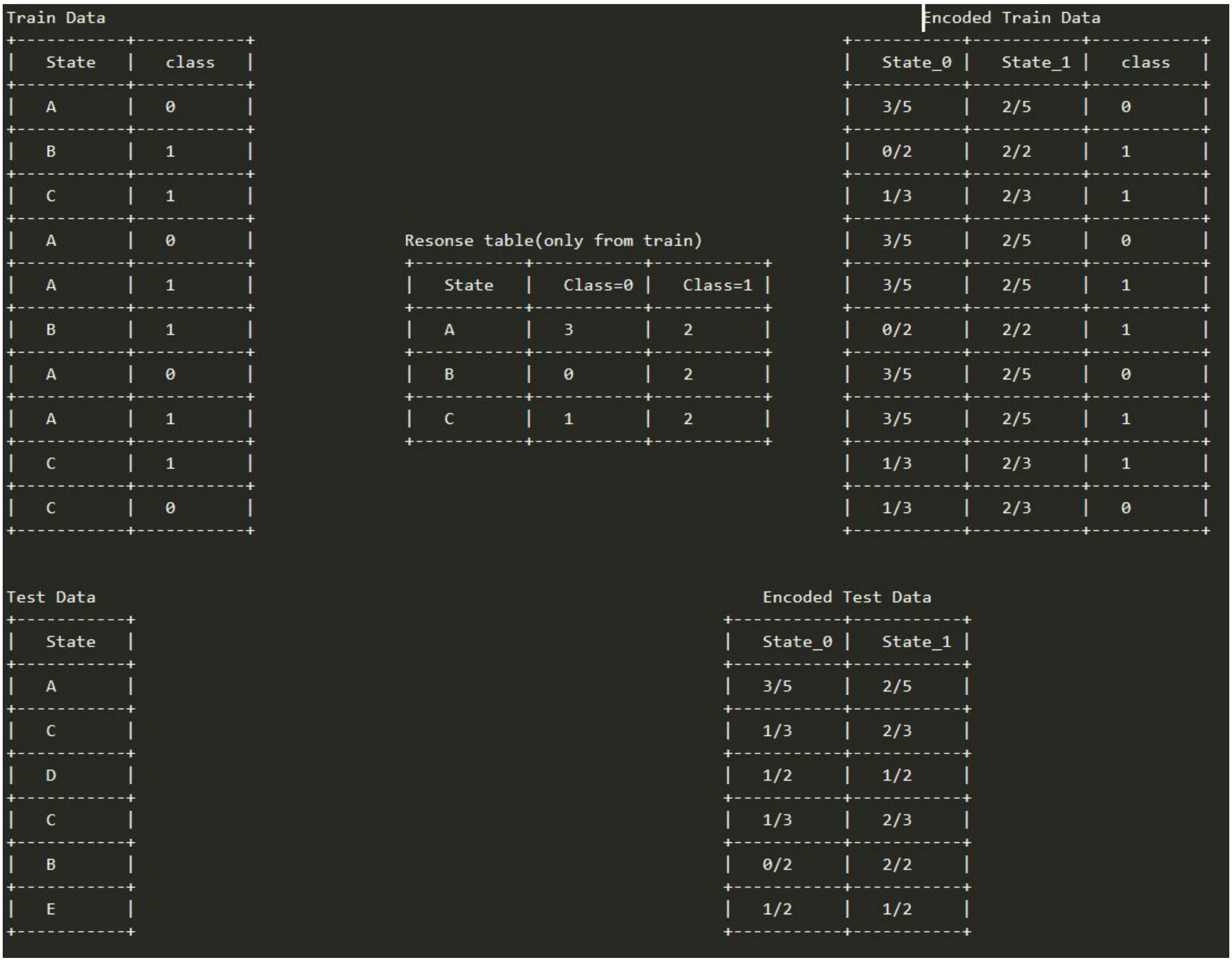
Assignment 9: GBDT

In [1]:

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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**

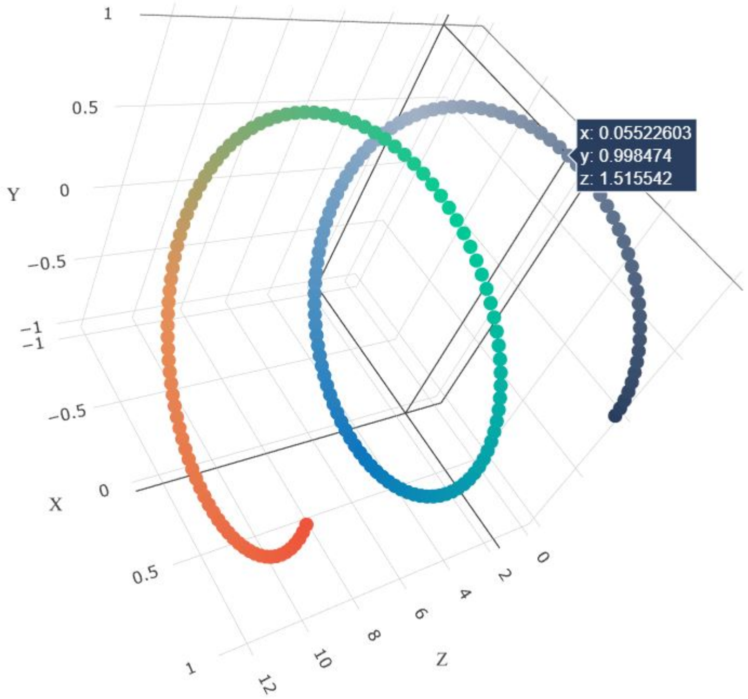
    - Set 1: categorical(instead of one hot encoding, try [response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): 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 (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/): 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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/)](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**

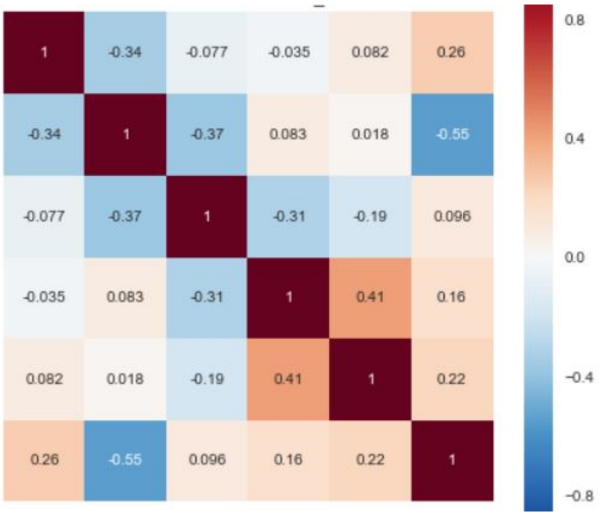
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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*

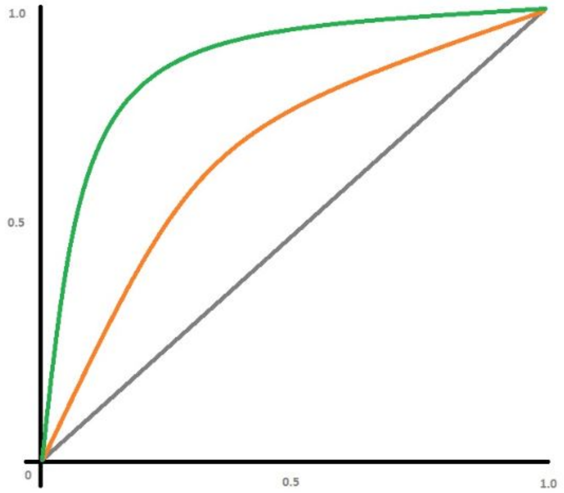
or

- 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

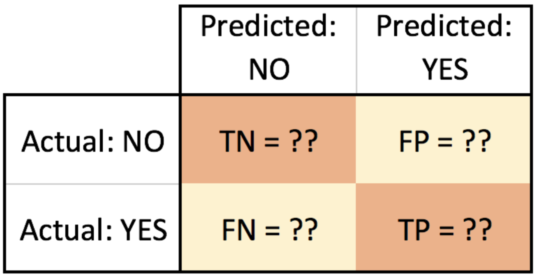


seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**

- You 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.

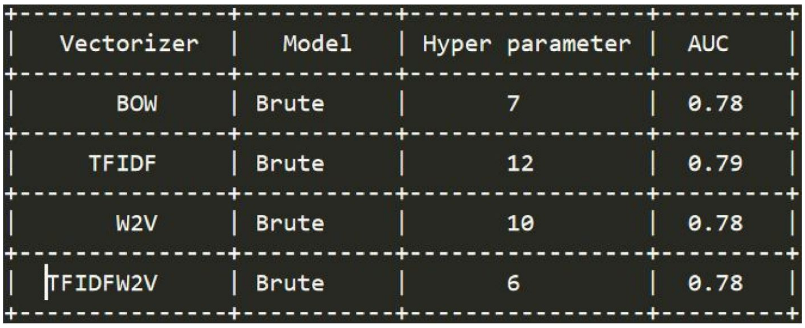


- Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points



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

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     |       12        |  0.79   |
+---------------+-----------+-----------------+---------+
|      W2V      | Brute     |       10        |  0.78   |
+---------------+-----------+-----------------+---------+
|   TFIDFW2V    | Brute     |        6        |  0.78   |
+---------------+-----------+-----------------+---------+
```

In [2]:

```
!pip install chart_studio
```

```
Collecting chart_studio
  Downloading chart_studio-1.1.0-py3-none-any.whl (64 kB)
     |████████████████████████████████| 64 kB 1.4 MB/s
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-
packages (from chart_studio) (1.15.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/pyt
hon3.7/dist-packages (from chart_studio) (1.3.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/di
st-packages (from chart_studio) (4.4.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/
dist-packages (from chart_studio) (2.23.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.2
1.1 in /usr/local/lib/python3.7/dist-packages (from requests->chart_
studio) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/
python3.7/dist-packages (from requests->chart_studio) (2021.5.30)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/p
ython3.7/dist-packages (from requests->chart_studio) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python
3.7/dist-packages (from requests->chart_studio) (2.10)
Installing collected packages: chart-studio
Successfully installed chart-studio-1.1.0
```

In [3]:

```
%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

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/

import pickle
from tqdm import tqdm
import os

import chart_studio.plotly as plotly
import plotly.graph_objs as go

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

1. GBDT (xgboost/lightgbm)

1.1 Loading Data

In [4]:

```
import pandas
data = pandas.read_csv('/content/drive/MyDrive/DT/preprocessed_data.csv', nrows=
20000)
```

In [5]:

```
data.shape
```

Out[5]:

```
(20000, 9)
```

In [6]:

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

Out[6]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_p
0	ca	mrs	grades_prek_2	

In [12]:

```
#data is highly imbalanced
unique, counts = np.unique(y, return_counts=True)
dict(zip(unique, counts))
```

Out[12]:

```
{0: 3047, 1: 16953}
```

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

In [14]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, strati
fy=y)
```

In [15]:

```
X_train.shape, X_test.shape
```

Out[15]:

```
((13400, 8), (6600, 8))
```

In [16]:

```
unique, counts = np.unique(y_train, return_counts=True)
dict(zip(unique, counts))
```

Out[16]:

```
{0: 2041, 1: 11359}
```

In [17]:

```
unique, counts = np.unique(y_test, return_counts=True)
dict(zip(unique, counts))
```

Out[17]:

```
{0: 1006, 1: 5594}
```

1.3 Make Data Model Ready: encoding eassay, and project_title

In [18]:

```
print("Before vectorizations")
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
print("="*100)
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)

# encoding eassay
vectorizer.fit(X_train['essay'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].values)

print("After TFIDF vectorization of essay")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

```
Before vectorizations
(13400, 8) (13400,)
(6600, 8) (6600,)
====================================================================
================================
After TFIDF vectorization of essay
(13400, 5000) (13400,)
(6600, 5000) (6600,)
====================================================================
================================
```

1.4 Make Data Model Ready: encoding numerical, categorical features

In [19]:

```
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_test_price_norm = normalizer.transform(X_test['price'].values.reshape (1,-1))
X_train_price_norm =X_train_price_norm.reshape(-1,1)
X_test_price_norm = X_test_price_norm.reshape(-1,1)
print("After vectorizations")
#print(X_train_price_norm_1.shape, y_train.shape)
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(13400, 1) (13400,)
(6600, 1) (6600,)
====================================================================
================================
```

In [20]:

```
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead: # array=[105.
22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.re
shape(1,-1))
X_train_teacher_number_of_previously_posted_projects_norm = normalizer.transform
(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_test_teacher_number_of_previously_posted_projects_norm = normalizer.transform(
X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_train_teacher_number_of_previously_posted_projects_norm = X_train_teacher_numb
er_of_previously_posted_projects_norm.reshape(-1,1)
X_test_teacher_number_of_previously_posted_projects_norm = X_test_teacher_number
_of_previously_posted_projects_norm.reshape(-1,1)
print("After vectorizations")
print(X_train_teacher_number_of_previously_posted_projects_norm.shape, y_train.s
hape)
print(X_test_teacher_number_of_previously_posted_projects_norm.shape, y_test.sha
pe)
print("="*100)
```

```
After vectorizations
(13400, 1) (13400,)
(6600, 1) (6600,)
====================================================================
================================
```

In [21]:

```
import nltk
nltk.download('all')
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_esp.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
[nltk_data]    | Downloading package crubadan to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/crubadan.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]    | Downloading package dolch to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dolch.zip.
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/europarl_raw.zip.
[nltk_data]    | Downloading package floresta to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/floresta.zip.
[nltk_data]    | Downloading package framenet_v15 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v15.zip.
[nltk_data]    | Downloading package framenet_v17 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v17.zip.
[nltk_data]    | Downloading package gazetteers to /root/nltk_dat
a...
[nltk_data]    |   Unzipping corpora/gazetteers.zip.
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/genesis.zip.
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gutenberg.zip.
[nltk_data]    | Downloading package ieer to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ieer.zip.
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/inaugural.zip.
[nltk_data]    | Downloading package indian to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/indian.zip.
[nltk_data]    | Downloading package jeita to /root/nltk_data...
[nltk_data]    | Downloading package kimmo to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/kimmo.zip.
[nltk_data]    | Downloading package knbc to /root/nltk_data...
[nltk_data]    | Downloading package lin_thesaurus to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/lin_thesaurus.zip.
[nltk_data]    | Downloading package mac_morpho to /root/nltk_dat
a...
[nltk_data]    |   Unzipping corpora/mac_morpho.zip.
[nltk_data]    | Downloading package machado to /root/nltk_data...
[nltk_data]    | Downloading package masc_tagged to /root/nltk_dat
a...
[nltk_data]    | Downloading package moses_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/moses_sample.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/movie_reviews.zip.
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package nombank.1.0 to /root/nltk_dat
a...
[nltk_data]    | Downloading package nps_chat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/nps_chat.zip.
[nltk_data]    | Downloading package omw to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/omw.zip.
[nltk_data]    | Downloading package opinion_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/opinion_lexicon.zip.
[nltk_data]    | Downloading package paradigms to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/paradigms.zip.
[nltk_data]    | Downloading package pil to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/pil.zip.
[nltk_data]    | Downloading package pl196x to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/pl196x.zip.
[nltk_data]    | Downloading package ppattach to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ppattach.zip.
[nltk_data]    | Downloading package problem_reports to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/problem_reports.zip.
[nltk_data]    | Downloading package propbank to /root/nltk_data...
[nltk_data]    | Downloading package ptb to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ptb.zip.
[nltk_data]    | Downloading package product_reviews_1 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/product_reviews_1.zip.
[nltk_data]    | Downloading package product_reviews_2 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/product_reviews_2.zip.
[nltk_data]    | Downloading package pros_cons to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/pros_cons.zip.
[nltk_data]    | Downloading package qc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/qc.zip.
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    | Downloading package rte to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/rte.zip.
[nltk_data]    | Downloading package semcor to /root/nltk_data...
```

```
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/senseval.zip.
[nltk_data]    | Downloading package sentiwordnet to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/sentiwordnet.zip.
[nltk_data]    | Downloading package sentence_polarity to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/sentence_polarity.zip.
[nltk_data]    | Downloading package shakespeare to /root/nltk_dat
a...
[nltk_data]    |   Unzipping corpora/shakespeare.zip.
[nltk_data]    | Downloading package sinica_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/sinica_treebank.zip.
[nltk_data]    | Downloading package smultron to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/smultron.zip.
[nltk_data]    | Downloading package state_union to /root/nltk_dat
a...
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package subjectivity to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/subjectivity.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package switchboard to /root/nltk_dat
a...
[nltk_data]    |   Unzipping corpora/switchboard.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/twitter_samples.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package verbnet to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet.zip.
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet3.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet.zip.
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_dat
a...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/ycoe.zip.
[nltk_data]    | Downloading package rslp to /root/nltk_data...
[nltk_data]    |   Unzipping stemmers/rslp.zip.
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Unzipping tokenizers/punkt.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package sample_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/sample_grammars.zip.
[nltk_data]    | Downloading package spanish_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/spanish_grammars.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package large_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/large_grammars.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/word2vec_sample.zip.
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package mte_teip5 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/mte_teip5.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru t
o
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package perluniprops to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping misc/perluniprops.zip.
[nltk_data]    | Downloading package nonbreaking_prefixes to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/nonbreaking_prefixes.zip.
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package porter_test to /root/nltk_dat
a...
```

```
[nltk_data]    |   Unzipping stemmers/porter_test.zip.
[nltk_data]    | Downloading package wmt15_eval to /root/nltk_dat
a...
[nltk_data]    |   Unzipping models/wmt15_eval.zip.
[nltk_data]    | Downloading package mwa_ppdb to /root/nltk_data...
[nltk_data]    |   Unzipping misc/mwa_ppdb.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

Out[21]:

```
True
```

In [22]:

```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
sentiment_neg, sentiment_neu, sentiment_pos, sentiment_comp = [], [], [], []

for sentence in tqdm(data['essay'].values):
  sid = SIA()
  sentiment_dict=sid.polarity_scores(sentence)
  sentiment_neg.append(sentiment_dict['neg'])
  sentiment_neu.append(sentiment_dict['neu'])
  sentiment_pos.append(sentiment_dict['pos'])
  sentiment_comp.append(sentiment_dict['compound'])
```

```
100%|██████████| 20000/20000 [03:36<00:00, 92.37it/s]
```

In [23]:

```python
import numpy as np
neg = np.array((sentiment_neg))
pos = np.array((sentiment_pos))
neu = np.array((sentiment_neu))
comp = np.array((sentiment_comp))
```

In [24]:

```python
X_train_neg   = neg[0:X_train.shape[0]].reshape(-1,1)
X_train_pos   = pos[0:X_train.shape[0]].reshape(-1,1)
X_train_comp  = comp[0:X_train.shape[0]].reshape(-1,1)
X_train_neu   = neu[0:X_train.shape[0]].reshape(-1,1)
```

In [25]:

```python
X_test_neg   = neg[X_train.shape[0]:].reshape(-1,1)
X_test_pos   = pos[X_train.shape[0]:].reshape(-1,1)
X_test_comp  = comp[X_train.shape[0]:].reshape(-1,1)
X_test_neu   = neu[X_train.shape[0]:].reshape(-1,1)
```

In [26]:

```
#https://stackoverflow.com/questions/11869910/pandas-filter-rows-of-dataframe-wi
th-operator-chaining
def mask(df, key, value):
    return df[df[key] == value]

def get_response(data,data_label):
    cat_values = np.unique(data).tolist()
    df = pd.DataFrame({'feature':data.tolist(),'label':data_label.tolist()})
    pd.DataFrame.mask = mask

    accep = {}
    reject={}
    prob_neg = {}
    prob_pos={}

    for i in cat_values:
        count_0      = len(df.mask('feature', i).mask('label', 0))
        count_1      = len(df.mask('feature', i).mask('label', 1))
        total        = count_0 + count_1
        prob_0       = count_0/total
        prob_1       = count_1/total
        accep[i]     = count_1
        reject[i]    = count_0
        prob_neg[i] = prob_0
        prob_pos[i] = prob_1

    return prob_neg, prob_pos
```

In [27]:

```
cat_0_train = get_response(X_train['clean_categories'],y_train)[0]
cat_1_train = get_response(X_train['clean_categories'],y_train)[1]
```

In [28]:

```
subcat_0_train = get_response(X_train['clean_subcategories'],y_train)[0]
subcat_1_train = get_response(X_train['clean_subcategories'],y_train)[1]
```

In [29]:

```
state_0_train = get_response(X_train['school_state'],y_train)[0]
state_1_train = get_response(X_train['school_state'],y_train)[1]
```

In [30]:

```
prefix_0_train = get_response(X_train['teacher_prefix'],y_train)[0]
prefix_1_train = get_response(X_train['teacher_prefix'],y_train)[1]
```

In [31]:

```
grad_cat_0_train = get_response(X_train['project_grade_category'],y_train)[0]
grad_cat_1_train = get_response(X_train['project_grade_category'],y_train)[1]
```

In [32]:

```
cat_neg_train = []
cat_pos_train = []

for i in X_train['clean_categories']:
    cat_neg_train.append(cat_0_train[i])
    cat_pos_train.append(cat_1_train[i])

cat_neg_train = np.array(cat_neg_train).reshape(-1, 1)
cat_pos_train = np.array(cat_pos_train).reshape(-1, 1)
```

In [33]:

```
subcat_neg_train = []
subcat_pos_train = []
for i in X_train['clean_subcategories']:
    subcat_neg_train.append(subcat_0_train[i])
    subcat_pos_train.append(subcat_1_train[i])
X_train['subcat_0'] = subcat_neg_train
X_train['subcat_1'] = subcat_pos_train

subcat_neg_train = np.array(subcat_neg_train).reshape(-1, 1)
subcat_pos_train = np.array(subcat_pos_train).reshape(-1, 1)
```

In [34]:

```
state_neg_train = []
state_pos_train = []
for i in X_train['school_state']:
    state_neg_train.append(state_0_train[i])
    state_pos_train.append(state_1_train[i])
X_train['state_0'] = state_neg_train
X_train['state_1'] = state_pos_train

state_neg_train = np.array(state_neg_train).reshape(-1, 1)
state_pos_train = np.array(state_pos_train).reshape(-1, 1)
```

In [35]:

```
prefix_neg_train = []
prefix_pos_train = []
for i in X_train['teacher_prefix']:
    prefix_neg_train.append(prefix_0_train[i])
    prefix_pos_train.append(prefix_1_train[i])
X_train['prefix_0'] = prefix_neg_train
X_train['prefix_1'] = prefix_pos_train

prefix_neg_train = np.array(prefix_neg_train).reshape(-1, 1)
prefix_pos_train = np.array(prefix_pos_train).reshape(-1, 1)
```

In [36]:

```
grade_neg_train = []
grade_pos_train = []
for i in X_train['project_grade_category']:
    grade_neg_train.append(grad_cat_0_train[i])
    grade_pos_train.append(grad_cat_1_train[i])
X_train['grade_0'] = grade_neg_train
X_train['grade_1'] = grade_pos_train

grade_neg_train = np.array(grade_neg_train).reshape(-1, 1)
grade_pos_train = np.array(grade_pos_train).reshape(-1, 1)
```

In [37]:

```
cat_neg_test = []
cat_pos_test = []

for i in X_test['clean_categories']:
  if i in cat_0_train.keys():
    cat_neg_test.append(cat_0_train[i])
    cat_pos_test.append(cat_1_train[i])
  elif i in cat_1_train.keys():
    cat_neg_test.append(cat_0_train[i])
    cat_pos_test.append(cat_1_train[i])
  else:
    cat_neg_test.append(0.5)
    cat_pos_test.append(0.5)

cat_neg_test_a = np.array((cat_neg_test)).reshape(-1, 1)
cat_pos_test_a = np.array((cat_pos_test)).reshape(-1, 1)
```

In [38]:

```
subcat_neg_test = []
subcat_pos_test = []

for i in X_test['clean_subcategories']:
  if i in subcat_0_train.keys():
    subcat_neg_test.append(subcat_0_train[i])
    subcat_pos_test.append(subcat_1_train[i])
  elif i in subcat_1_train.keys():
    subcat_neg_test.append(subcat_0_train[i])
    suncat_pos_test.append(subcat_1_train[i])
  else:
    subcat_neg_test.append(0.5)
    subcat_pos_test.append(0.5)

subcat_neg_test = np.array((subcat_neg_test)).reshape(-1, 1)
subcat_pos_test = np.array((subcat_pos_test)).reshape(-1, 1)
```

In [39]:

```
state_0_test = []
state_1_test = []

for i in X_test['school_state']:
  if i in subcat_0_train.keys():
    state_0_test.append(state_0_train[i])
    state_1_test.append(state_1_train[i])
  elif i in subcat_1_train.keys():
    state_0_test.append(state_0_train[i])
    state_1_test.append(state_1_train[i])
  else:
    state_0_test.append(0.5)
    state_1_test.append(0.5)

state_0_test = np.array((state_0_test)).reshape(-1, 1)
state_1_test = np.array((state_1_test)).reshape(-1, 1)
```

In [40]:

```
prefix_0_test = []
prefix_1_test = []

for i in X_test['teacher_prefix']:
  if i in subcat_0_train.keys():
    prefix_0_test.append(prefix_0_train[i])
    prefix_1_test.append(prefix_1_train[i])
  elif i in subcat_1_train.keys():
    prefix_0_test.append(prefix_0_train[i])
    prefix_1_test.append(prefix_1_train[i])
  else:
    prefix_0_test.append(0.5)
    prefix_1_test.append(0.5)

prefix_0_test = np.array((prefix_0_test)).reshape(-1, 1)
prefix_1_test = np.array((prefix_1_test)).reshape(-1, 1)
```

In [41]:

```
grad_cat_0_test = []
grad_cat_1_test = []

for i in X_test['project_grade_category']:
  if i in subcat_0_train.keys():
    grad_cat_0_test.append(grad_cat_0_train[i])
    grad_cat_1_test.append(grad_cat_1_train[i])
  elif i in subcat_1_train.keys():
    grad_cat_0_test.append(grad_cat_0_train[i])
    grad_cat_1_test.append(grad_cat_1_train[i])
  else:
    grad_cat_0_test.append(0.5)
    grad_cat_1_test.append(0.5)

grad_cat_0_test = np.array((grad_cat_0_test)).reshape(-1, 1)
grad_cat_1_test = np.array((grad_cat_1_test)).reshape(-1, 1)
```

In [42]:

```
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_tfidf, cat_neg_train, cat_pos_train, grade_neg_train, grade_pos_train, prefix_neg_train, prefix_pos_train, state_neg_train, state_pos_train, subcat_neg_train, subcat_pos_train, X_train_price_norm, X_train_teacher_number_of_previously_posted_projects_norm, X_train_neg, X_train_pos, X_train_comp, X_train_neu)).tocsr()
X_te = hstack((X_test_essay_tfidf, cat_neg_test_a, cat_pos_test_a, grad_cat_0_test,grad_cat_1_test, prefix_0_test, prefix_1_test, subcat_neg_test,
               subcat_pos_test, state_0_test, state_1_test, X_test_price_norm,
               X_test_teacher_number_of_previously_posted_projects_norm, X_test_neg, X_test_pos, X_test_comp, X_test_neu)).tocsr()
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(13400, 5016) (13400,)
(6600, 5016) (6600,)
====================================================================================================
```

Hyper-parameter Tuning

In [76]:

```
#RandomizedSearchCV

from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.model_selection import RandomizedSearchCV

gbdt = LGBMClassifier(is_unbalance = True)

learning_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]
n_estimators  = [5, 10, 50, 75, 100, 200]

#grid_params = {'n_estimators': [100, 200, 500, 1000], 'max_depth':[1, 5, 10, 5
0]}
grid_params = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3], 'n_estimat
ors':[5,10,50, 75, 100, 200]}

rs = RandomizedSearchCV(gbdt, grid_params, cv=3, scoring='roc_auc', return_train
_score=True)

rs.fit(X_tr, y_train)
```

Out[76]:

```
RandomizedSearchCV(cv=3, error_score=nan,
                   estimator=LGBMClassifier(boosting_type='gbdt',
                                            class_weight=None,
                                            colsample_bytree=1.0,
                                            importance_type='split',
                                            is_unbalance=True,
                                            learning_rate=0.1, max_d
epth=-1,
                                            min_child_samples=20,
                                            min_child_weight=0.001,
                                            min_split_gain=0.0,
                                            n_estimators=100, n_jobs
=-1,
                                            num_leaves=31, objective
=None,
                                            random_state=None, reg_a
lpha=0.0,
                                            reg_lambda=0.0, silent=T
rue,
                                            subsample=1.0,
                                            subsample_for_bin=20000
0,
                                            subsample_freq=0),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'learning_rate': [0.0001, 0.
001, 0.01,
                                                          0.1, 0.2,
0.3],
                                        'n_estimators': [5, 10, 50,
75, 100,
                                                         200]},
                   pre_dispatch='2*n_jobs', random_state=None, refit
=True,
                   return_train_score=True, scoring='roc_auc', verbo
se=0)
```

In [90]:

```
print('Best score: ',rs.best_score_)
print('k value with best score: ',rs.best_params_)
print('='*75)
print('Train AUC scores')
print(rs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(rs.cv_results_['mean_test_score'])
rs.estimator
```

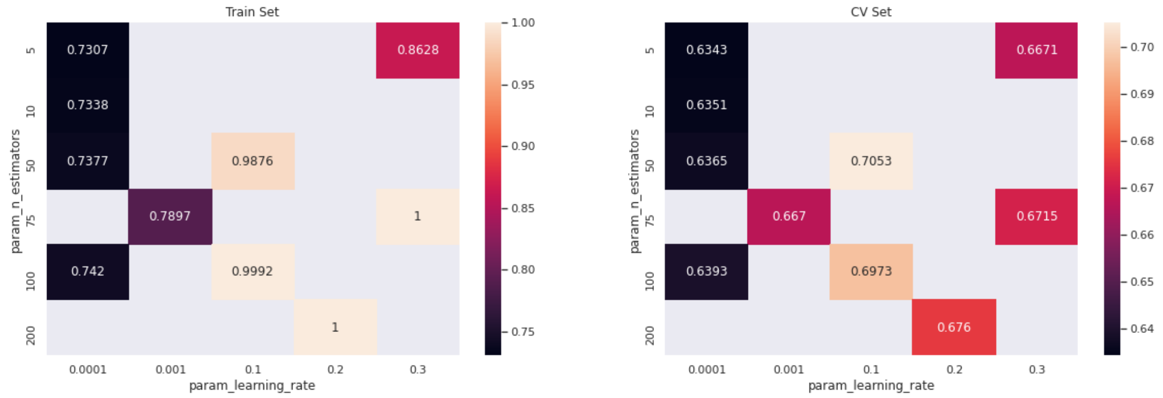
```
Best score:  0.7052646066717841
k value with best score:  {'n_estimators': 50, 'learning_rate': 0.1}
====================================================================
=======
Train AUC scores
[0.98757988 0.73773119 0.99999796 0.86277545 0.73067481 0.78971489
 0.99917231 0.73379614 0.74198226 1.        ]
CV AUC scores
[0.70526461 0.63654904 0.67145848 0.66711339 0.63431596 0.66696271
 0.69733459 0.63511369 0.6392589  0.67600915]
```

Out[90]:

```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_by
tree=1.0,
               importance_type='split', is_unbalance=True, learning_
rate=0.1,
               max_depth=-1, min_child_samples=20, min_child_weight=
0.001,
               min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_
leaves=31,
               objective=None, random_state=None, reg_alpha=0.0, reg
_lambda=0.0,
               silent=True, subsample=1.0, subsample_for_bin=200000,
               subsample_freq=0)
```

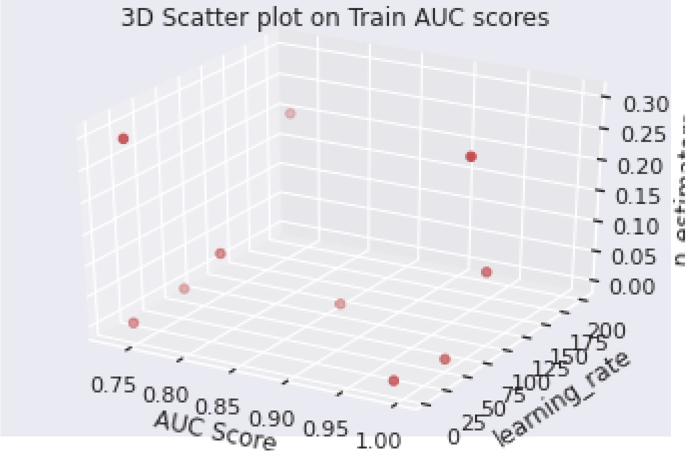
In [91]:

```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(rs.cv_results_).groupby(['param_n_estimators', 'param
_learning_rate']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



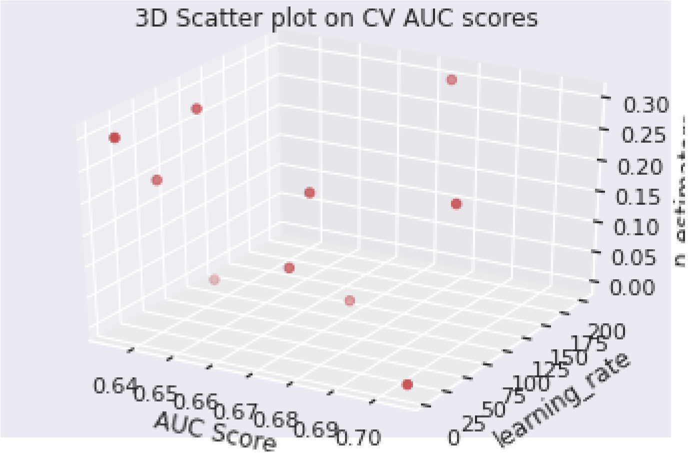
In [98]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

g1 = list(rs.cv_results_['mean_train_score'])     #Train AUC Score
g2 = [5, 10, 50,    100,   75, 50, 100,   200,    5,  75] # n_estimaters
g3 = [0.0001, 0.0001, 0.0001, 0.0001,     0.001, 0.1, .1,     .2,  .3, .3]
# learning rate


ax.scatter(g1, g2, g3, c='r', marker='o')

ax.set_xlabel('AUC Score')
ax.set_ylabel('learning_rate')
ax.set_zlabel('n_estimators')

plt.title('3D Scatter plot on Train AUC scores')
plt.show()
```



In [99]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

g1 = list(rs.cv_results_['mean_test_score'])                  #Train AUC Score
g2 = [5, 100, 100,    50,    50, 75, 100,       5,   75, 200] # n_estimaters
g3 = [0.0001, 0.0001, 0.01, 0.1,     0.2, 0.2, .2,      .3,   .3, .3]
# learning rate

ax.scatter(g1, g2, g3, c='r', marker='o')

ax.set_xlabel('AUC Score')
ax.set_ylabel('learning_rate')
ax.set_zlabel('n_estimators')

plt.title('3D Scatter plot on CV AUC scores')
plt.show()
```



In [94]:

```
learning_r = rs.best_params_['learning_rate']
n_est = rs.best_params_['n_estimators']
```

In [95]:

```
learning_r, n_est
```

Out[95]:

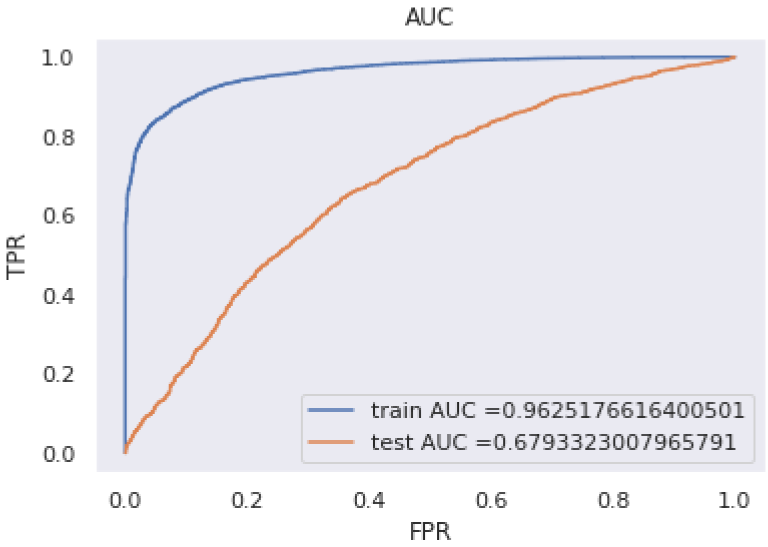
```
(0.1, 50)
```

In [96]:

```
def pred_prob(clf, data):
    y_pred = []
    y_pred = clf.predict_proba(data)[:,1]
    return y_pred
```

In [97]:

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

model = LGBMClassifier(learning_rate= learning_r, n_estimators = n_est, class_we
ight=None, is_unbalance = True)

model.fit(X_tr,y_train)

y_train_pred = pred_prob(model,X_tr)
y_test_pred = pred_prob(model,X_te)

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

plt.close
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("AUC")
plt.grid()
plt.show()
```



In [100]:

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

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

In [101]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.8032467957362666 for threshold 0.
476
Train confusion matrix
[[1878  163]
 [1443 9916]]
```

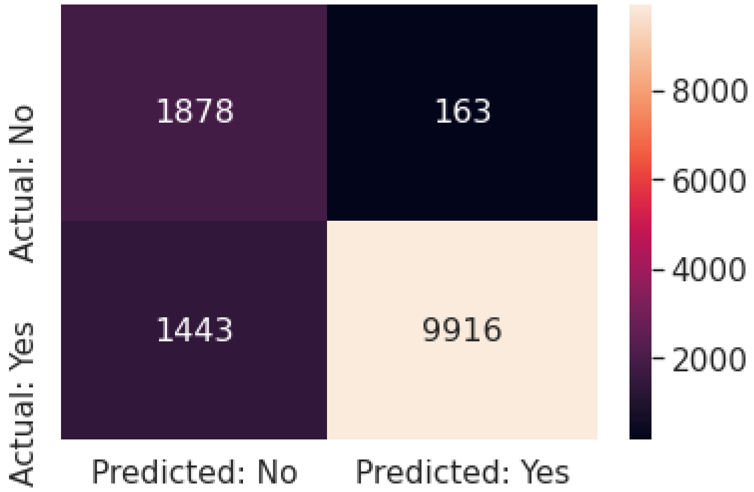
In [102]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_
best_t(y_train_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Pr
edicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt=
'g')
```

```
Train data confusion matrix
```

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb94c792d90>
```



In [103]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 932   74]
 [4684  910]]
```

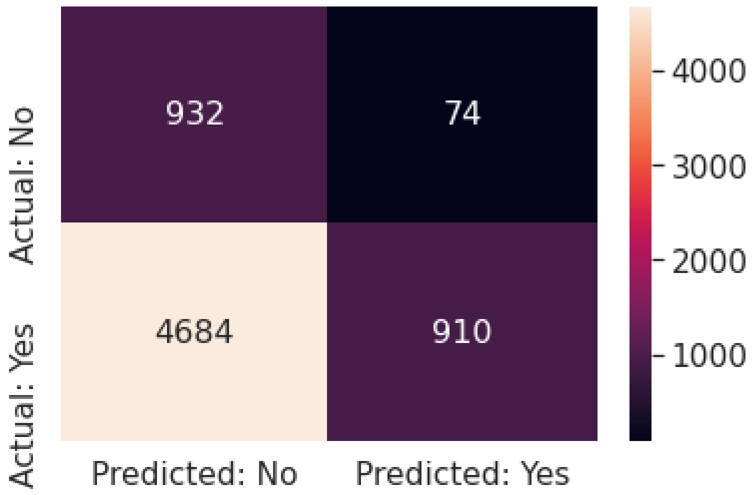
In [104]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True, annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb93b215910>
```



GBDT with TFIDF W2V

In [105]:

```
# 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('/content/drive/MyDrive/DT/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [106]:

```
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)

# encoding eassay
vectorizer.fit(X_train['essay'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].values)

print("After TFIDF vectorization of essay")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

```
After TFIDF vectorization of essay
(13400, 5000) (13400,)
(6600, 5000) (6600,)
====================================================================
================================
```

In [107]:

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

In [108]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(X_train['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 val
ue((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
    train_tfidf_w2v_essays.append(vector)

print(len(train_tfidf_w2v_essays))
print(len(train_tfidf_w2v_essays[0]))
```

```
100%|██████████| 13400/13400 [00:31<00:00, 424.68it/s]

13400
300
```

In [109]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in
 this list
for sentence in tqdm(X_test['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 val
ue((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
    test_tfidf_w2v_essays.append(vector)

print(len(test_tfidf_w2v_essays))
print(len(test_tfidf_w2v_essays[0]))
```

```
100%|██████████| 6600/6600 [00:15<00:00, 427.84it/s]

6600
300
```

In [110]:

```
len(test_tfidf_w2v_essays), len(train_tfidf_w2v_essays)
```

Out[110]:

```
(6600, 13400)
```

In [111]:

```
train_tfidf_w2v_essays1 = np.array(train_tfidf_w2v_essays)
test_tfidf_w2v_essays1 = np.array(test_tfidf_w2v_essays)
```

In [112]:

```
from scipy.sparse import coo_matrix, hstack
```

In [113]:

```
from scipy.sparse import hstack
X_tr = hstack((coo_matrix(train_tfidf_w2v_essays1), cat_neg_train, cat_pos_train
, grade_neg_train, grade_pos_train, prefix_neg_train,
               prefix_pos_train, state_neg_train, state_pos_train, subcat_neg_tr
ain, subcat_pos_train, X_train_price_norm,
               X_train_teacher_number_of_previously_posted_projects_norm, X_trai
n_neg, X_train_pos, X_train_comp, X_train_neu)).tocsr()
X_te = hstack((coo_matrix(test_tfidf_w2v_essays1), cat_neg_test_a, cat_pos_test_
a, grad_cat_0_test,grad_cat_1_test, prefix_0_test, prefix_1_test, subcat_neg_tes
t,
               subcat_pos_test, state_0_test, state_1_test, X_test_price_norm,
               X_test_teacher_number_of_previously_posted_projects_norm, X_test_
neg, X_test_pos, X_test_comp, X_test_neu)).tocsr()
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(13400, 316) (13400,)
(6600, 316) (6600,)
====================================================================
================================
```

In [115]:

```
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV

rf = LGBMClassifier(is_unbalance=True)

#grid_params = {'n_estimators': [100, 200, 500, 1000], 'max_depth':[1, 5, 10, 5
0]}
grid_params = {'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3], 'n_estimat
ors':[5,10,50, 75, 100, 200]}

rs = RandomizedSearchCV(rf,grid_params ,cv=3, scoring='roc_auc', return_train_sc
ore=True, random_state=100)
rs.fit(X_tr, y_train)
```

Out[115]:

```
RandomizedSearchCV(cv=3, error_score=nan,
                   estimator=LGBMClassifier(boosting_type='gbdt',
                                            class_weight=None,
                                            colsample_bytree=1.0,
                                            importance_type='split',
                                            is_unbalance=True,
                                            learning_rate=0.1, max_d
epth=-1,
                                            min_child_samples=20,
                                            min_child_weight=0.001,
                                            min_split_gain=0.0,
                                            n_estimators=100, n_jobs
=-1,
                                            num_leaves=31, objective
=None,
                                            random_state=None, reg_a
lpha=0.0,
                                            reg_lambda=0.0, silent=T
rue,
                                            subsample=1.0,
                                            subsample_for_bin=20000
0,
                                            subsample_freq=0),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'learning_rate': [0.0001, 0.
001, 0.01,
                                                          0.1, 0.2,
0.3],
                                        'n_estimators': [5, 10, 50,
75, 100,
                                                         200]},
                   pre_dispatch='2*n_jobs', random_state=100, refit=
True,
                   return_train_score=True, scoring='roc_auc', verbo
se=0)
```

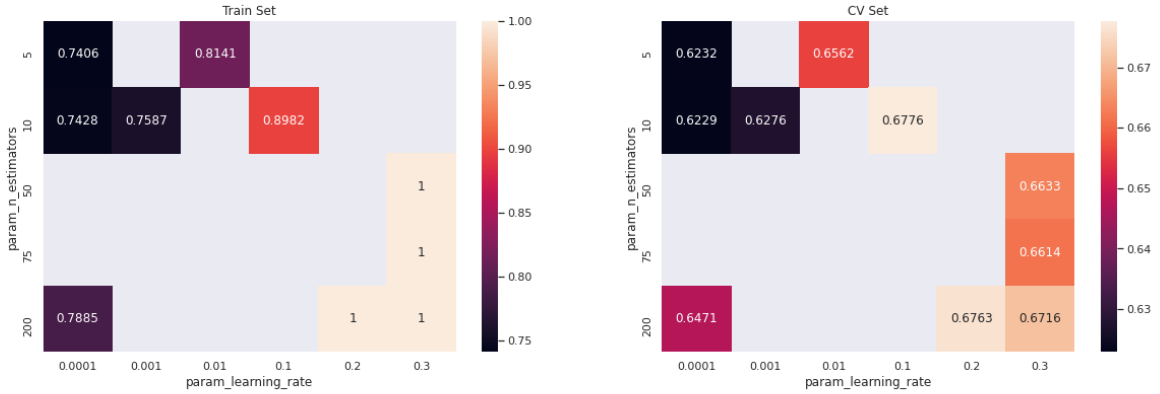
In [116]:

```
print('Best score: ',rs.best_score_)
print('k value with best score: ',rs.best_params_)
print('='*75)
print('Train AUC scores')
print(rs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(rs.cv_results_['mean_test_score'])
```

```
Best score:  0.6776221264723885
k value with best score:  {'n_estimators': 10, 'learning_rate': 0.1}
====================================================================
=======
Train AUC scores
[0.74064379 0.8982081  0.99999114 1.         0.75869098 0.8141041
 1.         1.         0.78847258 0.74279603]
CV AUC scores
[0.62320512 0.67762213 0.6632714  0.6715534  0.6275922  0.65621476
 0.67632686 0.66138222 0.6470958  0.62291759]
```

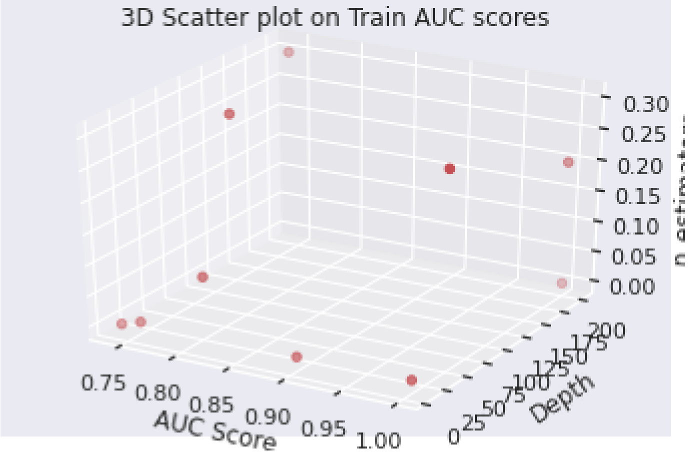
In [117]:

```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(rs.cv_results_).groupby(['param_n_estimators', 'param
_learning_rate']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



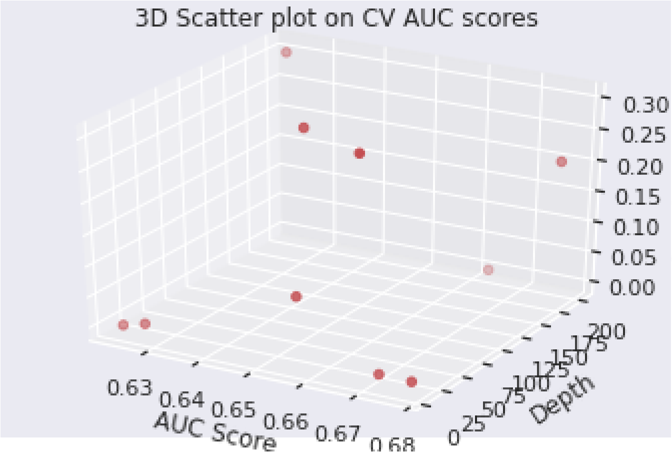
In [118]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

g1 = list(rs.cv_results_['mean_train_score'])      #Train AUC Score
g2 = [5, 10, 200, 10, 5, 10, 200, 50, 75, 200] # n_estimaters
g3 = [0.0001, 0.0001, 0.0001, 0.001, 0.01, 0.1, .2, .3, .3, .3]              # learning rate


ax.scatter(g1, g2, g3, c='r', marker='o')

ax.set_xlabel('AUC Score')
ax.set_ylabel('Depth')
ax.set_zlabel('n_estimators')

plt.title('3D Scatter plot on Train AUC scores')
plt.show()
```



In [119]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

g1 = list(rs.cv_results_['mean_test_score'])      #Train AUC Score
g2 = [5, 10, 200, 10, 5, 10, 200, 50, 75, 200] # n_estimaters
g3 = [0.0001, 0.0001, 0.0001, 0.001, 0.01, 0.1, .2, .3, .3, .3]            # learning rate

ax.scatter(g1, g2, g3, c='r', marker='o')

ax.set_xlabel('AUC Score')
ax.set_ylabel('Depth')
ax.set_zlabel('n_estimators')

plt.title('3D Scatter plot on CV AUC scores')
plt.show()
```



In [122]:

```
learning_r = rs.best_params_['learning_rate']
n_est = rs.best_params_['n_estimators']
```

In [123]:

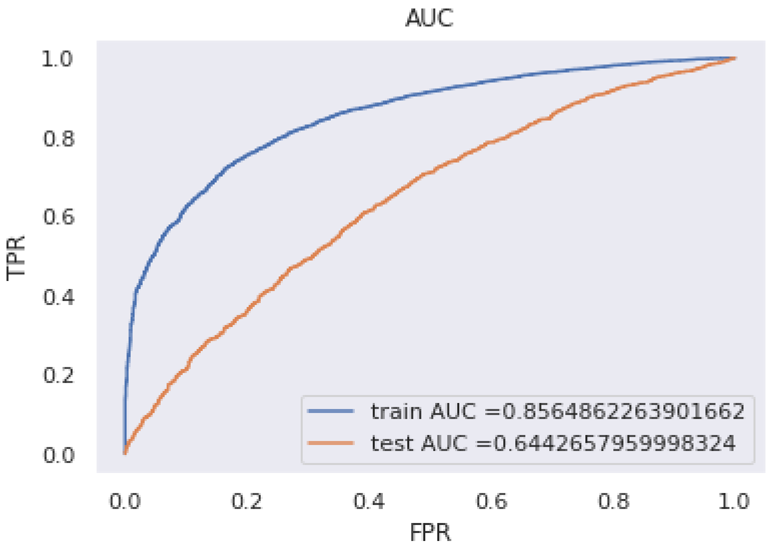
```
learning_r, n_est
```

Out[123]:

```
(0.1, 10)
```

In [124]:

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

model = LGBMClassifier(learning_rate= learning_r, n_estimators = n_est, is_unbal
ance=True)

model.fit(X_tr,y_train)

y_train_pred = pred_prob(model,X_tr)
y_test_pred = pred_prob(model,X_te)

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

plt.close
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("AUC")
plt.grid()
plt.show()
```



In [125]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.6042736284027598 for threshold 0.
609
Train confusion matrix
[[1665  376]
 [2945 8414]]
```

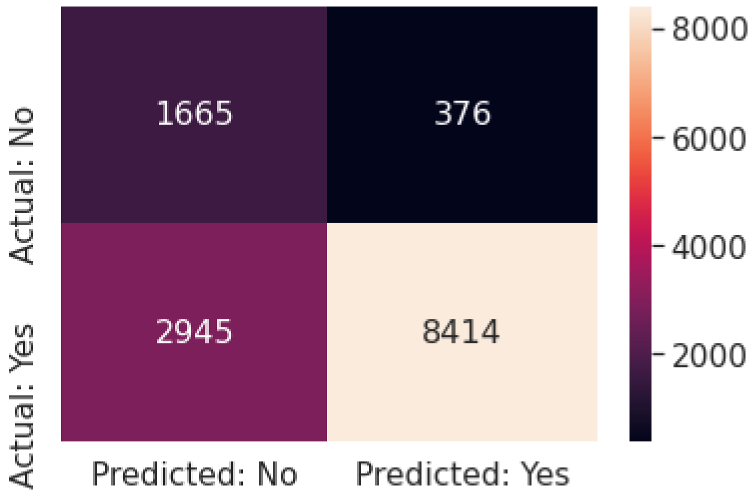
In [126]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[126]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb94b2ed890>
```



In [127]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 853  153]
 [3926 1668]]
```

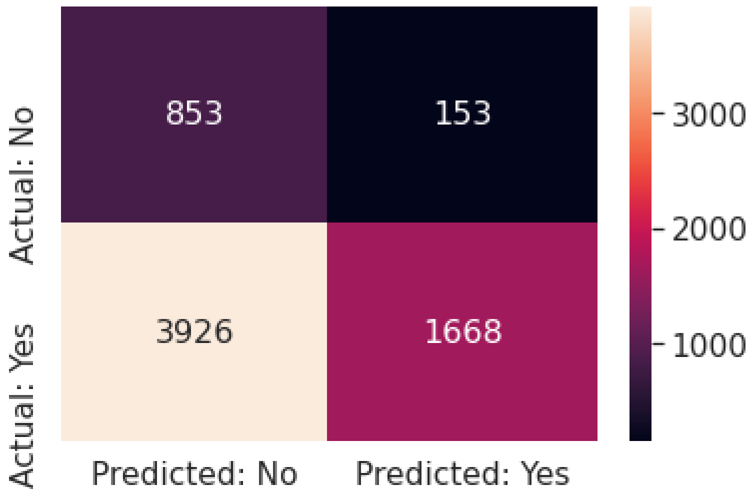
In [128]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[128]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb93b260b50>
```



3. Summary

In [130]:

```
from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 instal
l prettytable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperparameters(n_estimators,learning_r
ate)", "Test AUC"]

x.add_row(["TFIDF", "GBDT", "(0.1, 50)",  0.67])
x.add_row(["TFIDF W2V", "GBDT", "(0.1, 10)", 0.64])

print(x)
```

```
+------------+-------+--------------------------------------------+
----------+
| Vectorizer | Model | Hyperparameters(n_estimators,learning_rate) |
Test AUC |
+------------+-------+--------------------------------------------+
----------+
|   TFIDF    |  GBDT |                 (0.1, 50)                  |
0.67    |
| TFIDF W2V  |  GBDT |                 (0.1, 10)                  |
0.64    |
+------------+-------+--------------------------------------------+
----------+
```