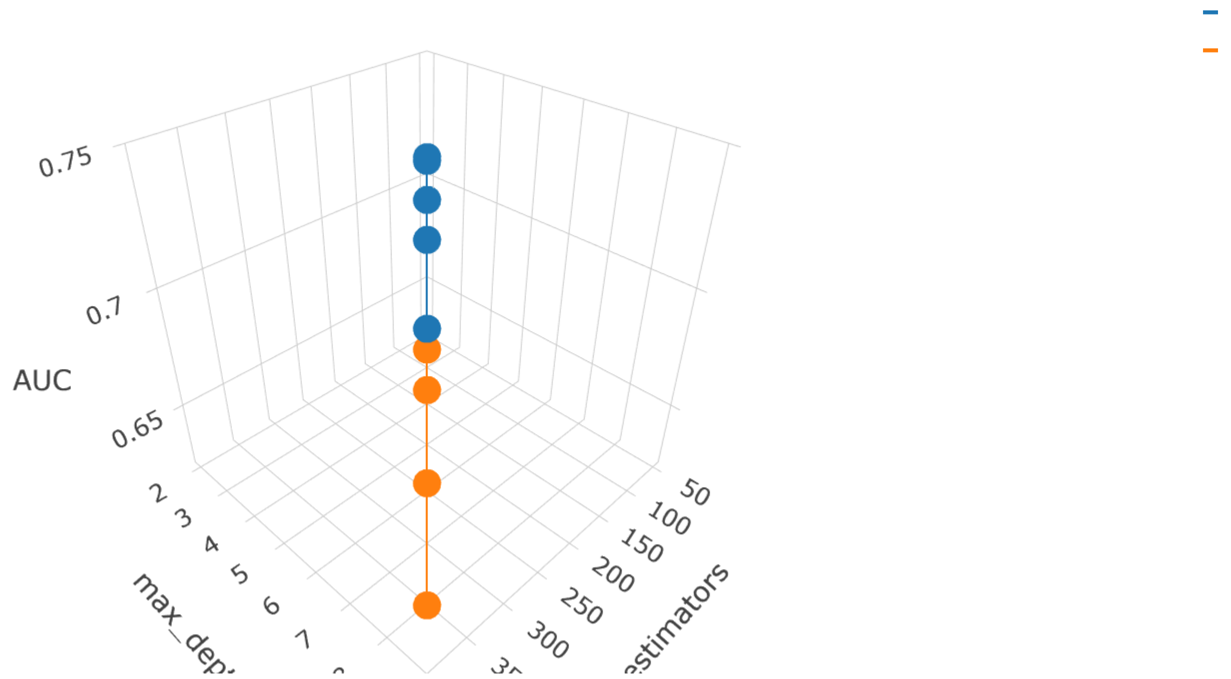
In [88]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

RandomF = RandomForestClassifier(max_features='sqrt', n_jobs=-1)
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(RandomF, parameters, cv=3, scoring='roc_auc')
clf.fit(lr_train_2, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=9, max_features='sqrt', 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=400, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

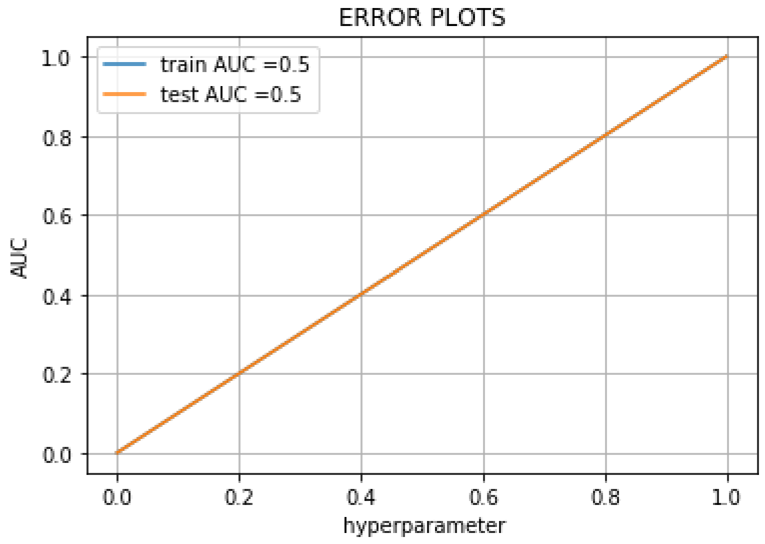
In [89]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [90]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [91]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

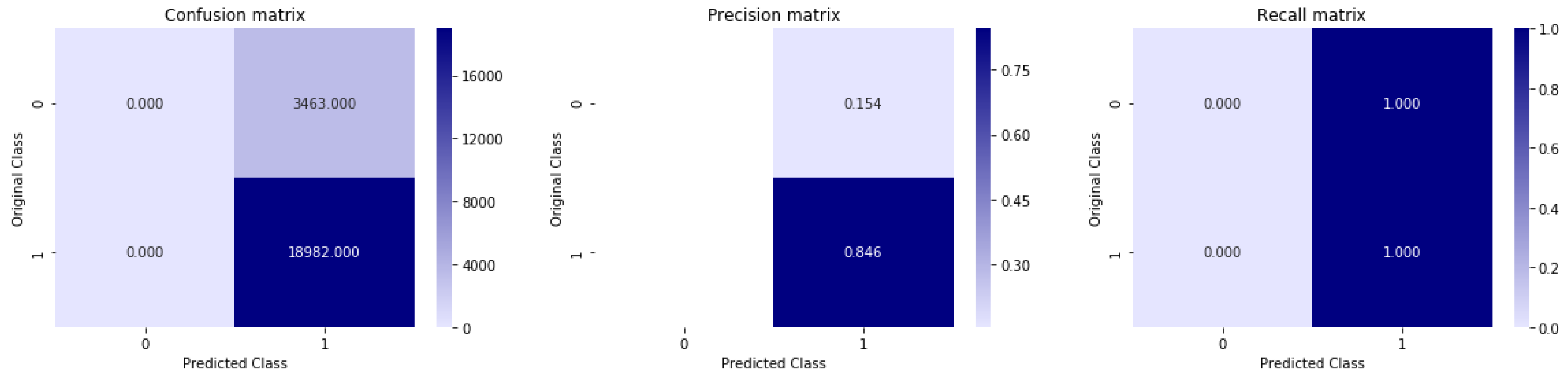

0.75
0.7
AUC
0.65
2
3
4
5
6
7
max_dep
50
100
150
200
250
300
estimators

In [92]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

RandomF = RandomForestClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_depth, n_jo
bs=-1)
RandomF.fit(lr_train_2, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = RandomF.predict(lr_train_2)
y_test_pred = RandomF.predict(lr_test_2)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

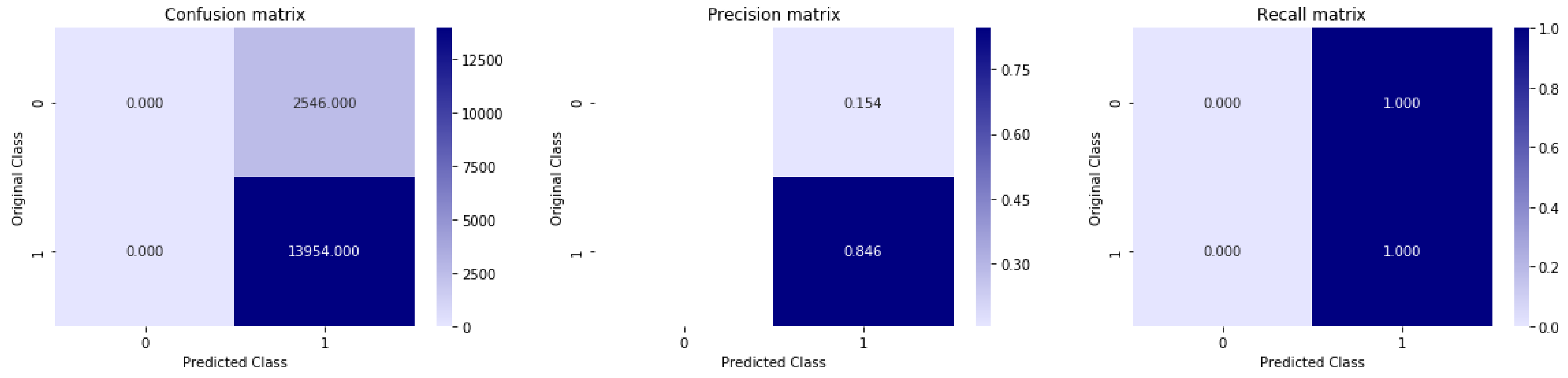


In [93]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



2.4.3 Applying Random Forests on AVG W2V, SET 3

In [94]:
```
# Please write all the code with proper documentation
```

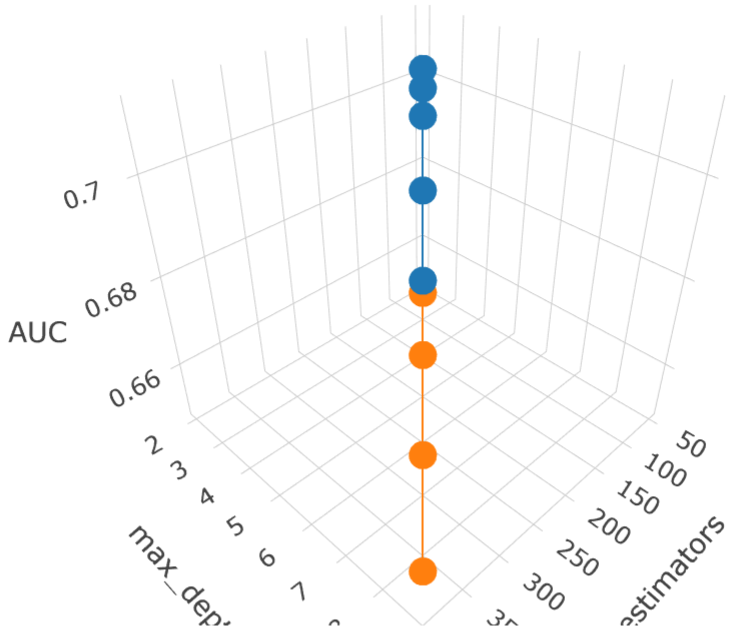
In [95]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

RandomF = RandomForestClassifier(max_features='sqrt', n_jobs=-1)
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(RandomF, parameters, cv=3, scoring='roc_auc')
clf.fit(lr_train_3, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=5, max_features='sqrt', 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=400, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

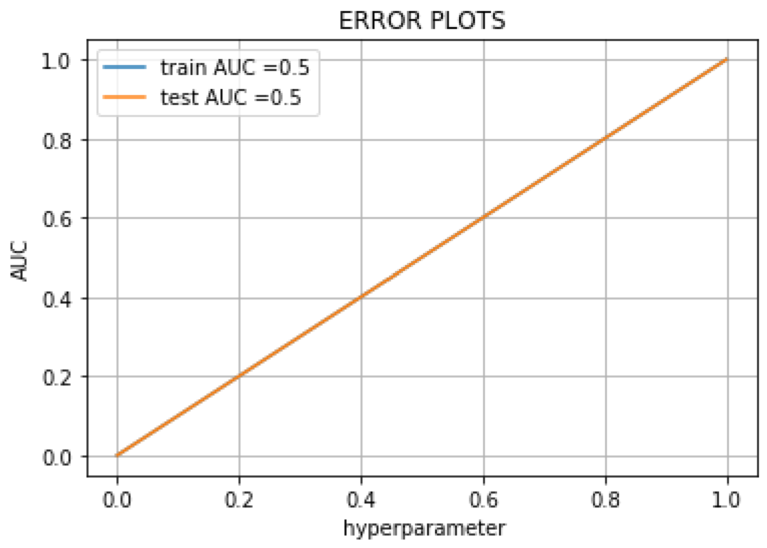
In [96]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [97]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [98]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

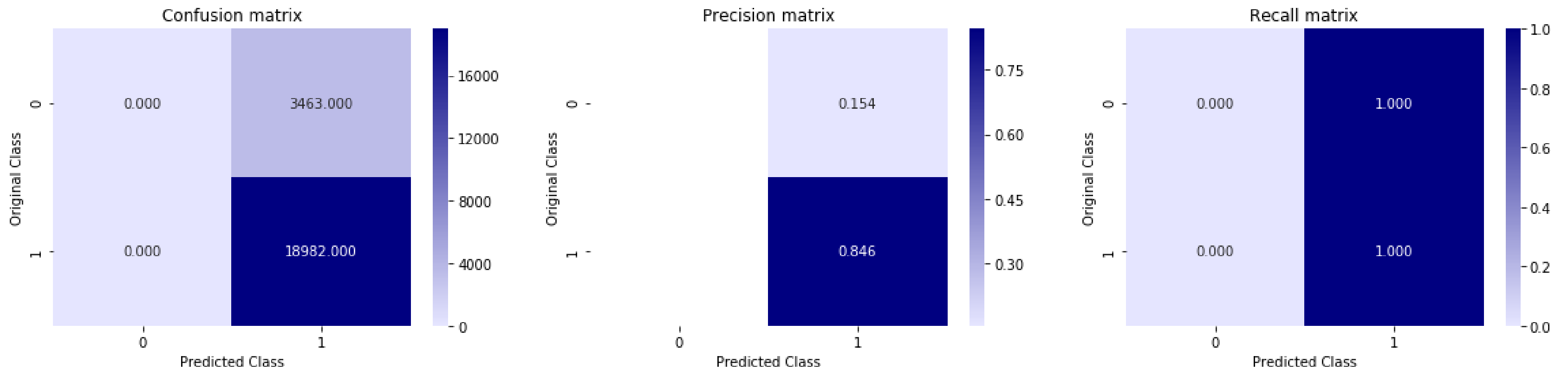

AUC
0.7
0.68
0.66
2
3
4
5
6
7
max_dept
50
100
150
200
250
300
estimators

In [99]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
from sklearn.metrics import roc_curve, auc

RandomF = RandomForestClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_depth, n_jo
bs=-1)
RandomF.fit(lr_train_3, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = RandomF.predict(lr_train_3)
y_test_pred = RandomF.predict(lr_test_3)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

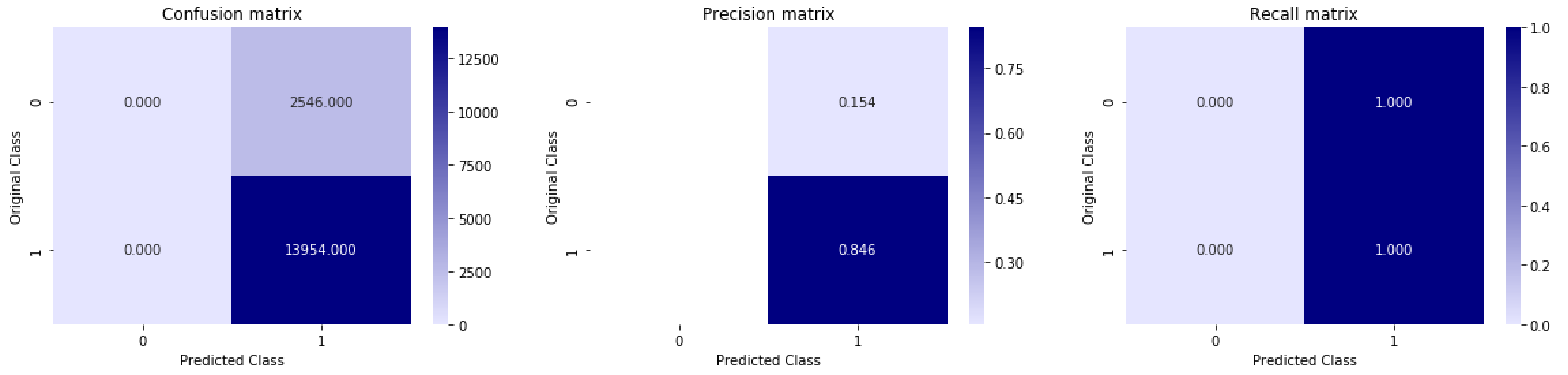


In [100]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



2.4.4 Applying Random Forests on TFIDF W2V, SET 4

In [101]:
```
# Please write all the code with proper documentation
```

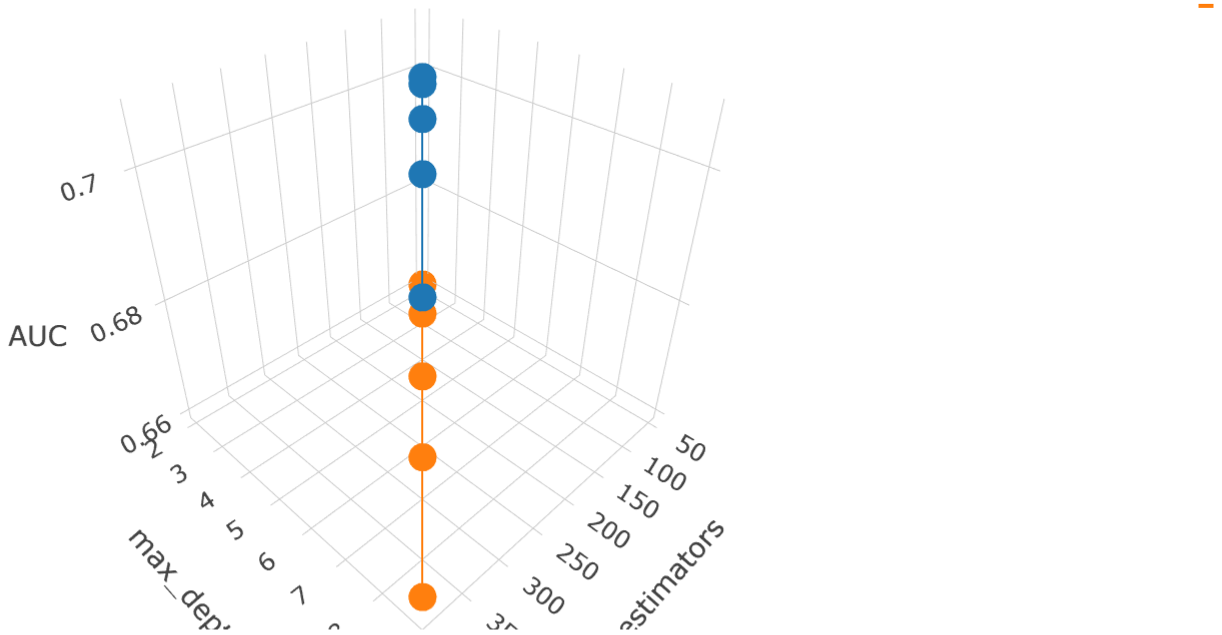
In [102]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

RandomF = RandomForestClassifier(max_features='sqrt', n_jobs=-1)
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(RandomF, parameters, cv=3, scoring='roc_auc')
clf.fit(lr_train_4, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=5, max_features='sqrt', 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=400, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

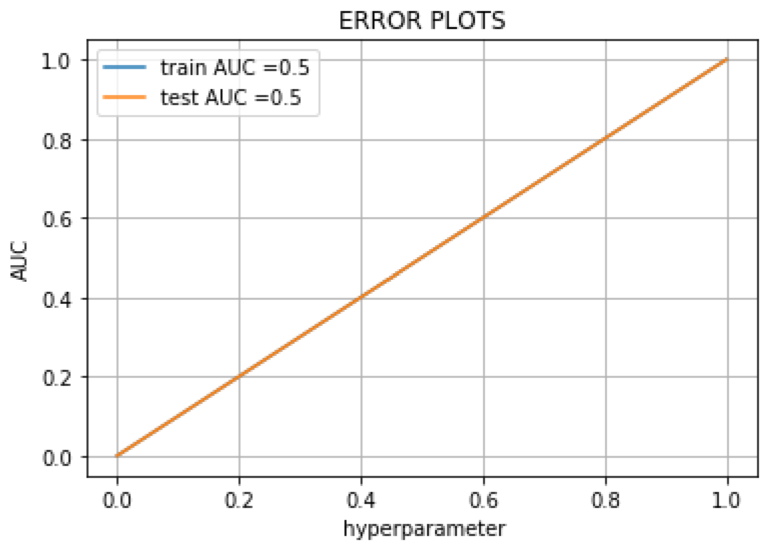
In [103]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [104]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [105]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

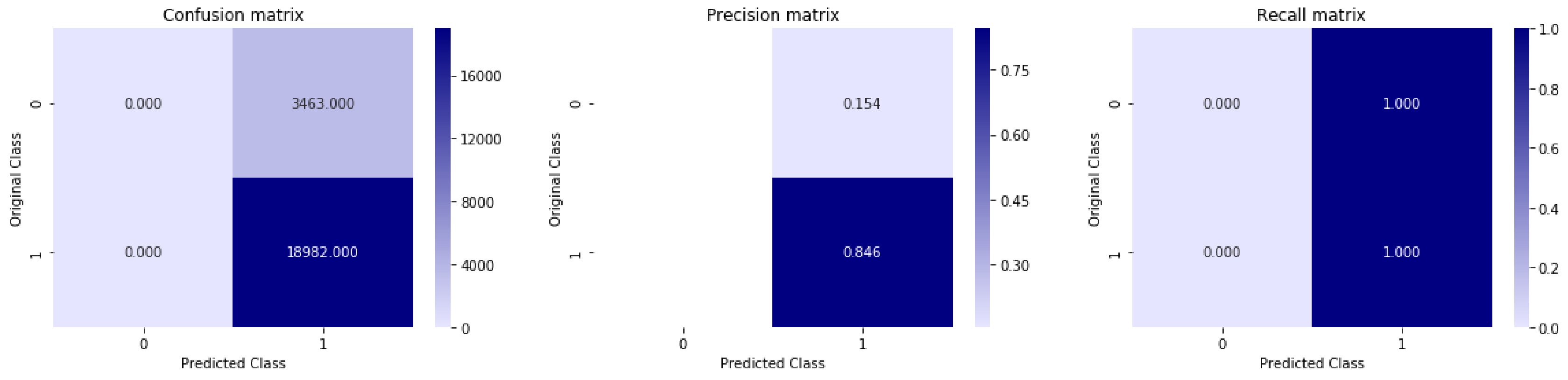

AUC
0.7
0.68
0.66
max_dept
2
3
4
5
6
7
estimators
50
100
150
200
250
300

In [106]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

RandomF = RandomForestClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_depth, n_jo
bs=-1)
RandomF.fit(lr_train_4, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = RandomF.predict(lr_train_4)
y_test_pred = RandomF.predict(lr_test_4)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

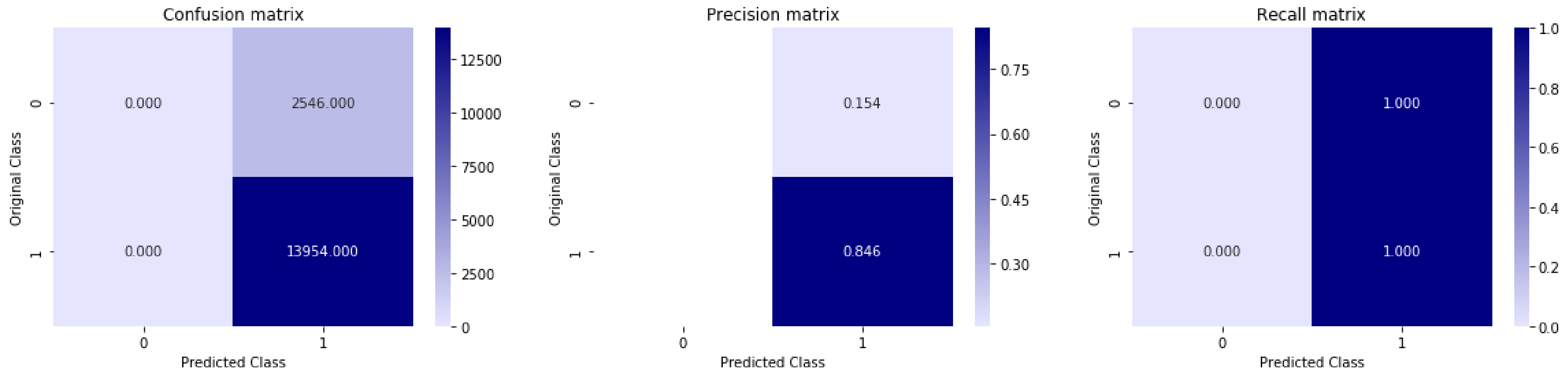


In [107]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



2.5 Applying GBDT

Apply GBDT on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

2.5.1 Applying XGBOOST on BOW, SET 1

In [108]:
```
# Please write all the code with proper documentation
```

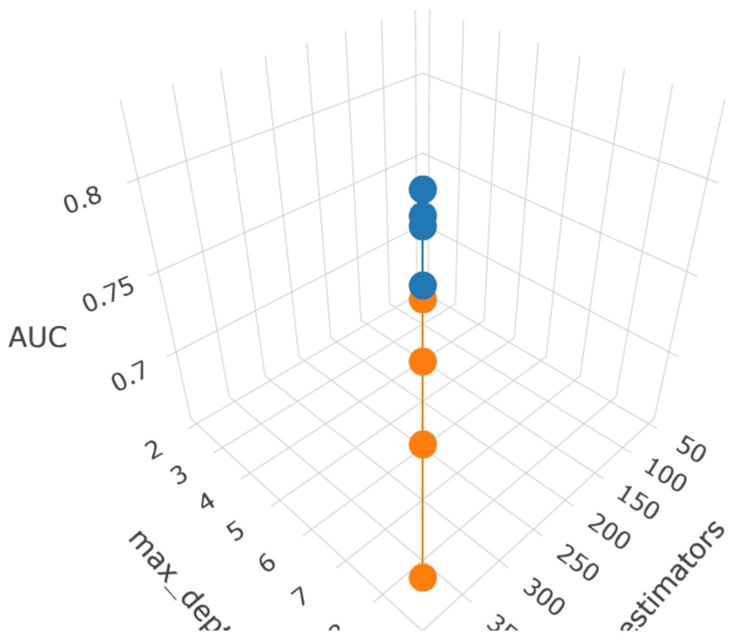
In [78]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

Grad_Boost = GradientBoostingClassifier(max_features='sqrt')
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(Grad_Boost, parameters, cv=3, scoring='roc_auc', n_jobs=-1)
clf.fit(lr_train_1, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
/home/navneetkumarbitsindri/.local/lib/python3.5/site-packages/sklearn/externals/joblib/externals/loky/proces
s_executor.py:706: UserWarning:

A worker stopped while some jobs were given to the executor. This can be caused by a too short worker timeout
or by a memory leak.


Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features='sqrt', 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=400,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

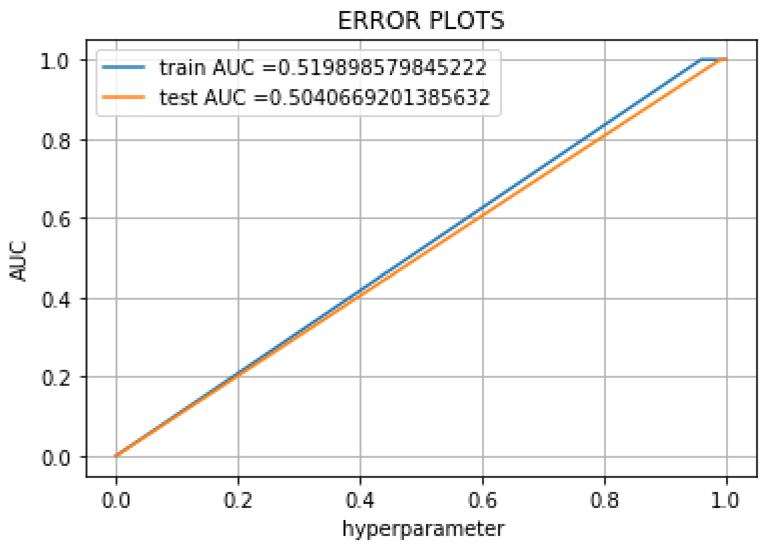
In [79]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [80]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [81]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

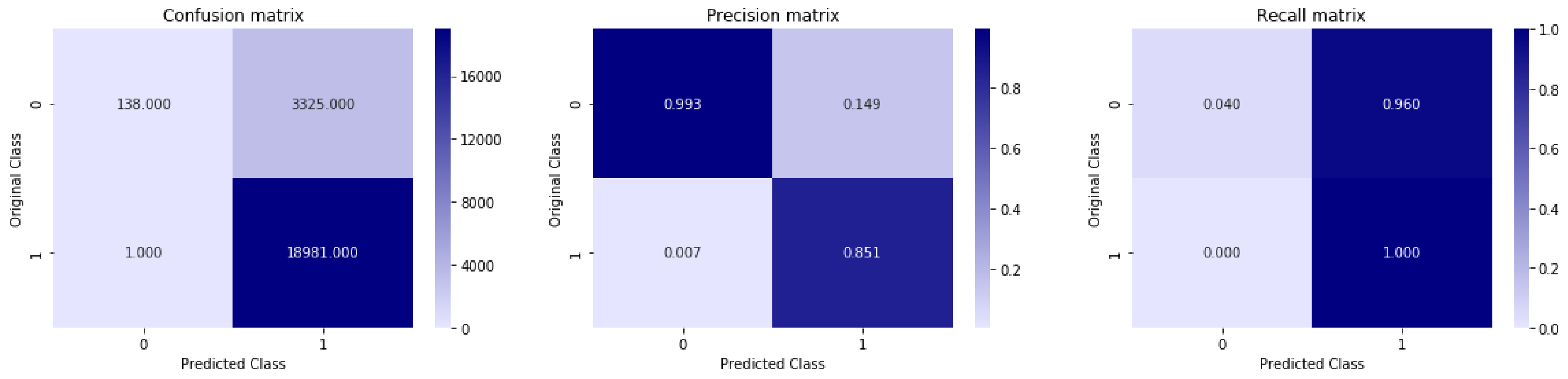

AUC
0.8
0.75
0.7
max_dept
2
3
4
5
6
7
50
100
150
200
250
300
estimators

In [82]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
from sklearn.metrics import roc_curve, auc

Grad_Boost = GradientBoostingClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_dept
h)
Grad_Boost.fit(lr_train_1, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = Grad_Boost.predict(lr_train_1)
y_test_pred = Grad_Boost.predict(lr_test_1)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

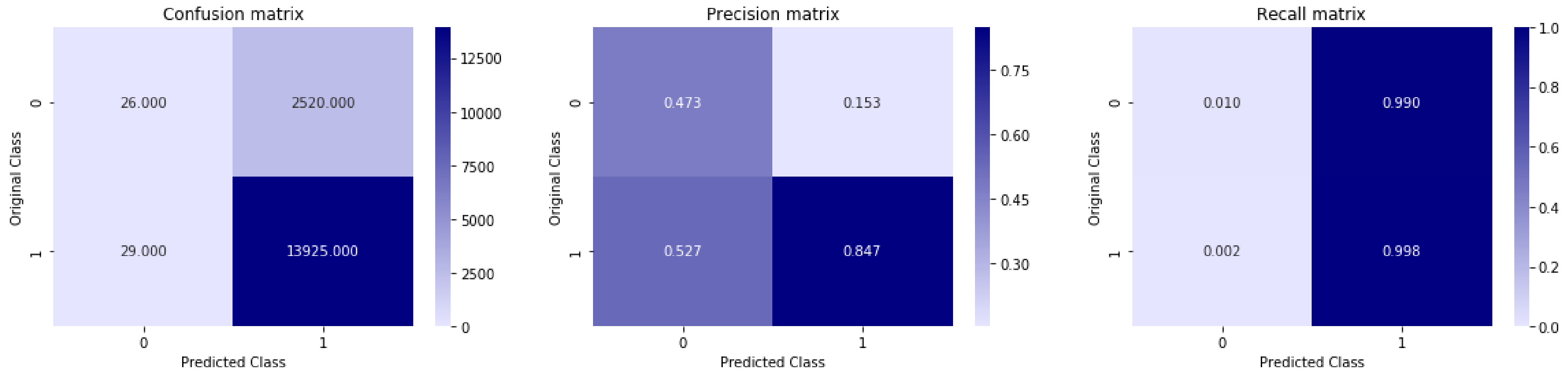


In [86]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



2.5.2 Applying XGBOOST on TFIDF, SET 2

In [87]:
```
# Please write all the code with proper documentation
```

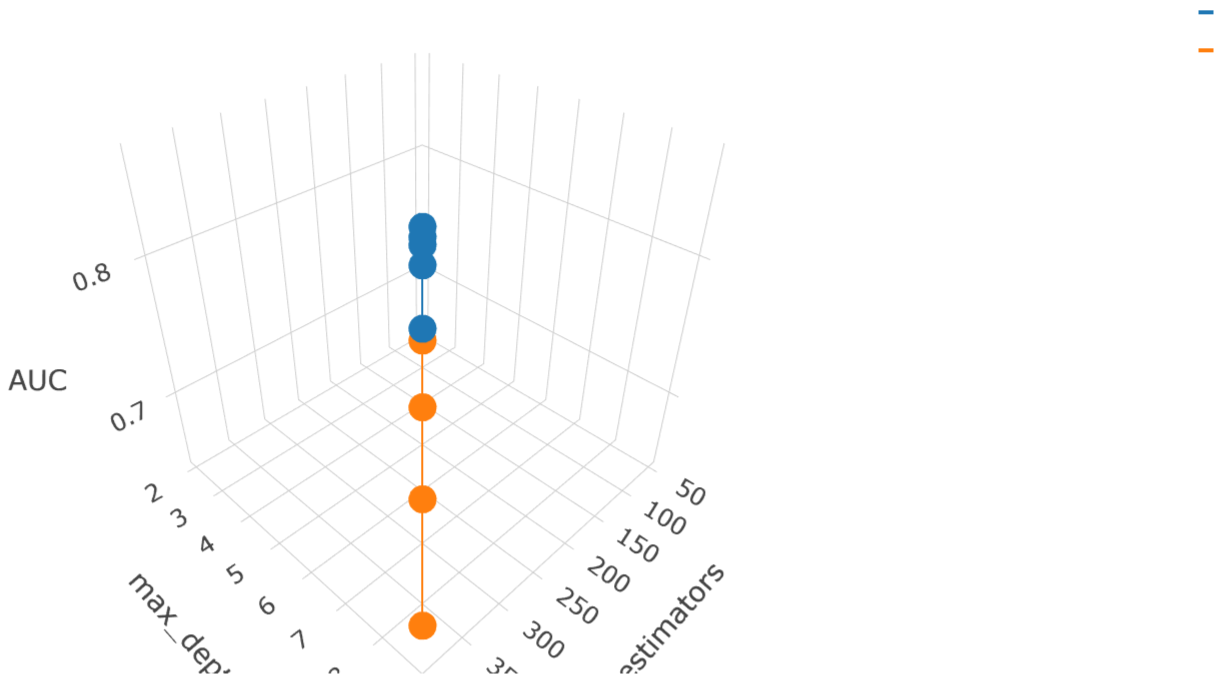
In [88]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

Grad_Boost = GradientBoostingClassifier(max_features='sqrt')
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(Grad_Boost, parameters, cv=3, scoring='roc_auc', n_jobs=-1)
clf.fit(lr_train_2, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=2,
              max_features='sqrt', 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=400,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

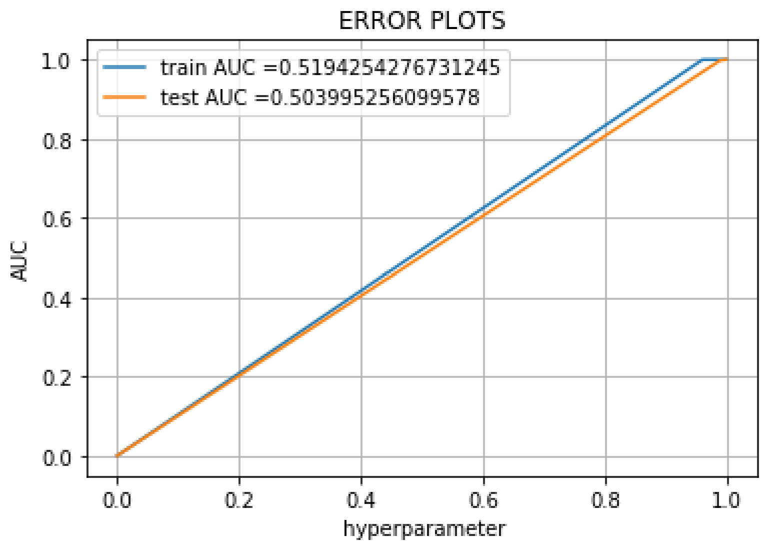
In [89]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [90]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [91]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

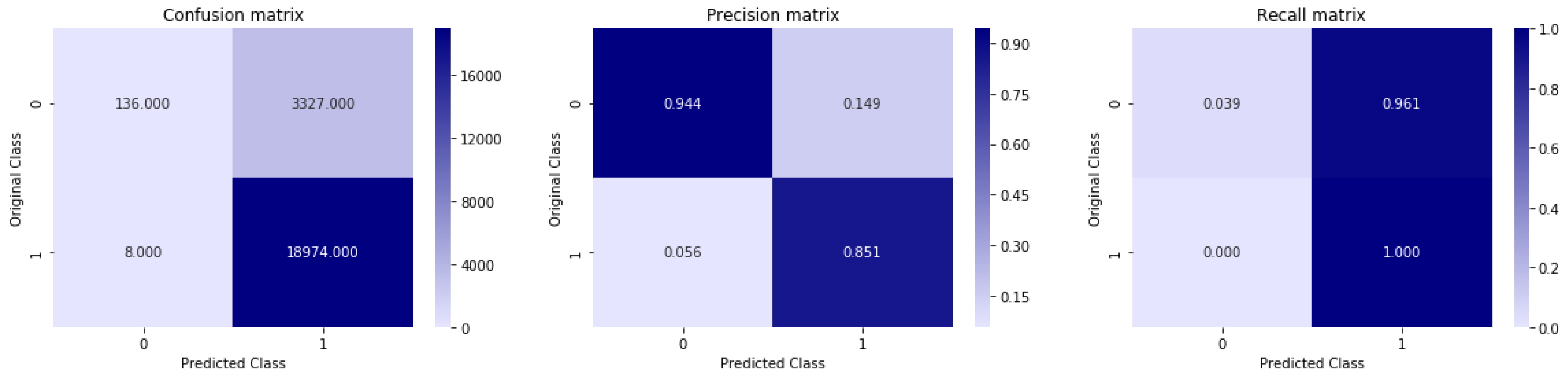

AUC
0.8
0.7
2
3
4
5
6
7
max_dep
50
100
150
200
250
300
estimators

In [92]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

Grad_Boost = GradientBoostingClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_dept
h)
Grad_Boost.fit(lr_train_2, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = Grad_Boost.predict(lr_train_2)
y_test_pred = Grad_Boost.predict(lr_test_2)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

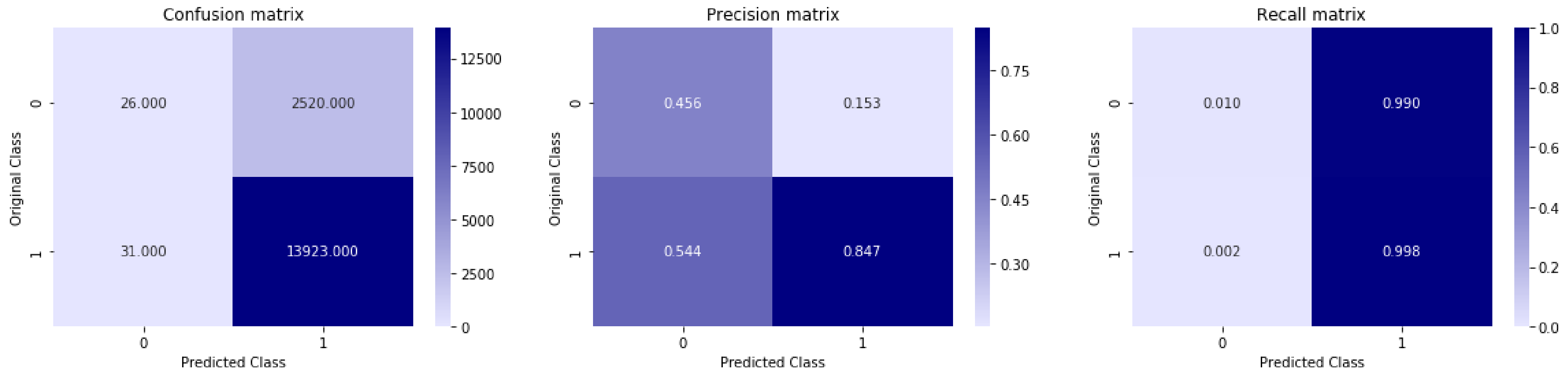


```
In [93]: print('Train confusion_matrix')
         plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
         print('Test confusion_matrix')
         plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



```
In [ ]:
```

2.5.3 Applying XGBOOST on AVG W2V, SET 3

In [94]:
```
# Please write all the code with proper documentation
```

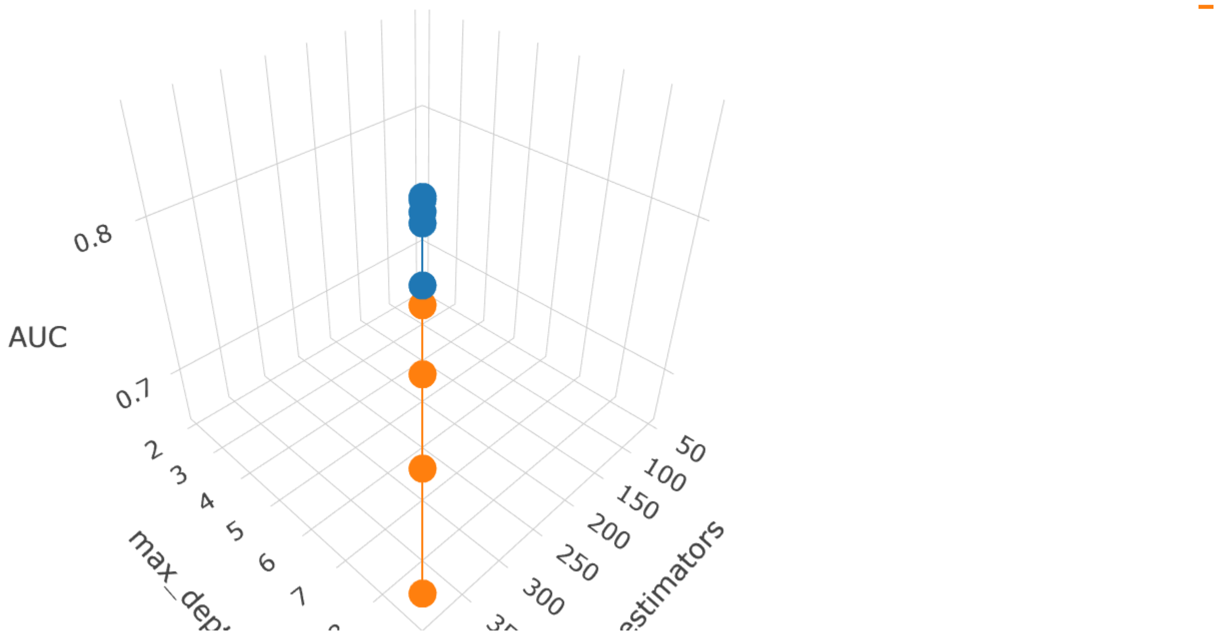
In [95]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

Grad_Boost = GradientBoostingClassifier(max_features='sqrt')
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(Grad_Boost, parameters, cv=3, scoring='roc_auc', n_jobs=-1)
clf.fit(lr_train_3, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=2,
              max_features='sqrt', 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=400,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

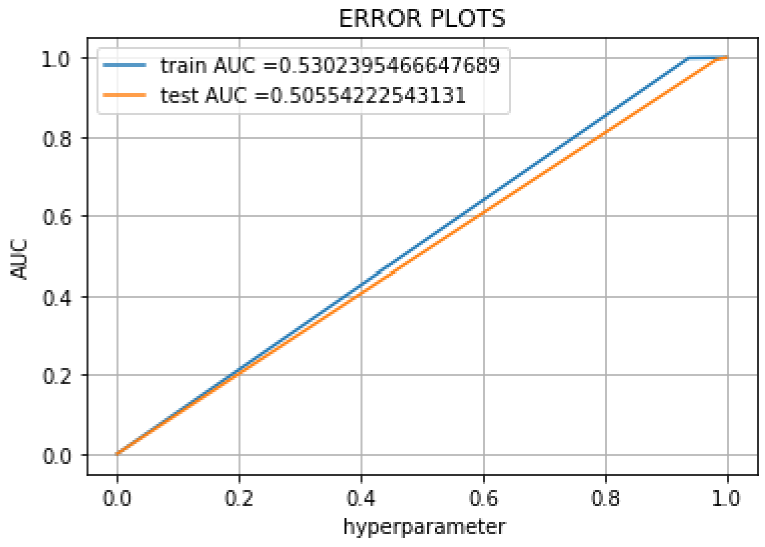
In [96]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [97]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [98]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

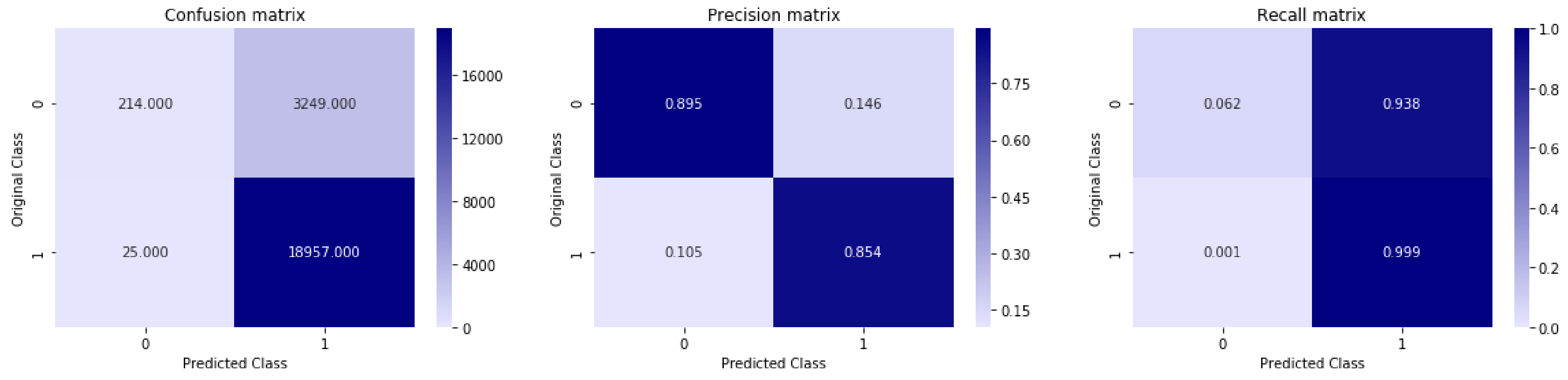

0.8
0.7
AUC
2
3
4
5
6
7
max_dep
50
100
150
200
250
300
estimators

In [99]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

Grad_Boost = GradientBoostingClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_dept
h)
Grad_Boost.fit(lr_train_3, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = Grad_Boost.predict(lr_train_3)
y_test_pred = Grad_Boost.predict(lr_test_3)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

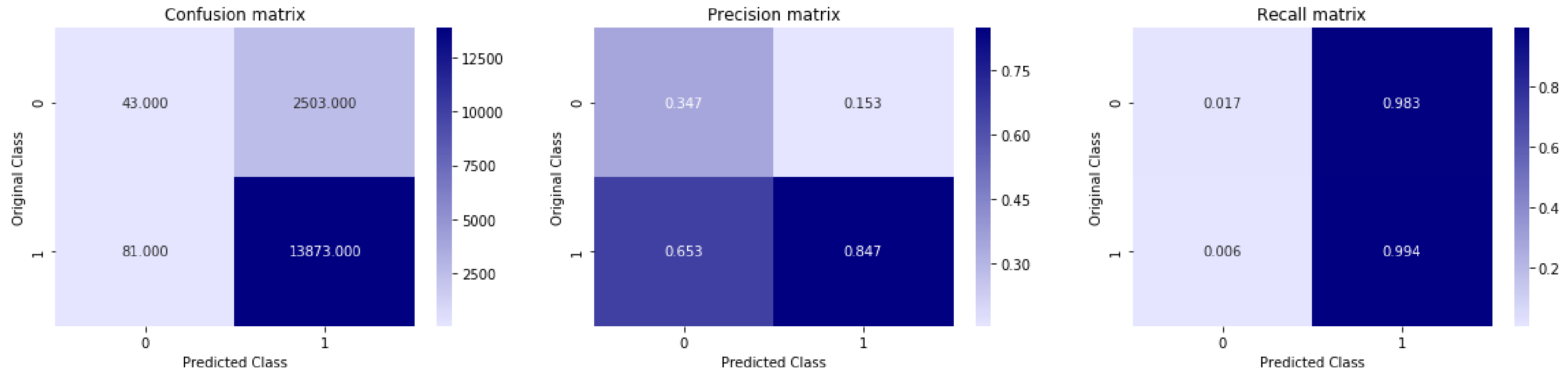


In [100]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



2.5.4 Applying XGBOOST on TFIDF W2V, SET 4

In [101]:
```
# Please write all the code with proper documentation
```

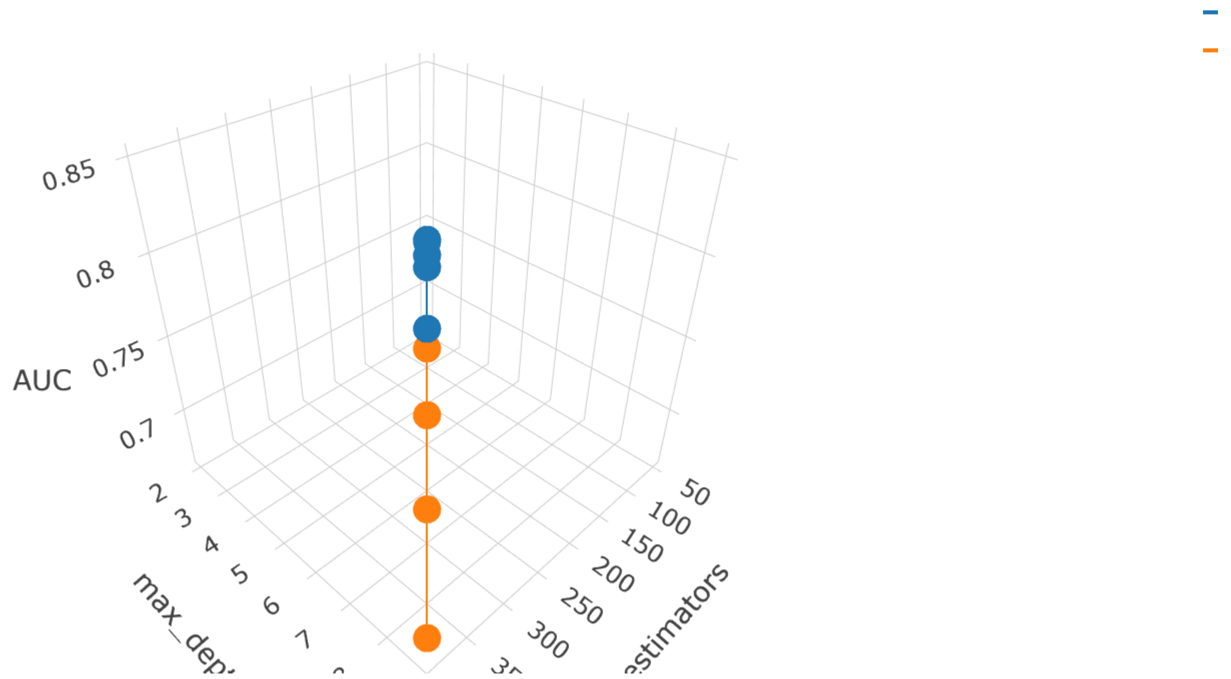
In [102]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

Grad_Boost = GradientBoostingClassifier(max_features='sqrt')
n_estimators=[50, 100, 200, 300, 400]
max_depth=[2, 3, 5, 7, 9]
parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(Grad_Boost, parameters, cv=3, scoring='roc_auc', n_jobs=-1)
clf.fit(lr_train_4, project_data_y_train)

print("Model with best parameters :\n",clf.best_estimator_)

train_auc= list(clf.cv_results_['mean_train_score'])
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = list(clf.cv_results_['mean_test_score'])
cv_auc_std= clf.cv_results_['std_test_score']

best_depth = clf.best_estimator_.max_depth
best_estimator = clf.best_estimator_.n_estimators
```

```
Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=2,
              max_features='sqrt', 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=400,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

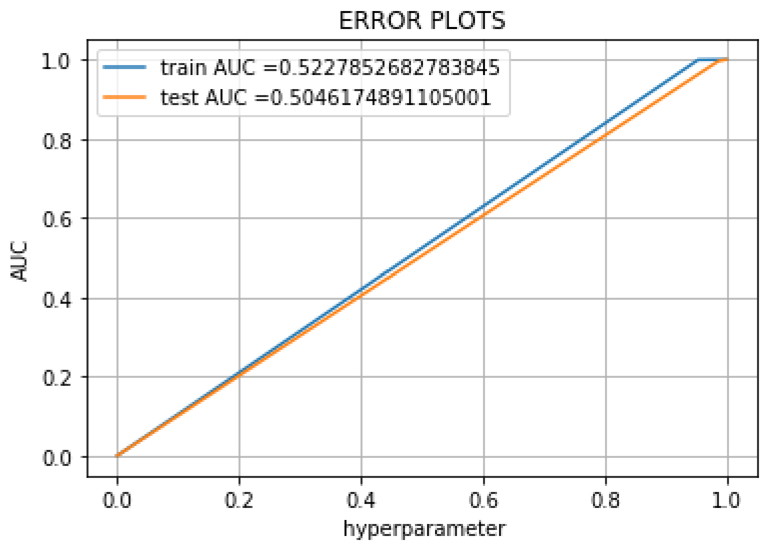
In [103]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [104]:
```
x1 = n_estimators
y1 = max_depth
z1 = train_auc

x2 = n_estimators
y2 = max_depth
z2 = cv_auc
```

In [105]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

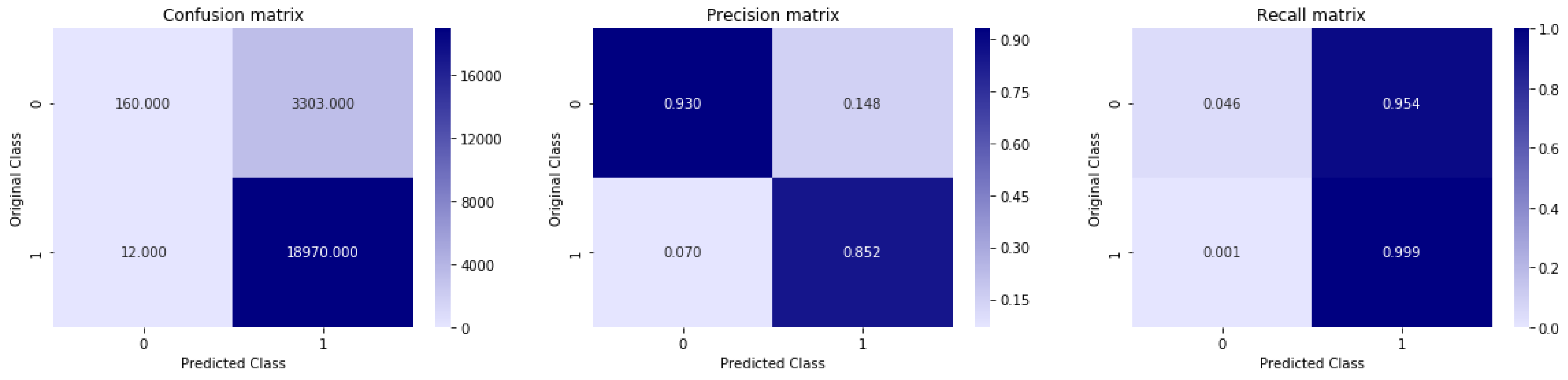

AUC
0.85
0.8
0.75
0.7
2
3
4
5
6
7
max_dept
50
100
150
200
250
300
estimators

In [106]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

Grad_Boost = GradientBoostingClassifier(max_features='sqrt', n_estimators=best_estimator, max_depth=best_dept
h)
Grad_Boost.fit(lr_train_4, project_data_y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = Grad_Boost.predict(lr_train_4)
y_test_pred = Grad_Boost.predict(lr_test_4)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

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

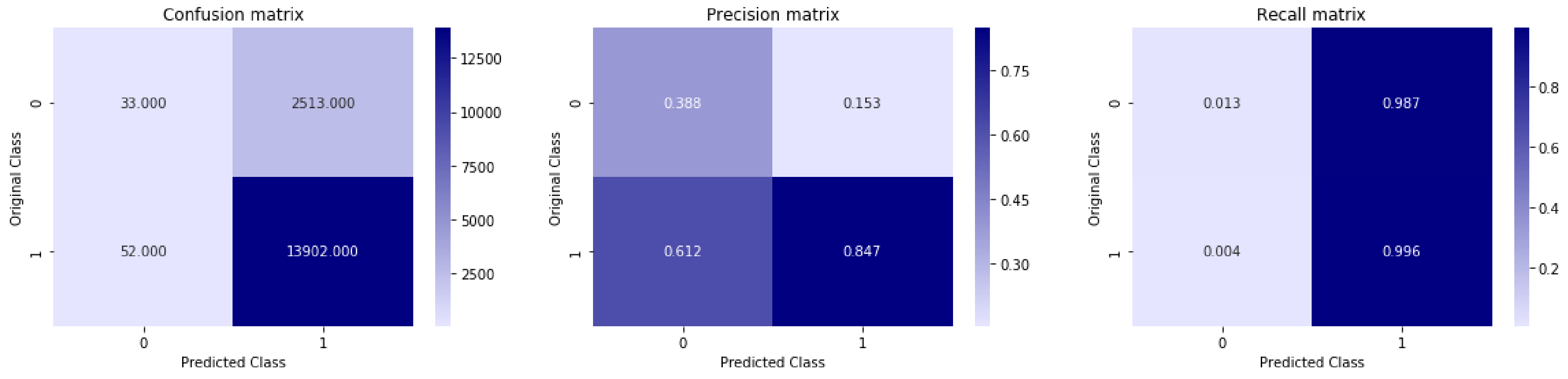


In [107]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



```
Test confusion_matrix
```



3. Conclusion

In [ ]:
```
# Please compare all your models using Prettytable library
```

In [1]:
```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names = ["Classifier", "Vectorizer", "n_estimators", "depth", "Auc"]

x.add_row(["RandomForest", "Bow", "300", "9", "0.5"])
x.add_row(["GradientBoost","Bow", "400", "3", "0.504"])
x.add_row(["RandomForest","Tfidf", "400", "9", "0.5"])
x.add_row(["GradientBoost","Tfidf", "400", "2", "0.503"])
x.add_row(["RandomForest","Avgw2v", "400", "5", "0.5"])
x.add_row(["GradientBoost","Avgw2v", "400", "2", "0.505"])
x.add_row(["RandomForest","Tfidfw2v", "400", "5", "0.5"])
x.add_row(["GradientBoost","Tfidfw2v", "400", "2", "0.5046"])
print(x)
```

```
+---------------+------------+--------------+-------+--------+
|   Classifier  | Vectorizer | n_estimators | depth |  Auc   |
+---------------+------------+--------------+-------+--------+
|  RandomForest |    Bow     |     300      |   9   |  0.5   |
| GradientBoost |    Bow     |     400      |   3   | 0.504  |
|  RandomForest |   Tfidf    |     400      |   9   |  0.5   |
| GradientBoost |   Tfidf    |     400      |   2   | 0.503  |
|  RandomForest |   Avgw2v   |     400      |   5   |  0.5   |
| GradientBoost |   Avgw2v   |     400      |   2   | 0.505  |
|  RandomForest |  Tfidfw2v  |     400      |   5   |  0.5   |
| GradientBoost |  Tfidfw2v  |     400      |   2   | 0.5046 |
+---------------+------------+--------------+-------+--------+
```

Steps Taken: (1)Importing the important library (2)Merging project_data and resources into project_data (3)Preprocessing all text and categorical value (4)taking 50000 dataset due to computation problem (5)Splitting into train_test_split (6)Vectorizing text data (7)Standardizing Numerical data (8)Applying RandomForest and GradientBoost