11_Assignment_GBDT_Instructions

May 7, 2021

0.1 Importing Packages

[42]:
```
import math
import numpy as np
import pandas as pd
import re
import pickle
from tqdm import tqdm
import nltk
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud, STOPWORDS
from tqdm import tqdm

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from collections import Counter
from sklearn.preprocessing import Normalizer
from scipy.sparse import hstack
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
import lightgbm as lgb

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import plotly
import plotly.offline as offline
import plotly.graph_objs as go
from mpl_toolkits.mplot3d import Axes3D
offline.init_notebook_mode()
```

[2]:
```
data = pd.read_csv('preprocessed_data.csv')
```

[3]:
```
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)
X.head(1)
```

[3]:
```
  school_state teacher_prefix project_grade_category  \
0           ca            mrs          grades_prek_2

   teacher_number_of_previously_posted_projects clean_categories  \
0                                            53     math_science

                 clean_subcategories  \
0  appliedsciences health_lifescience

                                                 essay   price
0  i fortunate enough use fairy tale stem kits cl…  725.05
```

0.2 Splitting Data

[4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪stratify=y)
```

0.3 Applying TFIDF on Essay Feature

[5]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vectorizer_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,4),␣
↪max_features=5000)
vectorizer_tfidf.fit(X_train['essay'].values)

X_train_essay_tfidf = vectorizer_tfidf.transform(X_train['essay'].values)
X_test_essay_tfidf = vectorizer_tfidf.transform(X_test['essay'].values)

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

```
(73196, 8) (73196,)
(36052, 8) (36052,)
============================================================================
====================
After vectorizations
```

```
(73196, 5000) (73196,)
(36052, 5000) (36052,)
==========================================================================================
====================
```

0.4 Applying TFIDF W2V on Essay Feature

[6]:
```
#please use below code to load glove vectors
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

[7]:
```
def get_tfidfw2v(tfidf_model, preprocessed_essays):
    dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.
 ↪idf_)))
    tfidf_words = set(tfidf_model.get_feature_names())
    tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in␣
 ↪this list
    for sentence in tqdm(preprocessed_essays): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        tf_idf_weight =0; # num of words with a valid vector in the sentence/
 ↪review
        for word in sentence.split(): # for each word in a review/sentence
            if (word in glove_words) and (word in tfidf_words):
                vec = model[word] # getting the vector for each word
                # here we are multiplying idf value(dictionary[word]) and the␣
 ↪tf value((sentence.count(word)/len(sentence.split())))
                tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.
 ↪split())) # getting the tfidf value for each word
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors.append(vector)
    print(len(tfidf_w2v_vectors))
    print(len(tfidf_w2v_vectors[0]))
    return np.array(tfidf_w2v_vectors)
```

[8]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vectorizer_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,4),␣
 ↪max_features=5000)
vectorizer_tfidf.fit(X_train['essay'].values)
```

```
X_train_essay_tfidf_w2v = get_tfidfw2v(vectorizer_tfidf, X_train['essay'].
 ↪values)
X_test_essay_tfidf_w2v = get_tfidfw2v(vectorizer_tfidf, X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf_w2v.shape, y_train.shape)
print(X_test_essay_tfidf_w2v.shape, y_test.shape)
print("="*100)
```

```
(73196, 8) (73196,)
(36052, 8) (36052,)
====================================================================================
====================
73196
300
36052
300
After vectorizations
(73196, 300) (73196,)
(36052, 300) (36052,)
====================================================================================
====================
```

0.5 Normalizing the Price Feature

[ ]:
```
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))

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

0.6 Normalizing the Previous Projects Features

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

X_train_prev_projects_norm = normalizer.
 ↪transform(X_train['teacher_number_of_previously_posted_projects'].values.
 ↪reshape(-1,1))
```

```
X_test_prev_projects_norm = normalizer.
 ↪transform(X_test['teacher_number_of_previously_posted_projects'].values.
 ↪reshape(-1,1))


print("After vectorizations")
print(X_train_prev_projects_norm.shape, y_train.shape)
print(X_test_prev_projects_norm.shape, y_test.shape)
print("="*100)
```

0.7 Calculating the Sentiment Scores of Pre Processed Essays

[ ]:
```
def get_sentiment_scores(essays):
    sid = SentimentIntensityAnalyzer()

    scores = np.zeros(shape=(len(essays),4))

    for i in tqdm(range(len(essays))):
        essay = essays.iloc[i]
        ss = sid.polarity_scores(essay)
        sentscores = [ss['neg'], ss['neu'], ss['pos'], ss['compound']]
        scores[i] = sentscores

    print(scores.shape)

    return scores
```

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

X_train_sent_scores = get_sentiment_scores(X_train['essay'])
# X_cv_sent_scores = get_sentiment_scores(X_cv['essay'])
X_test_sent_scores = get_sentiment_scores(X_test['essay'])

print("Sentiment Scores Shapes")
print(X_train_sent_scores.shape)
print(X_test_sent_scores.shape)
print("="*100)
```

0.8 List of Categorical Features

0.8.1 school_state

0.8.2 project_grade_category

0.8.3 teacher_prefix

0.8.4 clean_categories

0.8.5 clean_subcategories

[13]:
```
def get_response_code(X, y, col):

    unique_vals = np.unique(X[col])

    true_dict = dict.fromkeys(unique_vals, 0)
    false_dict = dict.fromkeys(unique_vals, 0)

    for i in range(X.shape[0]):
        if y[i] == 0:
            false_dict[X.iloc[i][col]] += 1
        if y[i] == 1:
            true_dict[X.iloc[i][col]] += 1

    for val in unique_vals:
        total = true_dict[val] + false_dict[val]
        true_dict[val] /= total
        false_dict[val] /= total

    response_code = np.zeros((X.shape[0], 2))

    for i in range(X.shape[0]):
        response_code[i][0] = false_dict[X.iloc[i][col]]
        response_code[i][1] = true_dict[X.iloc[i][col]]

    return true_dict, false_dict, response_code
```

[14]:
```
def get_response_code_test(X, y, col, true_dict, false_dict):

    response_code = np.zeros((X.shape[0], 2))

    unique_vals = list(true_dict.keys())

    for i in range(X.shape[0]):
        if([X.iloc[i][col]] in unique_vals):
            response_code[i][0] = false_dict[X.iloc[i][col]]
            response_code[i][0] = true_dict[X.iloc[i][col]]
        else:
            response_code[i][0] = 0.5
```

```
            response_code[i][1] = 0.5

    return response_code
```

0.9 Response Coding School State Feature

[15]:
```
true_dict, false_dict, X_train_school_state = get_response_code(X_train,␣
 ↪y_train, 'school_state')
X_test_school_state = get_response_code_test(X_test, y_test, 'school_state',␣
 ↪true_dict, false_dict)

print("School State Response Encoded")
print("="*50)
print(X_train_school_state.shape)
print(X_test_school_state.shape)
```

```
School State Response Encoded
==================================================
(73196, 2)
(36052, 2)
```

0.10 Project Grade Cateogory Encoded

[16]:
```
true_dict, false_dict, X_train_project_grade_category =␣
 ↪get_response_code(X_train, y_train, 'project_grade_category')
X_test_project_grade_category = get_response_code_test(X_test, y_test,␣
 ↪'project_grade_category', true_dict, false_dict)

print("Project Grade Category Encoded")
print("="*50)
print(X_train_project_grade_category.shape)
print(X_test_project_grade_category.shape)
```

```
Project Grade Category Encoded
==================================================
(73196, 2)
(36052, 2)
```

0.11 Teacher Prefix Encoded

[17]:
```
true_dict, false_dict, X_train_teacher_prefix = get_response_code(X_train,␣
 ↪y_train, 'teacher_prefix')
X_test_teacher_prefix = get_response_code_test(X_test, y_test,␣
 ↪'teacher_prefix', true_dict, false_dict)

print("Teacher Prefix Category Encoded")
print("="*50)
```

```
print(X_train_teacher_prefix.shape)
print(X_test_teacher_prefix.shape)
```

```
Teacher Prefix Category Encoded
==================================================
(73196, 2)
(36052, 2)
```

0.12 Clean Categories Encoded

[18]:
```
true_dict, false_dict, X_train_clean_categories = get_response_code(X_train,␣
 ↪y_train, 'clean_categories')
X_test_clean_categories = get_response_code_test(X_test, y_test,␣
 ↪'clean_categories', true_dict, false_dict)

print("Categories Encoded")
print("="*50)
print(X_train_clean_categories.shape)
print(X_test_clean_categories.shape)
```

```
Categories Encoded
==================================================
(73196, 2)
(36052, 2)
```

0.13 Clean Subcategories Encoded

[19]:
```
true_dict, false_dict, X_train_clean_subcategories = get_response_code(X_train,␣
 ↪y_train, 'clean_subcategories')
X_test_clean_subcategories = get_response_code_test(X_test, y_test,␣
 ↪'clean_subcategories', true_dict, false_dict)

print("Sub Categories Encoded")
print("="*50)
print(X_train_clean_subcategories.shape)
print(X_test_clean_subcategories.shape)
```

```
Sub Categories Encoded
==================================================
(73196, 2)
(36052, 2)
```

0.14 Stacking all Values into one array

[20]:
```
print("TFIDF Shapes:", X_train_school_state.shape, ␣
 ↪X_train_project_grade_category.shape,  X_train_teacher_prefix.shape, ␣
 ↪X_train_clean_categories.shape,  X_train_clean_subcategories.shape, ␣
 ↪X_train_price_norm.shape,  X_train_prev_projects_norm.shape, ␣
 ↪X_train_essay_tfidf.shape,  X_train_sent_scores.shape,  )
print("TFIDF W2V Shapes:",X_train_school_state.shape, ␣
 ↪X_train_project_grade_category.shape,  X_train_teacher_prefix.shape, ␣
 ↪X_train_clean_categories.shape,  X_train_clean_subcategories.shape, ␣
 ↪X_train_price_norm.shape,  X_train_prev_projects_norm.shape, ␣
 ↪X_train_essay_tfidf_w2v.shape,  X_train_sent_scores.shape,  )
```

```
TFIDF Shapes: (73196, 2) (73196, 2) (73196, 2) (73196, 2) (73196, 2) (73196, 1)
(73196, 1) (73196, 5000) (73196, 4)
TFIDF W2V Shapes: (73196, 2) (73196, 2) (73196, 2) (73196, 2) (73196, 2) (73196,
1) (73196, 1) (73196, 300) (73196, 4)
```

[21]:
```
X_tr_tfidf = hstack((X_train_school_state, X_train_project_grade_category,␣
 ↪X_train_teacher_prefix, X_train_clean_categories,␣
 ↪X_train_clean_subcategories, X_train_price_norm, X_train_prev_projects_norm,␣
 ↪X_train_essay_tfidf, X_train_sent_scores)).tocsr()
X_te_tfidf = hstack((X_test_school_state, X_test_project_grade_category,␣
 ↪X_test_teacher_prefix, X_test_clean_categories, X_test_clean_subcategories,␣
 ↪X_test_price_norm, X_test_prev_projects_norm, X_test_essay_tfidf,␣
 ↪X_test_sent_scores)).tocsr()

print("Final Data matrix: TFIDF")
print(X_tr_tfidf.shape, y_train.shape)
print(X_te_tfidf.shape, y_test.shape)
```

```
Final Data matrix: TFIDF
(73196, 5016) (73196,)
(36052, 5016) (36052,)
```

[29]:
```
X_tr_tfidf_w2v = X_train_school_state
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_project_grade_category␣
 ↪), axis=1)
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_teacher_prefix ),␣
 ↪axis=1)
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_clean_categories ),␣
 ↪axis=1)
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_clean_subcategories ),␣
 ↪axis=1)
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_price_norm ), axis=1)
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_prev_projects_norm ),␣
 ↪axis=1)
```

```
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_essay_tfidf_w2v ),␣
↪axis=1)
X_tr_tfidf_w2v = np.concatenate((X_tr_tfidf_w2v, X_train_sent_scores ), axis=1)

X_te_tfidf_w2v = X_test_school_state
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_project_grade_category␣
↪), axis=1)
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_teacher_prefix ),␣
↪axis=1)
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_clean_categories ),␣
↪axis=1)
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_clean_subcategories ),␣
↪axis=1)
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_price_norm ), axis=1)
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_prev_projects_norm ),␣
↪axis=1)
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_essay_tfidf_w2v ),␣
↪axis=1)
X_te_tfidf_w2v = np.concatenate((X_te_tfidf_w2v, X_test_sent_scores ), axis=1)

print("Final Data matrix: TFIDF W2V")
print(X_tr_tfidf_w2v.shape, y_train.shape)
print(X_te_tfidf_w2v.shape, y_test.shape)
```

```
Final Data matrix: TFIDF W2V
(73196, 316) (73196,)
(36052, 316) (36052,)
```

[30]:
```
from scipy import sparse
```

[31]:
```
X_tr_tfidf_w2v = sparse.csr_matrix(X_tr_tfidf_w2v)
X_te_tfidf_w2v = sparse.csr_matrix(X_te_tfidf_w2v)
```

[34]:
```
# np.save('X_tr_tfidf.npy', X_tr_tfidf)
# np.save('X_te_tfidf.npy', X_te_tfidf)

# np.save('X_tr_tfidf_w2v.npy', X_tr_tfidf_w2v)
# np.save('X_te_tfidf_w2v.npy', X_te_tfidf_w2v)
```

[ ]:
```
# X_tr_tfidf = np.load("X_tr_tfidf.npy")
# X_te_tfidf = np.load("X_te_tfidf.npy")
# X_tr_tfidf_w2v = np.load("X_tr_tfidf_w2v.npy")
# X_te_tfidf_w2v = np.load("X_te_tfidf_w2v.npy")

# a = np.load("X_tr_tfidf.npy")
# b = np.load("X_te_tfidf.npy")
# c = np.load("X_tr_tfidf_w2v.npy")
```

```
# d = np.load("X_te_tfidf_w2v.npy")
```

0.15 Using Cross Validation to find the best n_estimators and max_depth on TFIDF Data

[111]:
```
# Reference: https://mlfromscratch.com/gridsearch-keras-sklearn/#/

model = lgb.LGBMClassifier()
param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [3, 10, 15, 30],
    'is_unbalance' : [True]
}
```

[112]:
```
gs = GridSearchCV(estimator=model, param_grid = param_grid, cv = 5, n_jobs = 1,␣
 ↪scoring = "roc_auc", verbose = 2, return_train_score = True)
fitted_model = gs.fit(X_tr_tfidf, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   4.8s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   5.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   5.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   5.1s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   4.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=  11.0s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=  11.8s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=  12.8s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=  12.5s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=  12.0s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=  17.4s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=  16.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=  17.4s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=  16.7s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=  16.9s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=  29.7s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=  30.5s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=  30.3s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=  30.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=  31.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   8.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   7.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
```

```
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   8.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   7.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   8.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=  20.4s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=  21.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=  19.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=  19.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=  20.4s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=  31.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=  31.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=  32.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=  33.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=  35.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  53.5s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  52.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  53.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
```

```
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  53.5s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  56.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   8.4s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   8.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   8.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   8.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   8.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=  22.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=  22.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=  22.9s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=  22.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=  22.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=  33.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=  34.4s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=  29.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=  37.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
```

```
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=  36.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time= 1.0min
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=  59.9s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=  52.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=  50.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=  48.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   7.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   6.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   7.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   6.4s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   9.5s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=  23.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=  21.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=  25.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=  29.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=  24.9s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
```

```
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=  33.4s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=  39.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=  37.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=  42.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=  42.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time= 1.1min
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time= 1.1min
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=  51.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=  48.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=  49.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
```

[137]:
```
fitted_model.best_params_, fitted_model.best_score_
```

[137]:
```
({'is_unbalance': True, 'max_depth': 30, 'n_estimators': 100},
 0.7207814875858596)
```

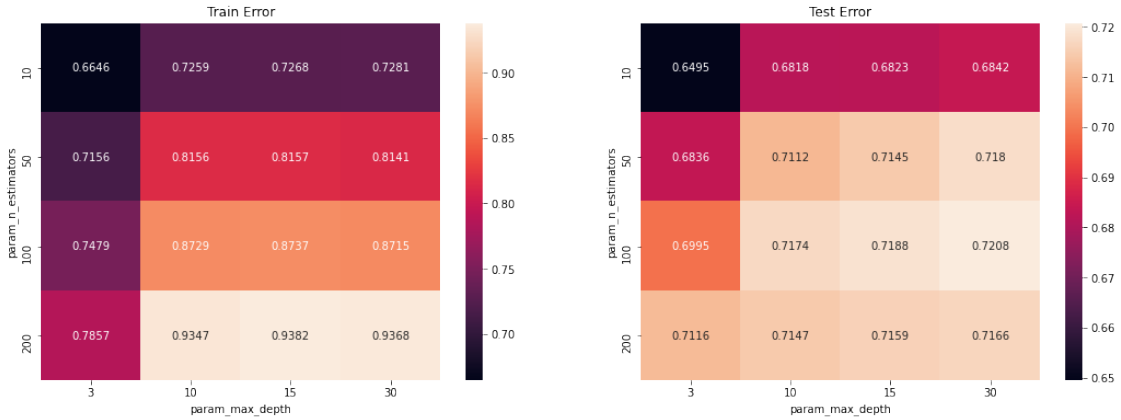
[125]:
```
estimators = []

for i in range(4):
    estimators.append(10)
    estimators.append(50)
    estimators.append(100)
    estimators.append(200)

depths = []

base_ = [3, 10, 15, 30]
```

```
for i in range(4):
    for j in range(4):
        depths.append(base_[i])
```

[127]:
```
trace1 = go.Scatter3d(x=estimators, y=depths, z=list(gs.
 ↪cv_results_['mean_train_score']), name = 'Train Error')
trace2 = go.Scatter3d(x=estimators, y=depths, z=list(gs.
 ↪cv_results_['mean_test_score']), name = 'Test Error')
data = [trace1, trace2]

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

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

[128]:
```
scores = pd.DataFrame(gs.cv_results_).
 ↪groupby(['param_n_estimators','param_max_depth']).max().
 ↪unstack()[['mean_train_score','mean_test_score']]
```

[129]:
```
fig, ax = plt.subplots(1,2, figsize=(18,6))

sns.heatmap(scores.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(scores.mean_test_score, annot = True, fmt='.4g', ax=ax[1])

ax[0].set_title('Train Error')
ax[1].set_title('Test Error')
plt.show()
```



0.16 Using the best params on TFIDF Data

[148]:
```
gbdt_tfidf = lgb.LGBMClassifier(max_depth= 30, n_estimators= 100)
gbdt_tfidf.fit(X_tr_tfidf, y_train)
```

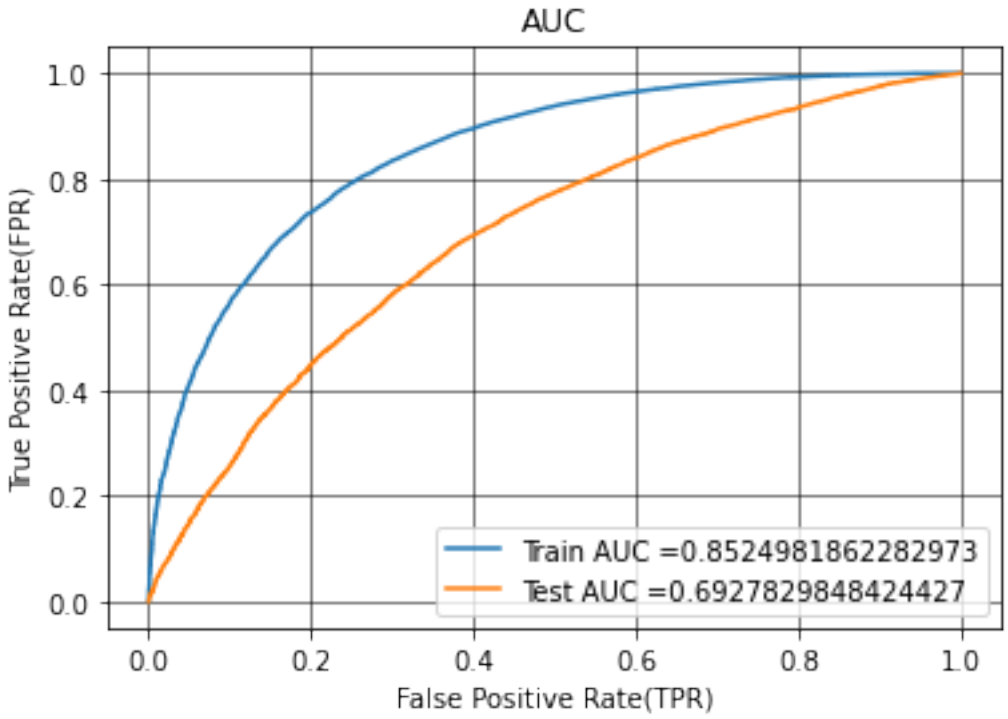
```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
```

[148]:
```
LGBMClassifier(max_depth=30)
```

[149]:
```
y_train_preds = gbdt_tfidf.predict_proba(X_tr_tfidf)
y_test_preds = gbdt_tfidf.predict_proba(X_te_tfidf)
```

[150]:
```
train_fpr_tfidf, train_tpr_tfidf, tr_thresholds_tfidf = roc_curve(y_train,␣
 ↪y_train_preds[:,1])
test_fpr_tfidf, test_tpr_tfidf, te_thresholds_tfidf = roc_curve(y_test,␣
 ↪y_test_preds[:,1])
```

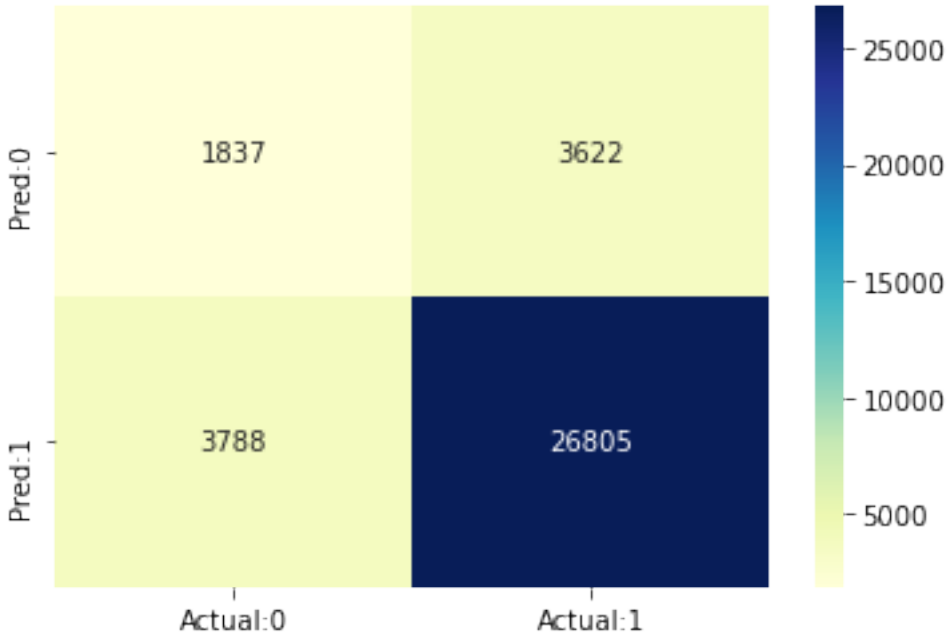
[151]:
```
plt.plot(train_fpr_tfidf, train_tpr_tfidf, label="Train AUC␣
 ↪="+str(auc(train_fpr_tfidf, train_tpr_tfidf)))
plt.plot(test_fpr_tfidf, test_tpr_tfidf, label="Test AUC␣
 ↪="+str(auc(test_fpr_tfidf, test_tpr_tfidf)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



[152]:
```
y_preds = gbdt_tfidf.predict(X_te_tfidf)
conf_mat = confusion_matrix(y_test, y_preds)
```

[153]:
```
df_cm = pd.DataFrame(conf_mat, index=["Pred:0", "Pred:1"], columns = ["Actual:
↪0", "Actual:1"])
plt.figure(figsize = (6,4))
sns.heatmap(df_cm, annot=True, fmt='g', cmap="YlGnBu")
```

[153]: <AxesSubplot:>



0.17 Using Cross Validation to find the best n_estimators and max_depth on TFIDF W2V Data

[147]:
```
model = lgb.LGBMClassifier()
param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [3, 10, 15, 30],
    'is_unbalance' : [True]
}
```

[97]:
```
gs = GridSearchCV(estimator=model, param_grid = param_grid, cv = 5, n_jobs = 1,␣
 ↪scoring = "roc_auc", verbose = 2, return_train_score = True)
fitted_model = gs.fit(X_tr_tfidf_w2v, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   1.8s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   1.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   1.7s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   1.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=10; total time=   1.7s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=   2.3s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=   3.2s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=   3.4s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=   3.3s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=50; total time=   2.6s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=   3.5s
```

```
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=   3.1s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=   3.2s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=   3.3s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=100; total time=   3.7s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=   4.8s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=   4.7s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=   5.0s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=   4.8s
[CV] END …is_unbalance=True, max_depth=3, n_estimators=200; total time=   4.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   2.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   2.5s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   2.5s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   2.5s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=10; total time=   3.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=   5.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=   5.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=   6.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=   4.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=10, n_estimators=50; total time=   4.9s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=   6.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=   6.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=   7.4s
```

```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=   9.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=100; total time=   6.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  11.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=   9.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  12.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  13.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=10, n_estimators=200; total time=  10.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   2.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   2.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   2.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   2.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=10; total time=   2.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=   5.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=   4.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=   4.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=   4.3s
```

```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=15, n_estimators=50; total time=   3.9s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=   5.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=   5.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=   8.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=   5.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=100; total time=   6.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=  14.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=  10.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=   9.9s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=  10.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=15, n_estimators=200; total time=   8.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   1.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   2.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   2.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   2.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=10; total time=   1.8s
```

```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=   3.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=   4.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=   4.6s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=   4.1s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END …is_unbalance=True, max_depth=30, n_estimators=50; total time=   3.4s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=   4.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=   5.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=   5.7s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=   5.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=100; total time=   5.0s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=   7.8s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=   7.9s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=   7.3s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=   8.2s
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set
num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[CV] END ..is_unbalance=True, max_depth=30, n_estimators=200; total time=   7.2s
```

[98]:
```
fitted_model.best_params_, fitted_model.best_score_
```

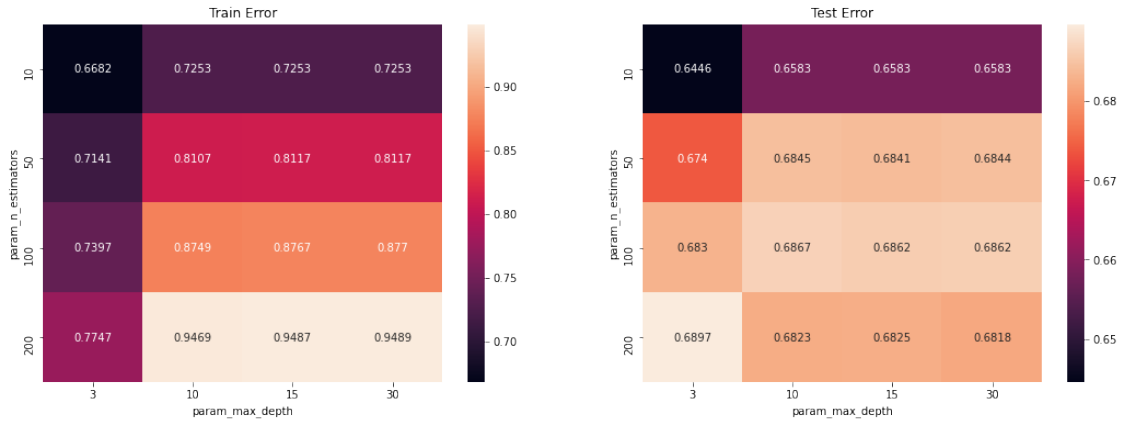
[98]:
```
({'is_unbalance': True, 'max_depth': 3, 'n_estimators': 200},
 0.689701367303171)
```

[99]:
```
estimators = []

for i in range(4):
    estimators.append(10)
    estimators.append(50)
    estimators.append(100)
    estimators.append(200)

depths = []

base_ = [3, 10, 15, 30]
for i in range(4):
    for j in range(4):
        depths.append(base_[i])
```

[100]:
```
trace1 = go.Scatter3d(x=estimators, y=depths, z=list(gs.
 ↪cv_results_['mean_train_score']), name = 'Train Error')
trace2 = go.Scatter3d(x=estimators, y=depths, z=list(gs.
 ↪cv_results_['mean_test_score']), name = 'Test Error')
data = [trace1, trace2]

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

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

[101]:
```
scores = pd.DataFrame(gs.cv_results_).
 ↪groupby(['param_n_estimators','param_max_depth']).max().
 ↪unstack()[['mean_train_score','mean_test_score']]
```

[102]:
```
fig, ax = plt.subplots(1,2, figsize=(18,6))

sns.heatmap(scores.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(scores.mean_test_score, annot = True, fmt='.4g', ax=ax[1])

ax[0].set_title('Train Error')
ax[1].set_title('Test Error')
plt.show()
```



0.18 Using best params on TFIDF W2V Data

[104]:
```
gs.best_params_
```

[104]:
```
{'is_unbalance': True, 'max_depth': 3, 'n_estimators': 200}
```

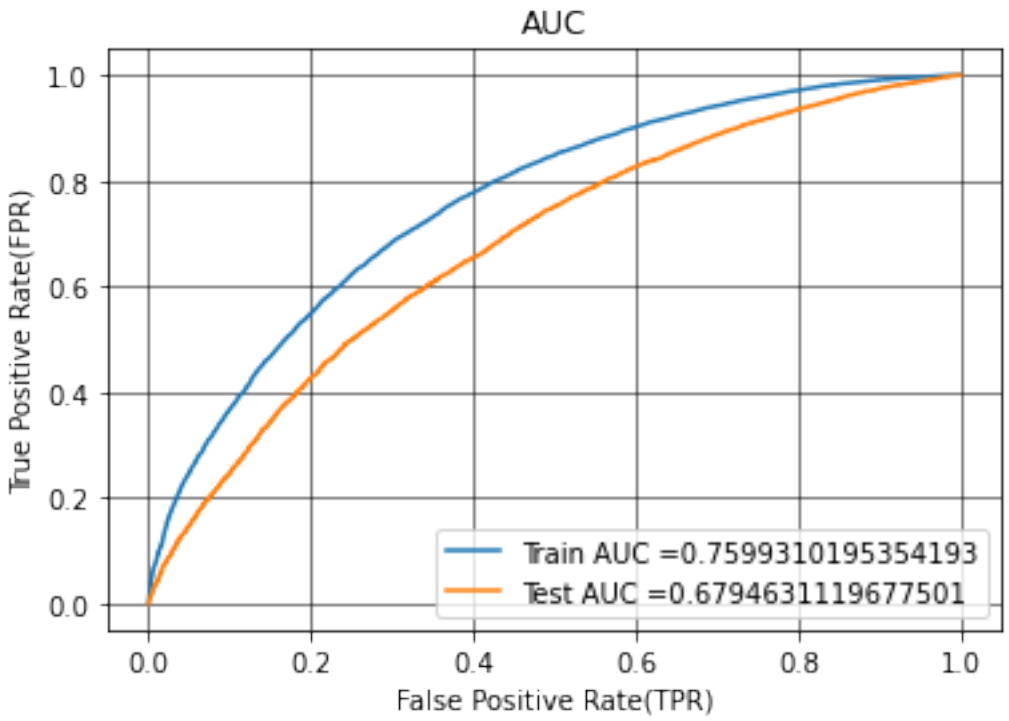
[160]:
```
gbdt_tfidf_w2v = lgb.LGBMClassifier(max_depth= 3, n_estimators= 200)
gbdt_tfidf_w2v.fit(X_tr_tfidf_w2v, y_train)
```

[160]:
```
LGBMClassifier(max_depth=3, n_estimators=200)
```

[161]:
```
y_train_preds = gbdt_tfidf_w2v.predict_proba(X_tr_tfidf_w2v)
y_test_preds = gbdt_tfidf_w2v.predict_proba(X_te_tfidf_w2v)
```

[162]:
```
train_fpr_tfidf_w2v, train_tpr_tfidf_w2v, tr_thresholds_tfidf_w2v =␣
↪roc_curve(y_train, y_train_preds[:,1])
test_fpr_tfidf_w2v, test_tpr_tfidf_w2v, te_thresholds_tfidf_w2v =␣
↪roc_curve(y_test, y_test_preds[:,1])
```

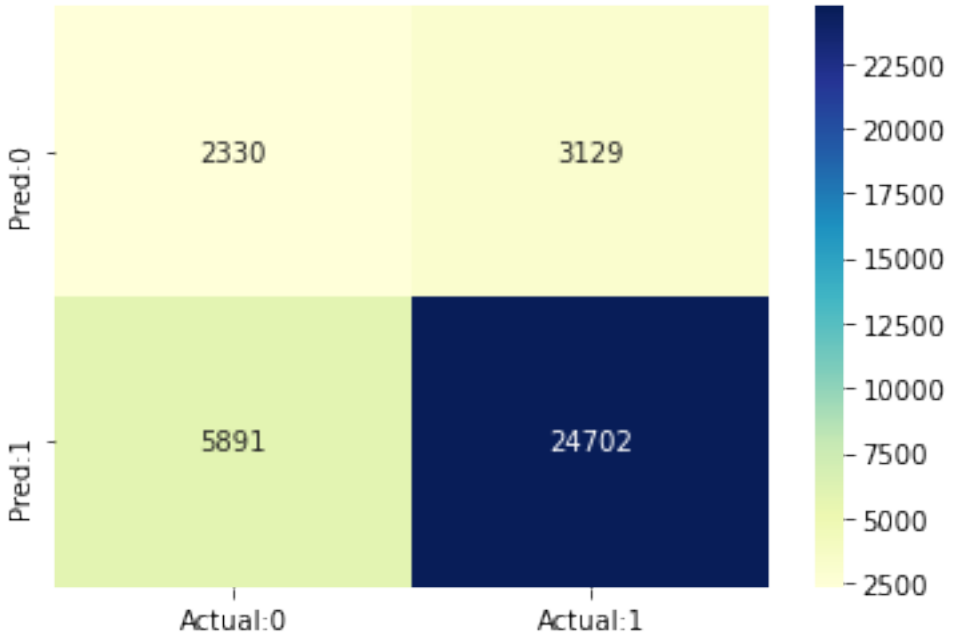
[163]:
```
plt.plot(train_fpr_tfidf_w2v, train_tpr_tfidf_w2v, label="Train AUC␣
↪="+str(auc(train_fpr_tfidf_w2v, train_tpr_tfidf_w2v)))
plt.plot(test_fpr_tfidf_w2v, test_tpr_tfidf_w2v, label="Test AUC␣
↪="+str(auc(test_fpr_tfidf_w2v, test_tpr_tfidf_w2v)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



[164]:
```
y_preds = gbdt_tfidf_w2v.predict(X_te_tfidf_w2v)
conf_mat = confusion_matrix(y_test, y_preds)
```

[165]:
```
df_cm = pd.DataFrame(conf_mat, index=["Pred:0", "Pred:1"], columns = ["Actual:
↪0", "Actual:1"])
plt.figure(figsize = (6,4))
sns.heatmap(df_cm, annot=True, fmt='g', cmap="YlGnBu")
```

[165]: <AxesSubplot:>



0.19 Summary

0.19.1 Using is_unbalance = True on best parameters is resulting in lots of False Positives on both TFIDF and TFIDF W2V Models

0.19.2 The difference in Train AUC and Test AUC is more in TFIDF compared to TFIDF W2V Model

[136]:
```
from prettytable import PrettyTable

x = PrettyTable()
x = PrettyTable(["Vectorizer", "Model", "n_estimators", "max_depth", "Train␣
 ↪AUC", "Test AUC"])
x.add_row(["TFIDF", "GBDT", 100, 30, 0.85249, 0.69278])
x.add_row(["TFIDF W2V", "GBDT", 200, 3 , 0.75993, 0.67946])

print(x)
```

```
+------------+-------+--------------+-----------+-----------+----------+
| Vectorizer | Model | n_estimators | max_depth | Train AUC | Test AUC |
+------------+-------+--------------+-----------+-----------+----------+
|   TFIDF    |  GBDT |     100      |     30    |  0.85249  | 0.69278  |
| TFIDF W2V  |  GBDT |     200      |     3     |  0.75993  | 0.67946  |
+------------+-------+--------------+-----------+-----------+----------+
```