Assignment 9: GBDT

Response Coding: Example

Train Data

State	class
A	0
B	1
C	1
A	0
A	1
B	1
A	0
A	1
C	1
C	0

Resonse table(only from train)

State	Class=0	Class=1
A	3	2
B	0	2
C	1	2

Encoded Train Data

State_0	State_1	class
3/5	2/5	0
0/2	2/2	1
1/3	2/3	1
3/5	2/5	0
3/5	2/5	1
0/2	2/2	1
3/5	2/5	0
3/5	2/5	1
1/3	2/3	1
1/3	2/3	0

Test Data

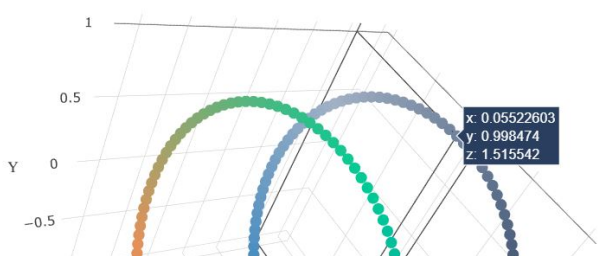
State
A
C
D
C
B
E

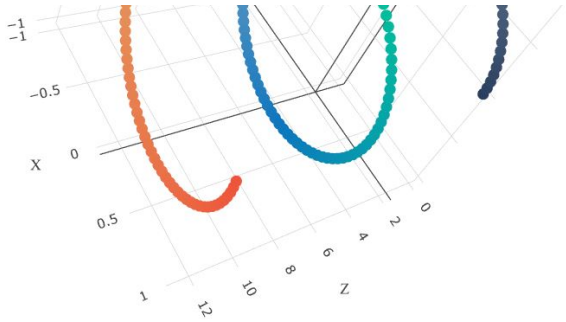
Encoded Test Data

State_0	State_1
3/5	2/5
1/3	2/3
1/2	1/2
1/3	2/3
0/2	2/2
1/2	1/2

> 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 GBDT on these feature sets**
   - Set 1: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)+sentiment Score of eassay(check the bellow example, include all 4 values as 4 features)
   - Set 2: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)
2. **The hyper paramter tuning (Consider any two hyper parameters)**
   - Find the best hyper parameter which will give the maximum AUC 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

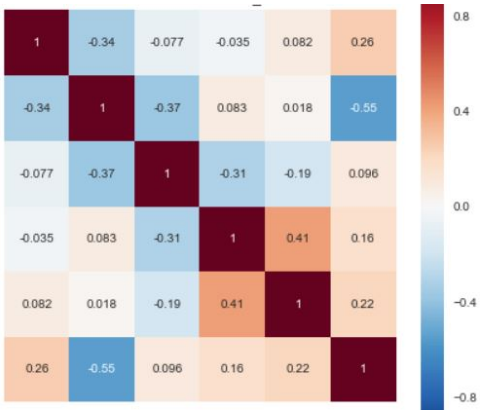




with X-axis as **n_estimators**, Y-axis as **max_depth**, and Z-axis as **AUC Score** , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

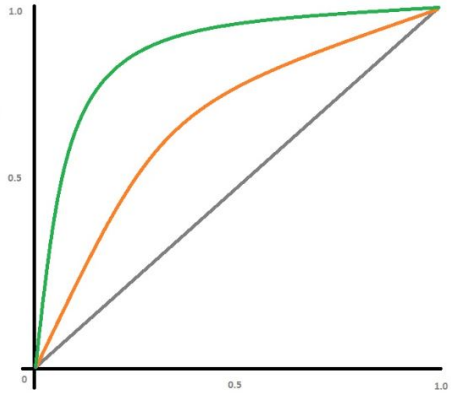
or

- 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 with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**

- You choose either of the plotting techniques out of 3d plot or heat map
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.



- Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points

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

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

Vectorizer	Model	Hyper parameter	AUC
BOW	Brute	7	0.78
TFIDF	Brute	13	0.79

TFIDF	Brute	12	0.79
W2V	Brute	10	0.78
TFIDFW2V	Brute	6	0.78

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 t
he biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelligence
s i use a wide range\
of techniques to help all my students succeed students in my class come from a variety of different bac
kgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a caring com
munity of successful \
learners which can be seen through collaborative student project based learning in and out of the class
room kindergarteners \
in my class love to work with hands on materials and have many different opportunities to practice a sk
ill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the kinderg
arten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role play in
our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food i wil
l take their idea \
and create common core cooking lessons where we learn important math and writing concepts while cooking
delicious healthy \
food for snack time my students will have a grounded appreciation for the work that went into making th
e food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project would exp
and our 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
healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

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

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

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

```
C:\Users\meera\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:20: UserWarning:

The twython library has not been installed. Some functionality from the twitter package will not be ava
ilable.
```

1. GBDT (xgboost/lightgbm)

1.1 Loading Data

In [1]:

```
import pandas
```

```
import pandas
data = pandas.read_csv('preprocessed_data.csv')
```

In [2]:

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

import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
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, roc_auc_score
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
from tqdm import tqdm
from scipy.sparse import csr_matrix
from scipy import sparse
import math
import operator
from sklearn.preprocessing import normalize
from gensim.models import Word2Vec
import pickle
import os

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

In [3]:

```
y = data['project_is_approved']
X = data.drop('project_is_approved',axis=1)
X.head(1)
```

Out[3]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	clean_cate
0	ca	mrs	grades_prek_2	53	math_scien

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

In [4]:

```
# 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
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

1.3 Make Data Model Ready: encoding eassay, and project_title

In [0]:

```
# 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 [62]:

```
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'])
lst_w2v = list(vectorizer.get_feature_names())
```

In [63]:

```
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].values)
X_cv_essay_tfidf = vectorizer.transform(X_cv['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================
```

In [60]:

```
def tfidf_w2v(w2v_corpus):
    '''This function computes tfidf_w2v and returns vector'''

    train_index = list(w2v_corpus.index)
    final_w2v_corpus = [x.split(' ') for x in w2v_corpus[train_index[:]]]
    final_tfidf_corpus = [x for x in w2v_corpus[train_index[:]]]

    Idf_values = open("data.pkl","rb")
    Idf_dict = pickle.load(Idf_values)

    model = Word2Vec(final_w2v_corpus,min_count = 1)

    if isinstance(final_tfidf_corpus,(list,)):
        tfidf_w2v = list()
        for idx,row in enumerate(tqdm(final_tfidf_corpus)):
            sum1 = model['this']
            sum2 = sum1*0
            word_freq = dict(Counter(row.split(' ')))
            for word,freq in word_freq.items():
                try:
                    sum2 += freq*Idf_dict[word]*model[word]
                except:
```

```
                continue

            tfidf_w2v.append(sum2)
        tfidf_w2v_array = np.array(tfidf_w2v)
        tfidf_w2v_sparse = sparse.csr_matrix(tfidf_w2v_array)
        output = normalize(tfidf_w2v_sparse, norm='l2', axis=1, copy=False, return_norm=False)

    Idf_values.close()

    return output
```

In [61]:

```
X_train_essay_tfidfw2v = tfidf_w2v(X_train['essay'])
X_cv_essay_tfidfw2v = tfidf_w2v(X_cv['essay'])
X_test_essay_tfidfw2v = tfidf_w2v(X_test['essay'])
```

```
100%|██████████| 49041/49041 [02:06<00
:00, 386.23it/s]
100%|██████████| 24155/24155 [01:01<00
:00, 395.27it/s]
100%|██████████| 36052/36052 [01:33<00
:00, 383.78it/s]
```

In [64]:

```
def senti(corpus):
    '''Returns the vector of sentiment analysis'''
    sid = SentimentIntensityAnalyzer()
    senti_list = list()

    indexes = list(corpus.index)
    senti_corpus = [x for x in corpus[indexes[:]]]

    for i in tqdm(range(len(senti_corpus))):
        senti_list.append(list(sid.polarity_scores(senti_corpus[i]).values()))

    senti_array = np.array(senti_list)
    senti_csr = sparse.csr_matrix(senti_array)
    output = normalize(senti_csr, norm='l2', axis=1, copy=False, return_norm=False)

    return output
```

In [65]:

```
X_train_essay_senti = senti(X_train['essay'])
X_cv_essay_senti = senti(X_cv['essay'])
X_test_essay_senti = senti(X_test['essay'])
```

```
100%|██████████| 49041/49041 [01:40<00
:00, 485.99it/s]
100%|██████████| 24155/24155 [00:39<00
:00, 605.61it/s]
100%|██████████| 36052/36052 [01:01<00
:00, 587.96it/s]
```

1.4 Make Data Model Ready: encoding numerical, categorical features

In [2]:

```
# 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


#applying response coding to categorical features
```

In [66]:

```
def response_code_train(corpus,y_values):
    '''this functions gives the response coded vector along with prob_values'''
    corpus_set = set()
    corpus_values = dict()
    to_arr_list = list()

    for i in tqdm(corpus.values):
        num = 0
        den = 0

        if(i in corpus_set):
            to_arr_list.append([corpus_values[i],1-corpus_values[i]])
        else:
            for k,j in enumerate(corpus.values):
                if(i == j):
                    den+=1
                    if(y_values.values[k]==1):
                        num+=1
                else:
                    continue

            corpus_set.add(i)

            try:
                temp = [(1-(num/den)),(num/den)]
            except:
                print('there is exception in the data')

            to_arr_list.append(temp)
            corpus_values[i] = temp[0]

    print(to_arr_list[:10],"\n\n this is how data is.. ")
    output = sparse.csr_matrix(np.array(to_arr_list))

    return output,corpus_values
```

In [67]:

```
def response_code_cv_or_test(test_corpus,test_dict):

    output = list()
    dict_keys = test_dict.keys()

    for i in test_corpus.values:
        if(i in dict_keys):
            temp = [test_dict[i],1-test_dict[i]]
        else:
            temp = [0.5,0.5]

        output.append(temp)

    return sparse.csr_matrix(np.array(output))
```

school_state teacher_prefix project_grade_category teacher_number_of_previously_posted_projects clean_categories clean_subcategories

In [68]:

```
X_train_state_resp,state_dict = response_code_train(X_train['school_state'],y_train)
X_cv_state_resp = response_code_cv_or_test(X_cv['school_state'],state_dict)
X_test_state_resp = response_code_cv_or_test(X_test['school_state'],state_dict)
```

```
100%|██████████| 49041/49041 [00:00<00:00, 222459.65it/s]
```

```
[[0.14403417760146475, 0.8559658223985352], [0.16573556797020483, 0.8342644320297952], [0.1450577663671374, 0.8549422336328626], [0.1651634723788049, 0.8348365276211951], [0.14383757506434092, 0.8561624249356591], [0.1475177304964539, 0.8524822695035461], [0.14383757506434092, 0.8561624249356591], [0.1869499241274658, 0.8130500758725342], [0.1450577663671374, 0.8549422336328626], [0.1362397820163488, 0.8637602179836512]]

 this is how data is..
```

In [69]:

```
X_train_teacher_resp,teacher_dict = response_code_train(X_train['teacher_prefix'],y_train)
X_cv_teacher_resp = response_code_cv_or_test(X_cv['teacher_prefix'],teacher_dict)
X_test_teacher_resp = response_code_cv_or_test(X_test['teacher_prefix'],teacher_dict)
```

```
100%|██████████| 49041/49041 [00:00<00:00, 67776.95it/s]
```

```
[[0.1449230290779039, 0.8550769709220961], [0.15657751560620814, 0.8434224843937919], [0.1449230290779039, 0.8550769709220961], [0.1449230290779039, 0.8550769709220961], [0.15657751560620814, 0.8434224843937919], [0.15657751560620814, 0.8434224843937919], [0.1449230290779039, 0.8550769709220961], [0.15657751560620814, 0.8434224843937919], [0.1449230290779039, 0.8550769709220961], [0.15657751560620814, 0.8434224843937919]]

 this is how data is..
```

In [70]:

```
X_train_grade_resp,grade_dict = response_code_train(X_train['project_grade_category'],y_train)
X_cv_grade_resp = response_code_cv_or_test(X_cv['project_grade_category'],grade_dict)
X_test_grade_resp = response_code_cv_or_test(X_test['project_grade_category'],grade_dict)
```

```
100%|██████████| 49041/49041 [00:00<00:00, 54266.72it/s]
```

```
[[0.15241018416020935, 0.8475898158397907], [0.15241018416020935, 0.8475898158397907], [0.15241018416020935, 0.8475898158397907], [0.14533228327376269, 0.8546677167262373], [0.15584247901872172, 0.8441575209812783], [0.15584247901872172, 0.8441575209812783], [0.14533228327376269, 0.8546677167262373], [0.15241018416020935, 0.8475898158397907], [0.15241018416020935, 0.8475898158397907], [0.14533228327376269, 0.8546677167262373]]

 this is how data is..
```

In [71]:

```
X_train_clean_cat_resp,clean_cat_dict = response_code_train(X_train['clean_categories'],y_train)
X_cv_clean_cat_resp = response_code_cv_or_test(X_cv['clean_categories'],clean_cat_dict)
X_test_clean_cat_resp = response_code_cv_or_test(X_test['clean_categories'],clean_cat_dict)
```

```
100%|██████████| 49041/49041 [00:00<00:00, 52089.29it/s]
```

```
[[0.13094521372667067, 0.8690547862733293], [0.16725978647686834, 0.8327402135231317], [0.13094521372667067, 0.8690547862733293], [0.08085808580858089, 0.9191419141914191], [0.17610062893081757, 0.8238993710691824], [0.16155686018699722, 0.8384431398130028], [0.13882618510158018, 0.8611738148984198], [0.13355450236966826, 0.8664454976303317], [0.16141001855287573, 0.8385899814471243], [0.13355450236966826, 0.8664454976303317]]

 this is how data is..
```

In [72]:

```
X_train_clean_sub_cat_resp,clean_sub_cat_dict = response_code_train(X_train['clean_subcategories'],y_train)
X_cv_clean_sub_cat_resp = response_code_cv_or_test(X_cv['clean_subcategories'],clean_sub_cat_dict)
X_test_clean_sub_cat_resp = response_code_cv_or_test(X_test['clean_subcategories'],clean_sub_cat_dict)
```

```
100%|██████████| 49041/49041 [00:01<00:00, 28700.46it/s]
```

```
[[0.09523809523809523, 0.9047619047619048], [0.1384615384615384, 0.8615384615384616], [0.12480127186009538, 0.8751987281399046], [0.08085808580858089, 0.9191419141914191], [0.18742085268045594, 0.8125791473195441], [0.19027484143763218, 0.8097251585623678], [0.14922813036020588, 0.8507718696397941], [0.11572356215213353, 0.8842764378478665], [0.125, 0.875], [0.14842578710644683, 0.8515742128935532]]
```

this is how data is..

In [73]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(X_train['price'].values.reshape(1,-1))

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

In [74]:

```
normalizer = Normalizer()
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

X_train_ppp_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_cv_ppp_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_ppp_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
```

1.5 Appling Models on different kind of featurization as mentioned in the instructions

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

In [0]:

```
# 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 [75]:

```
#TFIDF featurization
from scipy.sparse import hstack
X_tr1 = hstack((X_train_essay_tfidf,X_train_essay_senti, X_train_state_resp, X_train_teacher_resp, X_train_grade_resp,X_train_clean_cat_resp,X_train_clean_sub_cat_resp,X_train_ppp_norm,X_train_price_norm)).tocsr()
X_cr1 = hstack((X_cv_essay_tfidf,X_cv_essay_senti, X_cv_state_resp, X_cv_teacher_resp, X_cv_grade_resp, X_cv_clean_cat_resp, X_cv_clean_sub_cat_resp, X_cv_ppp_norm, X_cv_price_norm)).tocsr()
X_te1 = hstack((X_test_essay_tfidf,X_test_essay_senti, X_test_state_resp, X_test_teacher_resp, X_test_grade_resp,X_test_clean_cat_resp,X_test_clean_sub_cat_resp,X_test_ppp_norm,X_test_price_norm)).tocsr()
```

In [76]:

```
X_tr2 = hstack((X_train_essay_tfidfw2v,X_train_essay_senti, X_train_state_resp, X_train_teacher_resp, X
_train_grade_resp,X_train_clean_cat_resp,X_train_clean_sub_cat_resp,X_train_ppp_norm,X_train_price_norm
)).tocsr()
X_cr2 = hstack((X_cv_essay_tfidfw2v,X_cv_essay_senti, X_cv_state_resp, X_cv_teacher_resp, X_cv_grade_re
sp, X_cv_clean_cat_resp, X_cv_clean_sub_cat_resp, X_cv_ppp_norm, X_cv_price_norm)).tocsr()
X_te2 = hstack((X_test_essay_tfidfw2v,X_test_essay_senti, X_test_state_resp, X_test_teacher_resp, X_tes
t_grade_resp,X_test_clean_cat_resp,X_test_clean_sub_cat_resp,X_test_ppp_norm,X_test_price_norm)).tocsr(
)
```

In [77]:

```
import dill as pickle

f1_x_train = open("X_tr1.pkl","wb")
f1_x_cv = open("X_cr1.pkl","wb")
f1_x_test = open("X_te1","wb")

f2_x_train = open("X_tr2.pkl","wb")
f2_x_cv = open("X_cr2.pkl","wb")
f2_x_test = open("X_te2.pkl","wb")

pickle.dump(X_tr1,f1_x_train)
pickle.dump(X_cr1,f1_x_cv)
pickle.dump(X_te1,f1_x_test)

pickle.dump(X_tr2,f2_x_train)
pickle.dump(X_cr2,f2_x_cv)
pickle.dump(X_te2,f2_x_test)

f1_x_train.close()
f1_x_cv.close()
f1_x_test.close()
f2_x_train.close()
f2_x_cv.close()
f2_x_test.close()
```

In [5]:

```
import dill as pickle

X_tr1_value = open("X_tr1.pkl","rb")
X_cr1_value = open("X_cr1.pkl","rb")
X_te1_value = open("X_te1","rb")

X_tr2_value = open("X_tr2.pkl","rb")
X_cr2_value = open("X_cr2.pkl","rb")
X_te2_value = open("X_te2.pkl","rb")

X_tr1 = pickle.load(X_tr1_value)
X_cr1 = pickle.load(X_cr1_value)
X_te1 = pickle.load(X_te1_value)

X_tr2 = pickle.load(X_tr2_value)
X_cr2 = pickle.load(X_cr2_value)
X_te2 = pickle.load(X_te2_value)

X_tr1_value.close()
X_cr1_value.close()
X_te1_value.close()
X_tr2_value.close()
X_cr2_value.close()
X_te2_value.close()
```

In [11]:

```
print(y_train.shape,X_tr1.shape,X_tr2.shape)
print(y_cv.shape,X_cr1.shape,X_cr2.shape)
print(y_test.shape,X_te1.shape,X_te2.shape)
```

```
print("This output clearly shows there is no upsampling of the data!!!")
```

```
(49041,) (49041, 5016) (49041, 116)
(24155,) (24155, 5016) (24155, 116)
(36052,) (36052, 5016) (36052, 116)
This output clearly shows there is no upsampling of the data!!!
```

In [6]:

```
from xgboost import XGBClassifier
from tqdm import tqdm

estimators = [5,10,50, 75, 100]
l_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]

train_auc = []
cv_auc = []
for i in tqdm(estimators):
    for j in tqdm(l_rate):
        pred_list = []
        pred_list_cv = []

        clf = XGBClassifier(n_estimators = i,learning_rate = j,n_jobs = -1)
        clf.fit(X_tr1,y_train)

        train_pred = clf.predict_proba(X_tr1)
        cv_pred = clf.predict_proba(X_cr1)


        train_auc.append(roc_auc_score(y_train,np.transpose(train_pred)[1]))
        cv_auc.append(roc_auc_score(y_cv,np.transpose(cv_pred)[1]))

        print("train_auc = ",train_auc,"\ncv_auc = ",cv_auc)
        print("roc_auc_score train: ",roc_auc_score(y_train,np.transpose(train_pred)[1]))
        print("roc_auc_score cv: ",roc_auc_score(y_cv,np.transpose(cv_pred)[1]))
```

```
  0%|                                                                  | 0/5
[00:00<?, ?it/s]
  0%|                                                                  | 0/6
[00:00<?, ?it/s]
 17%|█████                                                  | 1/6 [00:08<0
0:41,  8.24s/it]
```

```
train_auc =  [0.5044410371850591]
cv_auc =  [0.5002742686061558]
roc_auc_score train:  0.5044410371850591
roc_auc_score cv:  0.5002742686061558
```

```
 33%|██████████                                             | 2/6 [00:15<0
0:31,  7.96s/it]
```

```
train_auc =  [0.5044410371850591, 0.5042744950951678]
cv_auc =  [0.5002742686061558, 0.49988552742481956]
roc_auc_score train:  0.5042744950951678
roc_auc_score cv:  0.49988552742481956
```

```
 50%|███████████████                                        | 3/6 [00:22<0
0:23,  7.71s/it]
```

```
train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047]
roc_auc_score train:  0.5048169517435664
roc_auc_score cv:  0.49964152114669047
```

```
 67%|████████████████████                                   | 4/6 [00:29<0
0:15,  7.51s/it]
```

```
train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451]
```

cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582]
roc_auc_score train: 0.5160565559683451
roc_auc_score cv: 0.5000544615182582

 83%|[redacted]| 5/6 [00:36<0
0:07, 7.38s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263]
roc_auc_score train: 0.542138379789161
roc_auc_score cv: 0.4975674077515263

100%|[redacted]| 6/6 [00:44<0
0:00, 7.36s/it]
 20%|[redacted]| 1/5 [00:44<0
2:56, 44.15s/it]
  0%| | 0/6
[00:00<?, ?it/s]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692]
roc_auc_score train: 0.5461437693108429
roc_auc_score cv: 0.4996518131396692

 17%|[redacted]| 1/6 [00:09<0
0:48, 9.68s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408]
roc_auc_score train: 0.50453571963304
roc_auc_score cv: 0.5003717323789408

 33%|[redacted]| 2/6 [00:19<0
0:38, 9.68s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103]
roc_auc_score train: 0.5043220919552958
roc_auc_score cv: 0.49988556744553103

 50%|[redacted]| 3/6 [00:28<0
0:28, 9.64s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087]
roc_auc_score train: 0.5072601310300237
roc_auc_score cv: 0.49904493907020087

 67%|[redacted]| 4/6 [00:39<0
0:19, 9.78s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018]

```
roc_auc_score train:  0.5675598356020177
roc_auc_score cv:  0.5007436515212018

 83%|                                                                                | 5/6 [00:48<0
0:09,  9.79s/it]

train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265]
roc_auc_score train:  0.6015886248908248
roc_auc_score cv:  0.4998149042092265

100%|                                                                               | 6/6 [00:59<0
0:00,  9.85s/it]
 40%|                                                                               | 2/5 [01:43<0
2:25, 48.63s/it]
  0%|                                                                                | 0/6
[00:00<?, ?it/s]

train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591]
roc_auc_score train:  0.6309855041689887
roc_auc_score cv:  0.4975280874024591

 17%|                                                                               | 1/6 [00:31<0
2:36, 31.39s/it]

train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527]
roc_auc_score train:  0.5043220725398929
roc_auc_score cv:  0.49988554743517527

 33%|                                                                               | 2/6 [01:03<0
2:06, 31.66s/it]

train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402]
roc_auc_score train:  0.5048643188547606
roc_auc_score cv:  0.499641561167402

 50%|                                                                               | 3/6 [01:35<0
1:35, 31.82s/it]

train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386]
roc_auc_score train:  0.5307907773859355
roc_auc_score cv:  0.49849005857511386
```

67%|[redacted]| 4/6 [02:07<01:03, 31.62s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.542138379789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355, 0.7693662592463024]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.4975674077515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.5007436515212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.49849005857511386, 0.49575773787783983]
roc_auc_score train: 0.7693662592463024
roc_auc_score cv: 0.49575773787783983

83%|[redacted]| 5/6 [02:38<00:31, 31.60s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.542138379789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355, 0.7693662592463024, 0.8161894851420233]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.4975674077515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.5007436515212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.49849005857511386, 0.49575773787783983, 0.4979944154031928]
roc_auc_score train: 0.8161894851420233
roc_auc_score cv: 0.4979944154031928

100%|[redacted]| 6/6 [03:10<00:00, 31.67s/it]
60%|[redacted]| 3/5 [04:53<03:02, 91.06s/it]
0%| | 0/6 [00:00<?, ?it/s]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.542138379789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.4975674077515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.5007436515212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.49849005857511386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098]
roc_auc_score train: 0.8332259073699543
roc_auc_score cv: 0.4892382038886098

17%|[redacted]| 1/6 [00:45<03:45, 45.06s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.542138379789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.4975674077515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.5007436515212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.49849005857511386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939]
roc_auc_score train: 0.5043220612142413
roc_auc_score cv: 0.4998855541052939

33%|[redacted]| 2/6 [01:30<03:00, 45.13s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.542138379789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740

77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229]
roc_auc_score train: 0.5072134402220294
roc_auc_score cv: 0.4990420042180229

50%|[redacted]| 3/6 [02:16<0
2:16, 45.41s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891]
roc_auc_score train: 0.5719327942948744
roc_auc_score cv: 0.5008471450811891

67%|[redacted]| 4/6 [03:01<0
1:30, 45.46s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891, 0.4958661006243951]
roc_auc_score train: 0.8360597601569981
roc_auc_score cv: 0.4958661006243951

83%|[redacted]| 5/6 [03:47<0
0:45, 45.35s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309]
roc_auc_score train: 0.8744845720193174
roc_auc_score cv: 0.4930016115273309

100%|[redacted]| 6/6 [04:32<0
0:00, 45.45s/it]
80%|[redacted]| 4/5 [09:26<02
:25, 145.56s/it]
  0%|| 0/6
[00:00<?, ?it/s]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174, 0.8893444143293568]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180

229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309, 0.48779873893670783]
roc_auc_score train: 0.8893444143293568
roc_auc_score cv: 0.48779873893670783

 17%|[redacted]| 1/6 [00:58<0
4:52, 58.46s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174, 0.8893444143293568, 0.504839887806153]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309, 0.48779873893670783, 0.499642268199972
2]
roc_auc_score train: 0.504839887806153
roc_auc_score cv: 0.4996422681999722

 33%|[redacted]| 2/6 [01:57<0
3:54, 58.64s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174, 0.8893444143293568, 0.504839887806153, 0.508
5408163057283]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309, 0.48779873893670783, 0.499642268199972
2, 0.49914721866860534]
roc_auc_score train: 0.5085408163057283
roc_auc_score cv: 0.49914721866860534

 50%|[redacted]| 3/6 [02:56<0
2:56, 58.73s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174, 0.8893444143293568, 0.504839887806153, 0.508
5408163057283, 0.5895460724112336]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309, 0.48779873893670783, 0.499642268199972
2, 0.49914721866860534, 0.5013179087107826]
roc_auc_score train: 0.5895460724112336
roc_auc_score cv: 0.5013179087107826

 67%|[redacted]| 4/6 [03:56<0
1:58, 59.14s/it]

train_auc = [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174, 0.8893444143293568, 0.504839887806153, 0.508
5408163057283, 0.5895460724112336, 0.8732006702585378]
cv_auc = [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180

```
229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309, 0.48779873893670783, 0.499642268199972
2, 0.49914721866860534, 0.5013179087107826, 0.4961243809562992]
roc_auc_score train:  0.8732006702585378
roc_auc_score cv:  0.4961243809562992
```

```
 83%|                                                                                | 5/6 [04:55<0
0:59, 59.14s/it]
```

```
train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174, 0.8893444143293568, 0.504839887806153, 0.508
5408163057283, 0.5895460724112336, 0.8732006702585378, 0.9087881005843421]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309, 0.48779873893670783, 0.499642268199972
2, 0.49914721866860534, 0.5013179087107826, 0.4961243809562992, 0.49186532347553835]
roc_auc_score train:  0.9087881005843421
roc_auc_score cv:  0.49186532347553835
```

```
100%|                                                                               | 6/6 [05:54<0
0:00, 59.06s/it]
100%|                                                                               | 5/5 [15:20<00
:00, 184.08s/it]
```

```
train_auc =  [0.5044410371850591, 0.5042744950951678, 0.5048169517435664, 0.5160565559683451, 0.5421383
79789161, 0.5461437693108429, 0.50453571963304, 0.5043220919552958, 0.5072601310300237, 0.5675598356020
177, 0.6015886248908248, 0.6309855041689887, 0.5043220725398929, 0.5048643188547606, 0.5307907773859355
, 0.7693662592463024, 0.8161894851420233, 0.8332259073699543, 0.5043220612142413, 0.5072134402220294, 0
.5719327942948744, 0.8360597601569981, 0.8744845720193174, 0.8893444143293568, 0.504839887806153, 0.508
5408163057283, 0.5895460724112336, 0.8732006702585378, 0.9087881005843421, 0.9251333102009595]
cv_auc =  [0.5002742686061558, 0.49988552742481956, 0.49964152114669047, 0.5000544615182582, 0.49756740
77515263, 0.4996518131396692, 0.5003717323789408, 0.49988556744553103, 0.49904493907020087, 0.500743651
5212018, 0.4998149042092265, 0.4975280874024591, 0.49988554743517527, 0.499641561167402, 0.498490058575
11386, 0.49575773787783983, 0.4979944154031928, 0.4892382038886098, 0.4998855541052939, 0.4990420042180
229, 0.5008471450811891, 0.4958661006243951, 0.4930016115273309, 0.48779873893670783, 0.499642268199972
2, 0.49914721866860534, 0.5013179087107826, 0.4961243809562992, 0.49186532347553835, 0.4911904208666069
]
roc_auc_score train:  0.9251333102009595
roc_auc_score cv:  0.4911904208666069
```

In [8]:

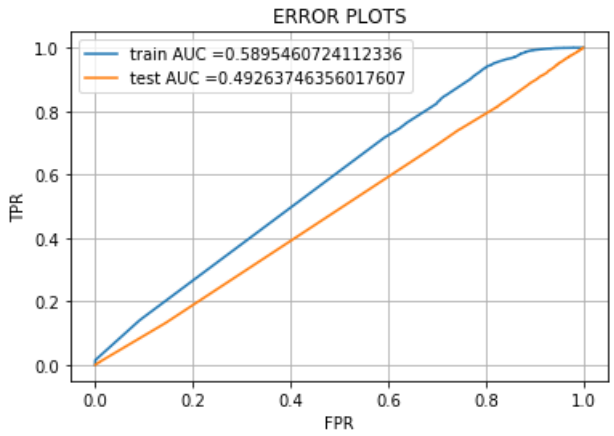
```
x1 = estimators*4
y1 = [0.0001,0.0001,0.0001,0.0001,0.001,0.001,0.001,0.001,0.01,0.01,0.01,0.01,0.1,0.1,0.1,0.1,0.2,0.2,0
.2,0.2,0.3,0.3,0.3,0.3]
z1 = train_auc
z2 = cv_auc
```

In [9]:

```
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x1,y=y1,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

In [11]:

```
from sklearn.metrics import roc_curve, auc
from xgboost import XGBClassifier

estimators = 100
l_rate = 0.01

clf = XGBClassifier(n_estimators = estimators,learning_rate = l_rate)
clf.fit(X_tr1,y_train)

train_pred = clf.predict_proba(X_tr1)
test_pred = clf.predict_proba(X_te1)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,np.transpose(train_pred)[1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,np.transpose(test_pred)[1])

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("FPR")
plt.ylabel("TPR")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



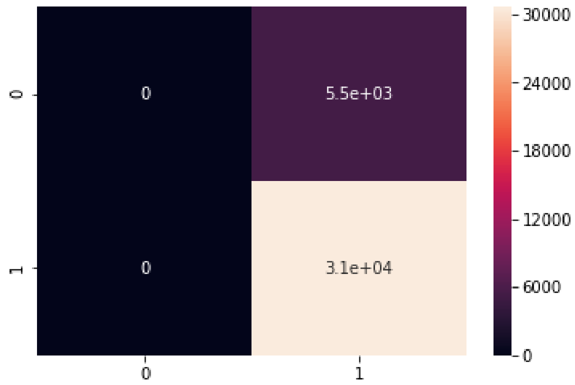
In [15]:

```
y_test_pred = clf.predict(X_te1)
cf_matrix = confusion_matrix(y_test,y_test_pred)
print(cf_matrix)
sns.heatmap(cf_matrix,annot = True)
```

```
[[    0  5459]
 [    0 30593]]
```

Out[15]:

<matplotlib.axes._subplots.AxesSubplot at 0x24005dda710>



In [16]:

```
from xgboost import XGBClassifier
import numpy as np
from sklearn.metrics import roc_curve, auc, roc_auc_score

estimators = [5,10,50, 75, 100]
l_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]

train_auc_2 = []
cv_auc_2 = []
for i in tqdm(estimators):
    for j in tqdm(l_rate):
        pred_list = []
        pred_list_cv = []
        clf = XGBClassifier(max_depth = i,learning_rate = j,n_jobs = -1)
        clf.fit(X_tr2,y_train)

        train_pred = clf.predict_proba(X_tr2)
        cv_pred = clf.predict_proba(X_cr2)

        train_auc_2.append(roc_auc_score(y_train,np.transpose(train_pred)[1]))
        cv_auc_2.append(roc_auc_score(y_cv,np.transpose(cv_pred)[1]))

        print("train_auc = ",train_auc_2,"\ncv_auc = ",cv_auc_2)
        print("roc_auc_score train: ",roc_auc_score(y_train,np.transpose(train_pred)[1]))
        print("roc_auc_score cv: ",roc_auc_score(y_cv,np.transpose(cv_pred)[1]))
```

```
  0%|                                                                | 0/5
[00:00<?, ?it/s]
  0%|                                                                | 0/6
[00:00<?, ?it/s]
 17%|                                                    | 1/6 [00:19<0
1:38, 19.76s/it]
```

```
train_auc =  [0.5169125940243467]
cv_auc =  [0.5021658475360836]
roc_auc_score train:  0.5169125940243467
roc_auc_score cv:  0.5021658475360836
```

```
 33%|                                                    | 2/6 [00:40<0
1:20, 20.10s/it]
```

```
train_auc =  [0.5169125940243467, 0.5312180052362694]
cv_auc =  [0.5021658475360836, 0.5041999335496107]
roc_auc_score train:  0.5312180052362694
roc_auc_score cv:  0.5041999335496107
```

```
 50%|[redacted]| 3/6 [01:02<0
1:01, 20.48s/it]
```

```
train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723]
roc_auc_score train:  0.6042677983991289
roc_auc_score cv:  0.49591908137632723
```

```
 67%|[redacted]| 4/6 [01:23<0
0:41, 20.78s/it]
```

```
train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565]
roc_auc_score train:  0.8325639246476565
roc_auc_score cv:  0.4886215380850565
```

```
 83%|[redacted]| 5/6 [01:45<0
0:21, 21.19s/it]
```

```
train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748]
roc_auc_score train:  0.8800175783821657
roc_auc_score cv:  0.5043833751509748
```

```
100%|[redacted]| 6/6 [02:08<0
0:00, 21.34s/it]
 20%|[redacted]| 1/5 [02:08<08
:32, 128.07s/it]
  0%|| 0/6
[00:00<?, ?it/s]
```

```
train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494]
roc_auc_score train:  0.9012534406116317
roc_auc_score cv:  0.49374426253074494
```

```
 17%|[redacted]| 1/6 [00:38<0
3:14, 38.99s/it]
```

```
train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763]
roc_auc_score train:  0.535015494624053
roc_auc_score cv:  0.5022775253315763
```

```
 33%|[redacted]| 2/6 [01:18<0
2:37, 39.29s/it]
```

```
train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109]
roc_auc_score train:  0.5565113485133092
roc_auc_score cv:  0.502935812675109
```

```
 50%|[redacted]| 3/6 [01:50<0
```

50%|[redacted]| 3/6 [01:59<01:58, 39.64s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414]
roc_auc_score train: 0.8171584512708499
roc_auc_score cv: 0.5024163238292414

67%|[redacted]| 4/6 [02:40<01:20, 40.08s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657]
roc_auc_score train: 0.9978898612086692
roc_auc_score cv: 0.49108521975626157

83%|[redacted]| 5/6 [03:22<00:40, 40.71s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603]
roc_auc_score train: 0.9998806794057812
roc_auc_score cv: 0.49851642555388603

100%|[redacted]| 6/6 [04:05<00:00, 40.90s/it]
40%|[redacted]| 2/5 [06:13<08:09, 163.27s/it]
0%| | 0/6 [00:00<?, ?it/s]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603, 0.4919220928548276]
roc_auc_score train: 0.9999893490335772
roc_auc_score cv: 0.4919220928548276

17%|[redacted]| 1/6 [02:55<14:39, 175.98s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076]
roc_auc_score train: 0.8955228534015091
roc_auc_score cv: 0.4980886241581076

33%|[redacted]| 2/6 [05:54<11:46, 176.72s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658]

```
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307]
roc_auc_score train:  0.976444429120658
roc_auc_score cv:  0.5002161585330307

 50%|                                                                                | 3/6 [08:55<08
:54, 178.07s/it]

train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815]
roc_auc_score train:  0.999973561075146
roc_auc_score cv:  0.5041356135960815

 67%|                                                                                | 4/6 [10:53<05
:19, 159.92s/it]

train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475]
roc_auc_score train:  0.9999983480728061
roc_auc_score cv:  0.5041966451811475

 83%|                                                                                | 5/6 [12:36<02
:22, 142.79s/it]

train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517]
roc_auc_score train:  0.9999984224985171
roc_auc_score cv:  0.5006070474925517

100%|                                                                               | 6/6 [14:06<00
:00, 141.14s/it]
 60%|                                                                               | 3/5 [20:20<12
:16, 368.34s/it]
  0%|                                                                                          | 0/6
[00:00<?, ?it/s]

train_auc =  [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119]
cv_auc =  [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814]
roc_auc_score train:  0.999998435442119
roc_auc_score cv:  0.502114647705814

 17%|                                                                                | 1/6 [03:36<18
:03, 216.75s/it]
```

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623]
roc_auc_score train: 0.9585061598116111
roc_auc_score cv: 0.4967911126699623

33%|[redacted]| 2/6 [07:05<14:17, 214.30s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804]
roc_auc_score train: 0.9967908523293904
roc_auc_score cv: 0.49913191741656804

50%|[redacted]| 3/6 [10:25<10:30, 210.09s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.9999905835296096]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804, 0.5004937822088353]
roc_auc_score train: 0.9999905835296096
roc_auc_score cv: 0.5004937822088353

67%|[redacted]| 4/6 [12:24<06:05, 182.75s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.9999905835296096, 0.9999983513087065]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804, 0.5004937822088353, 0.4985067939026472]
roc_auc_score train: 0.9999983513087065
roc_auc_score cv: 0.4985067939026472

83%|[redacted]| 5/6 [14:05<02:38, 158.22s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.9978898612086692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.9999905835296096, 0.9999983513087065, 0.9999984257344176]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.5043833751509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.4910852197562657, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804

, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521]
roc_auc_score train: 0.9999984257344176
roc_auc_score cv: 0.4924700964576521

100%|[redacted]| 6/6 [15:25<00
:00, 154.18s/it]
 80%|[redacted]| 4/5 [35:45<08
:55, 535.36s/it]
  0%| | 0/6
[00:00<?, ?it/s]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.
9999905835296096, 0.9999983513087065, 0.9999984257344176, 0.999998435442119]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804
, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521, 0.5079937369187302]
roc_auc_score train: 0.999998435442119
roc_auc_score cv: 0.5079937369187302

 17%|[redacted]| 1/6 [03:42<18
:34, 222.95s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.
9999905835296096, 0.9999983513087065, 0.9999984257344176, 0.999998435442119, 0.9661592602136101]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804
, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521, 0.5079937369187302, 0.4970688764182573]
roc_auc_score train: 0.9661592602136101
roc_auc_score cv: 0.4970688764182573

 33%|[redacted]| 2/6 [07:42<15
:11, 227.82s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.
9999905835296096, 0.9999983513087065, 0.9999984257344176, 0.999998435442119, 0.9661592602136101, 0.9978
735021138034]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804
, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521, 0.5079937369187302, 0.4970688764182573, 0
.5013429750164305]
roc_auc_score train: 0.9978735021138034
roc_auc_score cv: 0.5013429750164305

 50%|[redacted]| 3/6 [10:52<10
:49, 216.51s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.
9999905835296096, 0.9999983513087065, 0.9999984257344176, 0.999998435442119, 0.9661592602136101, 0.9978
735021138034, 0.9999937094094711]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804

, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521, 0.5079937369187302, 0.4970688764182573, 0
.5013429750164305, 0.5007847127712199]
roc_auc_score train: 0.9999937094094711
roc_auc_score cv: 0.5007847127712199

 67%|▋| 4/6 [12:56<06
:17, 188.94s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.
9999905835296096, 0.9999983513087065, 0.9999984257344176, 0.999998435442119, 0.9661592602136101, 0.9978
735021138034, 0.9999937094094711, 0.9999983577805075]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804
, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521, 0.5079937369187302, 0.4970688764182573, 0
.5013429750164305, 0.5007847127712199, 0.49768013275563705]
roc_auc_score train: 0.9999983577805075
roc_auc_score cv: 0.49768013275563705

 83%|▊| 5/6 [14:44<02
:44, 164.44s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.
9999905835296096, 0.9999983513087065, 0.9999984257344176, 0.999998435442119, 0.9661592602136101, 0.9978
735021138034, 0.9999937094094711, 0.9999983577805075, 0.9999984127908157]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804
, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521, 0.5079937369187302, 0.4970688764182573, 0
.5013429750164305, 0.5007847127712199, 0.49768013275563705, 0.5024558909726962]
roc_auc_score train: 0.9999984127908157
roc_auc_score cv: 0.5024558909726962

100%|▉| 6/6 [16:18<00
:00, 163.01s/it]
100%|▉| 5/5 [52:03<00
:00, 624.70s/it]

train_auc = [0.5169125940243467, 0.5312180052362694, 0.6042677983991289, 0.8325639246476565, 0.8800175
783821657, 0.9012534406116317, 0.535015494624053, 0.5565113485133092, 0.8171584512708499, 0.99788986120
86692, 0.9998806794057812, 0.9999893490335772, 0.8955228534015091, 0.976444429120658, 0.999973561075146
, 0.9999983480728061, 0.9999984224985171, 0.999998435442119, 0.9585061598116111, 0.9967908523293904, 0.
9999905835296096, 0.9999983513087065, 0.9999984257344176, 0.999998435442119, 0.9661592602136101, 0.9978
735021138034, 0.9999937094094711, 0.9999983577805075, 0.9999984127908157, 0.9999984322062184]
cv_auc = [0.5021658475360836, 0.5041999335496107, 0.49591908137632723, 0.4886215380850565, 0.504383375
1509748, 0.49374426253074494, 0.5022775253315763, 0.502935812675109, 0.5024163238292414, 0.491085219756
26157, 0.49851642555388603, 0.4919220928548276, 0.4980886241581076, 0.5002161585330307, 0.5041356135960
815, 0.5041966451811475, 0.5006070474925517, 0.502114647705814, 0.4967911126699623, 0.49913191741656804
, 0.5004937822088353, 0.4985067939026472, 0.4924700964576521, 0.5079937369187302, 0.4970688764182573, 0
.5013429750164305, 0.5007847127712199, 0.49768013275563705, 0.5024558909726962, 0.4958734844456703]
roc_auc_score train: 0.9999984322062184
roc_auc_score cv: 0.4958734844456703

In [17]:

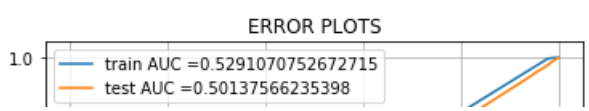
```
x1 = estimators*4
y1 = [0.0001,0.0001,0.0001,0.0001,0.001,0.001,0.001,0.001,0.01,0.01,0.01,0.01,0.1,0.1,0.1,0.1,0.2,0.2,0
.2,0.2,0.3,0.3,0.3,0.3]
z1 = train_auc_2
z2 = cv_auc_2
```

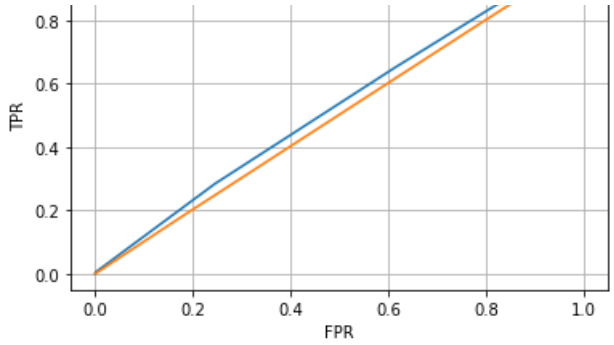
In [18]:

```
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x1,y=y1,z=z2, name = 'Cross validation')
data = [trace1, trace2]

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

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

In [19]:

```
from sklearn.metrics import roc_curve, auc
from xgboost import XGBClassifier
import dill as pickle


clf = XGBClassifier(n_estimators = 50,learning_rate = 0.001,n_jobs = -1)
clf.fit(X_tr2,y_train)

train_pred = clf.predict_proba(X_tr2)
test_pred = clf.predict_proba(X_te2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,np.transpose(train_pred)[1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,np.transpose(test_pred)[1])

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("FPR")
plt.ylabel("TPR")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```





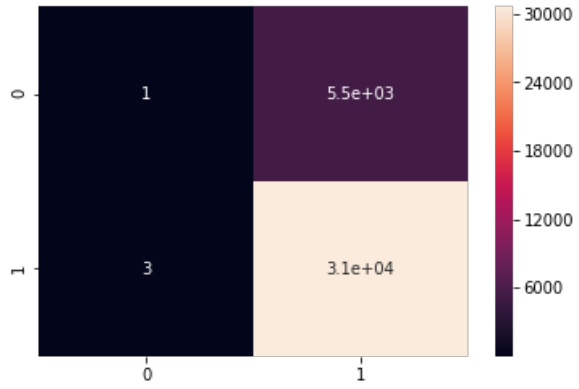
In [20]:

```
y_test_pred = clf.predict(X_te2)
cf_matrix = confusion_matrix(y_test,y_test_pred)
print(cf_matrix)
sns.heatmap(cf_matrix,annot = True)
```

```
[[    1  5458]
 [    3 30590]]
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24005e4ef98>
```



3. Summary

as mentioned in the step 4 of instructions

In [21]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["vectorizer","model","n_estimators","learning_rate","AUC"]
x.add_row(["tfidf","GBDT","100","0.01","0.492"])
x.add_row(["tfidf_w2v","GBDT","50","0.001","0.501"])

print(x)
```

```
+------------+-------+--------------+---------------+-------+
| vectorizer | model | n_estimators | learning_rate |  AUC  |
+------------+-------+--------------+---------------+-------+
|   tfidf    |  GBDT |     100      |      0.01     | 0.492 |
| tfidf_w2v  |  GBDT |      50      |     0.001     | 0.501 |
+------------+-------+--------------+---------------+-------+
```