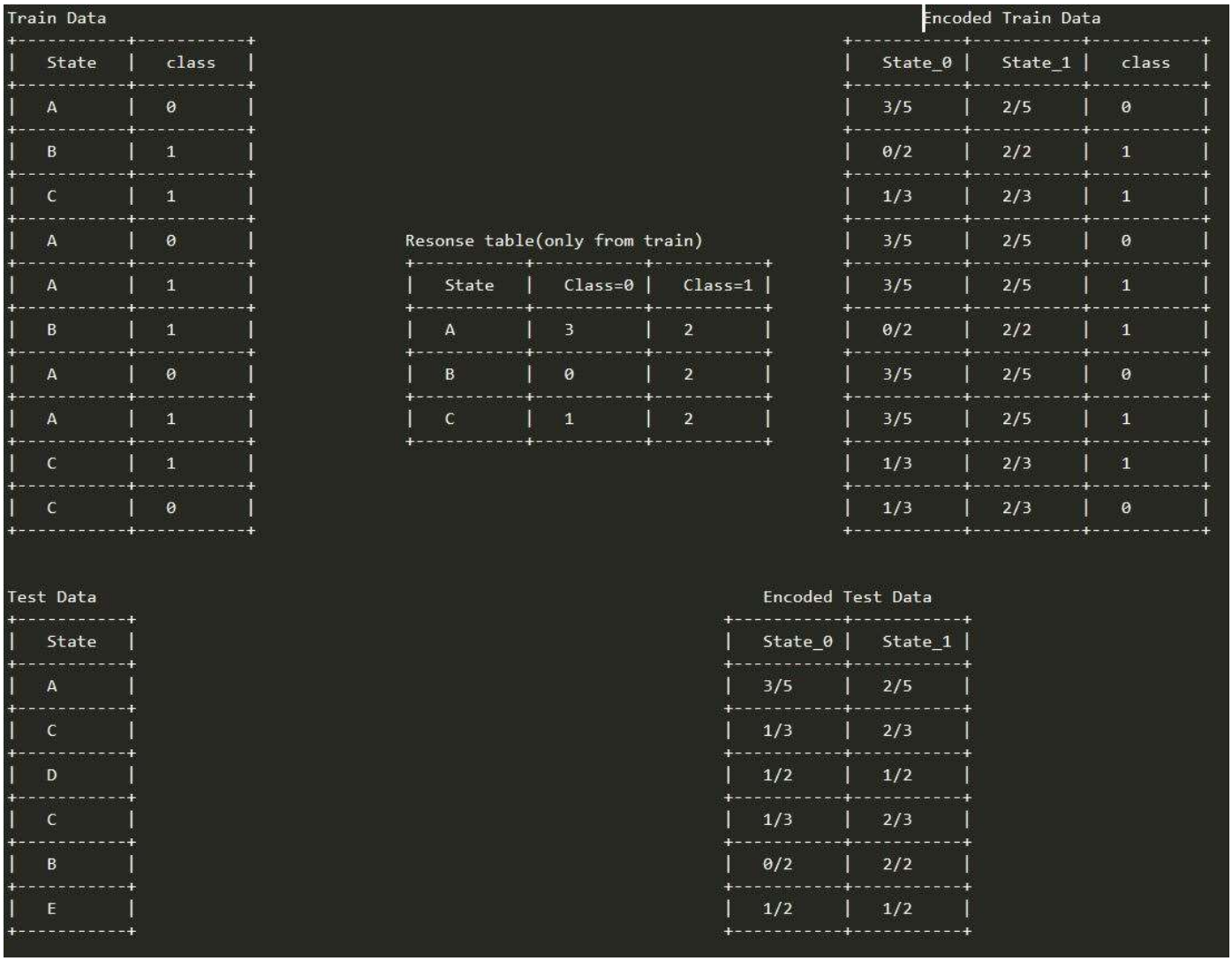
Assignment 9: GBDT

Response Coding: Example



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

In [1]:

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

In [2]:

```
import seaborn as sns
```

In [3]:

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

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\PRASAD\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[3]:

True

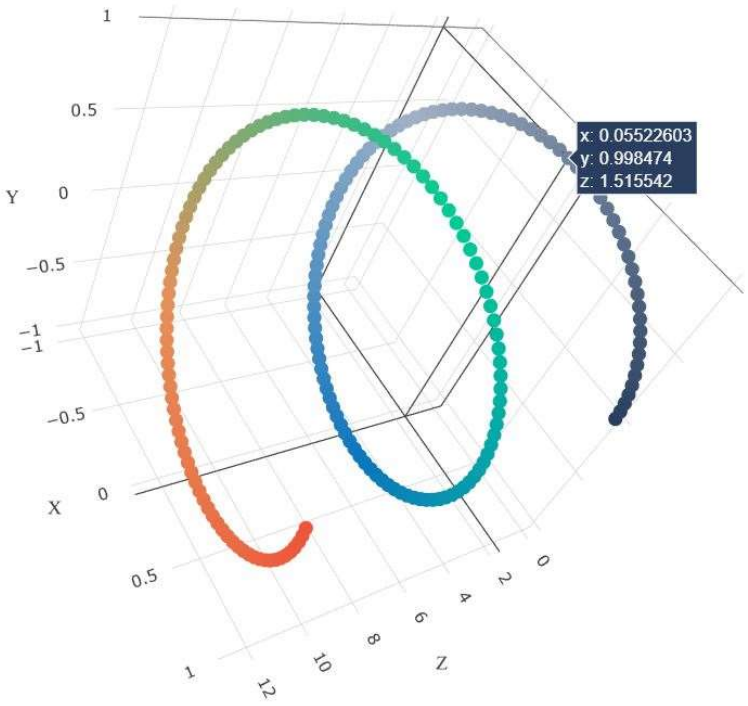
1. **Apply GBDT on these feature sets**

   - 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/): 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/): use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

   - Here in response encoding you need to apply the **laplase smoothing** value for test set. Laplase smoothing means, If test point is present in test but not in train then you need to apply default 0.5 as probability value for that data point (Refer the Response Encoding Image from above cell)
   - Please use atleast **35k** data points
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/) 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**

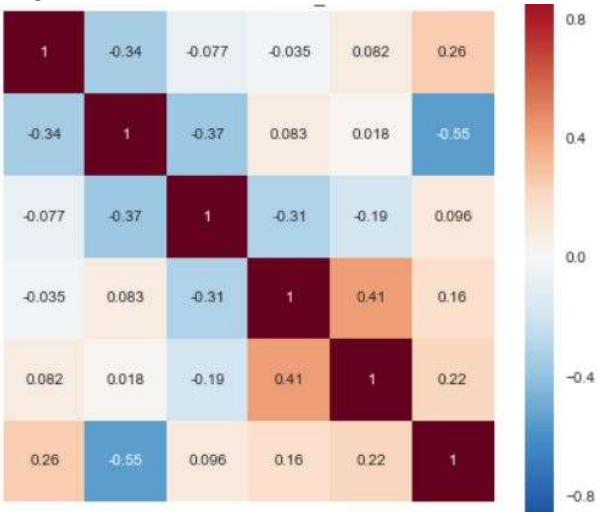
   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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

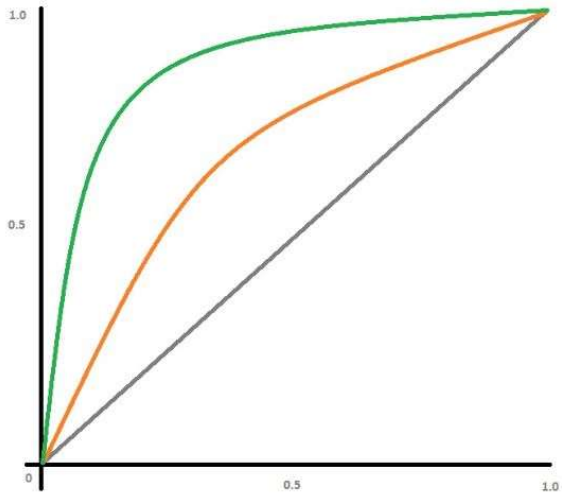
or

- 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. Make sure that you are using predict_proba method to calculate AUC curves, because AUC is calcualted on class probabilities and not on class labels.



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

Few Notes

1. Use atleast 35k data points
2. Use classifier.Predict_proba() method instead of predict() method while calculating roc_auc scores
3. Be sure that you are using laplase smoothing in response encoding function. Laplase smoothing means applying the default (0.5) value to test data if the test data is not present in the train set

1. GBDT (xgboost/lightgbm)

1.1 Loading Data

In [14]:

```
import pandas
data = pandas.read_csv('preprocessed_data.csv', nrows=35000)
```

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

In [0]:

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

In [15]:

```
from sklearn.model_selection import train_test_split
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[15]:

```
((23450, 8), (11550, 8), (23450,), (11550,))
```

In [16]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import numpy as np

def sentiment_score(X,feature):
  neg=[]
  neu=[]
  pos=[]
  compound=[]
  sid = SentimentIntensityAnalyzer()
  for i in range(len(X)):
    for_sentiment = X[feature].iloc[i]
    ss = sid.polarity_scores(for_sentiment)
    neg.append(ss['neg'])
    neu.append(ss['neu'])
    pos.append(ss['pos'])
    compound.append(ss['compound'])
  return np.asarray(neg).reshape(-1,1),np.asarray(neu).reshape(-1,1),np.asarray(pos).reshap

negative,neutral,postive,compound = sentiment_score(X_train,"essay")
X_train["sen_neg"]=negative
X_train["sen_pos"]=postive
X_train["sen_neu"]=neutral
X_train["sen_comp"]=compound

negative,neutral,postive,compound = sentiment_score(X_test,"essay")
X_test["sen_neg"]=negative
X_test["sen_pos"]=postive
X_test["sen_neu"]=neutral
X_test["sen_comp"]=compound
```

```
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:21: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)

C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:22: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)

C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:23: Settin
gWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)

C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:24: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)

C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:27: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)

C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:28: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)

C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:29: Settin
gWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)

C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:30: Settin
gWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
```

1.3 Make Data Model Ready: encoding eassay, and project_title

In [0]:

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

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

In [17]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

preprocessed_essays = data['essay'].values

vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['essay'].values)
Xtrn_tfidf = vectorizer.transform(X_train['essay'].values)
Xtst_tfidf = vectorizer.transform(X_test['essay'].values)
```

In [18]:

```
import pickle
from tqdm import tqdm

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

# 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())

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay'].values): # 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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            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)
Xtrn_tfidf_w2v = np.array(tfidf_w2v_vectors)

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay'].values): # 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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            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)
Xtst_tfidf_w2v = np.array(tfidf_w2v_vectors)

print(Xtrn_tfidf_w2v.shape)
print(Xtst_tfidf_w2v.shape)
```

```
100%|████████████████████████████████████████████████████████████████
████| 23450/23450 [01:21<00:00, 287.05it/s]
100%|████████████████████████████████████████████████████████████████
████| 11550/11550 [00:39<00:00, 291.30it/s]
```

```
(23450, 300)
(11550, 300)
```

1.4 Make Data Model Ready: encoding numerical, categorical features

In [0]:

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

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

In [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))

normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)
X_train_numpreproj_norm = normalizer.transform(X_train['teacher_number_of_previously_posted
X_test_numpreproj_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_p
```

In [20]:

```
# The below code is refered from the following link: https://stackoverflow.com/questions/66

def response_coding(xtrain, ytrain, feature):
    dictionary = dict()
    unique_cat_labels = xtrain[feature].unique()
    for i in range(len(unique_cat_labels)):
        total_count = xtrain.loc[:,feature][(xtrain[feature] == unique_cat_labels[i])].cour
        p_0 = xtrain.loc[:, feature][((xtrain[feature] == unique_cat_labels[i]) & (ytrain==
        p_1 = xtrain.loc[:, feature][((xtrain[feature] == unique_cat_labels[i]) & (ytrain==
        dictionary[unique_cat_labels[i]] = [p_1/total_count, p_0/total_count]
    return dictionary

def get_response_coding_array(X, Y, feature):
    resp_feature = response_coding(X, Y, feature)
    state_0 = []
    state_1 = []
    for item in X[feature]:
        if item in resp_feature:
            state_0.append(resp_feature.get(item)[1])
            state_1.append(resp_feature.get(item)[0])
        else:
            state_0.append(0.5)
            state_1.append(0.5)
    return np.concatenate((np.array(state_0).reshape(-1,1),np.array(state_1).reshape(-1,1))

Xtrn_teacher_prefix = get_response_coding_array(X_train, y_train, 'teacher_prefix')
Xtst_teacher_prefix = get_response_coding_array(X_test, y_test, 'teacher_prefix')

Xtrn_school_state = get_response_coding_array(X_train, y_train, 'school_state')
Xtst_school_state = get_response_coding_array(X_test, y_test, 'school_state')

Xtrn_pgc = get_response_coding_array(X_train, y_train, 'project_grade_category')
Xtst_pgc = get_response_coding_array(X_test, y_test, 'project_grade_category')

Xtrn_clean_categories = get_response_coding_array(X_train, y_train, 'clean_categories')
Xtst_clean_categories = get_response_coding_array(X_test, y_test, 'clean_categories')

Xtrn_clean_subcategories = get_response_coding_array(X_train, y_train, 'clean_subcategories
Xtst_clean_subcategories = get_response_coding_array(X_test, y_test, 'clean_subcategories')

Xtrn_teacher_prefix.shape, Xtrn_school_state.shape, Xtrn_pgc.shape, Xtrn_clean_categories.s
```

Out[20]:

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

In [21]:

```
from scipy.sparse import hstack

X_tr_set1 = hstack((Xtrn_tfidf, Xtrn_school_state, Xtrn_teacher_prefix, Xtrn_pgc, Xtrn_clea
X_tst_set1 = hstack((Xtst_tfidf, Xtst_school_state, Xtst_teacher_prefix, Xtst_pgc, Xtst_cle
X_tr_set1.shape, X_tst_set1.shape
```

Out[21]:

```
((23450, 8932), (11550, 8932))
```

In [22]:

```
import scipy as scipy
X_tr_set2 = hstack((scipy.sparse.csr_matrix(Xtrn_tfidf_w2v), Xtrn_school_state, Xtrn_teache
X_tst_set2 = hstack((scipy.sparse.csr_matrix(Xtst_tfidf_w2v), Xtst_school_state, Xtst_teach
X_tr_set2.shape, X_tst_set2.shape
```

Out[22]:

```
((23450, 316), (11550, 316))
```

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

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

In [0]:

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

In [23]:

```
import xgboost as xgb
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import pandas as pd
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()

parameters = {'n_estimators':[5, 10, 100, 500], 'learning_rate':[0.1,1.0,2.0,3.0,5.0]}
param_dist = {'objective':'binary:logistic', 'n_estimators':2}
XGBC = xgb.XGBClassifier(**param_dist)
clf = GridSearchCV(XGBC, parameters, cv=3, scoring='roc_auc', return_train_score=True)
clf.fit(X_tr_set1, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results_n_estimators = results.sort_values(['param_n_estimators'])
results_learning_rate = results.sort_values(['param_learning_rate'])

train_auc= results['mean_train_score']
cv_auc = results['mean_test_score']
n_estimators =  np.array(results['param_n_estimators'])
learning_rate =  np.array(results['param_learning_rate'])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=learning_rate,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=learning_rate,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

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

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

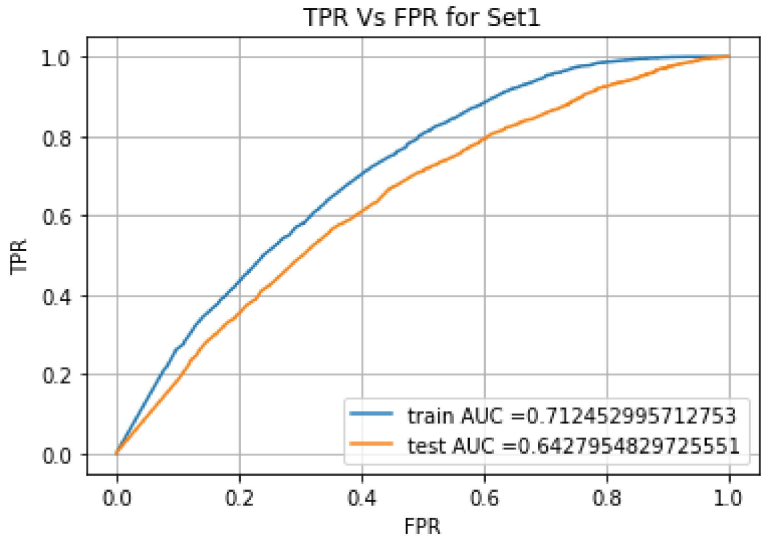
```
[07:00:50] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.

C:\Users\PRASAD\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserW
arning:

The use of label encoder in XGBClassifier is deprecated and will be remove
d in a future release. To remove this warning, do the following: 1) Pass o
ption use_label_encoder=False when constructing XGBClassifier object; and
2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ...,
[num_class - 1].
```

```
[07:00:53] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
```

In [24]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your tr_loop will be 49041 - 49041%1000 = 4900
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred

from sklearn.metrics import roc_curve, auc
#Best Hyperparameters
#n_estimators = 5
#learning_rate = 0.1
param_dist = {'objective':'binary:logistic', 'n_estimators':5, 'learning_rate':0.1}
clf_tfidf = xgb.XGBClassifier(**param_dist)
clf_tfidf.fit(X_tr_set1, y_train)
y_trn_predict = batch_predict(clf_tfidf, X_tr_set1)
y_tst_predict = batch_predict(clf_tfidf, X_tst_set1)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_trn_predict)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_tst_predict)
set1_tr_AUC = auc(train_fpr, train_tpr)
set1_tst_AUC = auc(test_fpr, test_tpr)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(set1_tr_AUC))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(set1_tst_AUC))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("TPR Vs FPR for Set1")
plt.grid()
plt.show()
```

```
C:\Users\PRASAD\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWar
ning:

The use of label encoder in XGBClassifier is deprecated and will be removed
in a future release. To remove this warning, do the following: 1) Pass optio
n use_label_encoder=False when constructing XGBClassifier object; and 2) Enc
ode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_cla
ss - 1].


[07:05:27] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
```


TPR Vs FPR for Set1
TPR
FPR
train AUC =0.712452995712753
test AUC =0.6427954829725551
0.0
0.2
0.4
0.6
0.8
1.0

In [25]:

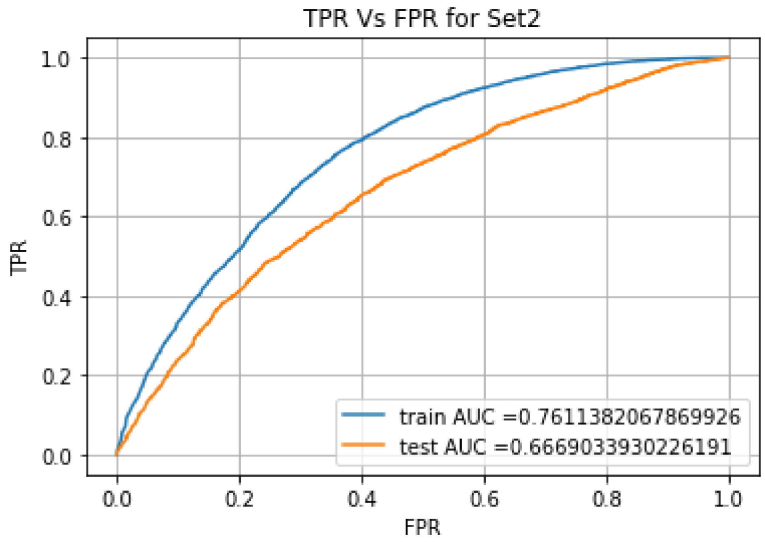
```
import warnings
warnings.filterwarnings("ignore")

clf.fit(X_tr_set2, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results_n_estimators = results.sort_values(['param_n_estimators'])
results_learning_rate = results.sort_values(['param_learning_rate'])

train_auc= results['mean_train_score']
cv_auc = results['mean_test_score']
n_estimators =  np.array(results['param_n_estimators'])
learning_rate =  np.array(results['param_learning_rate'])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=learning_rate,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=learning_rate,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

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

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

```
old behavior.
[07:35:14] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[07:36:13] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[07:36:16] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
[07:36:18] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
```

In [27]:

```
#Best Hyperparameters
#n_estimators = 5
#learning_rate = 0.1
param_dist = {'objective':'binary:logistic', 'n_estimators':5, 'learning_rate':0.1}
clf_tfidfW2v = xgb.XGBClassifier(**param_dist)
clf_tfidfW2v.fit(X_tr_set2, y_train)
y_trn_predict = batch_predict(clf_tfidfW2v, X_tr_set2)
y_tst_predict = batch_predict(clf_tfidfW2v, X_tst_set2)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_trn_predict)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_tst_predict)
set2_tr_AUC = auc(train_fpr, train_tpr)
set2_tst_AUC = auc(test_fpr, test_tpr)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(set2_tr_AUC))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(set2_tst_AUC))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("TPR Vs FPR for Set2")
plt.grid()
plt.show()
```

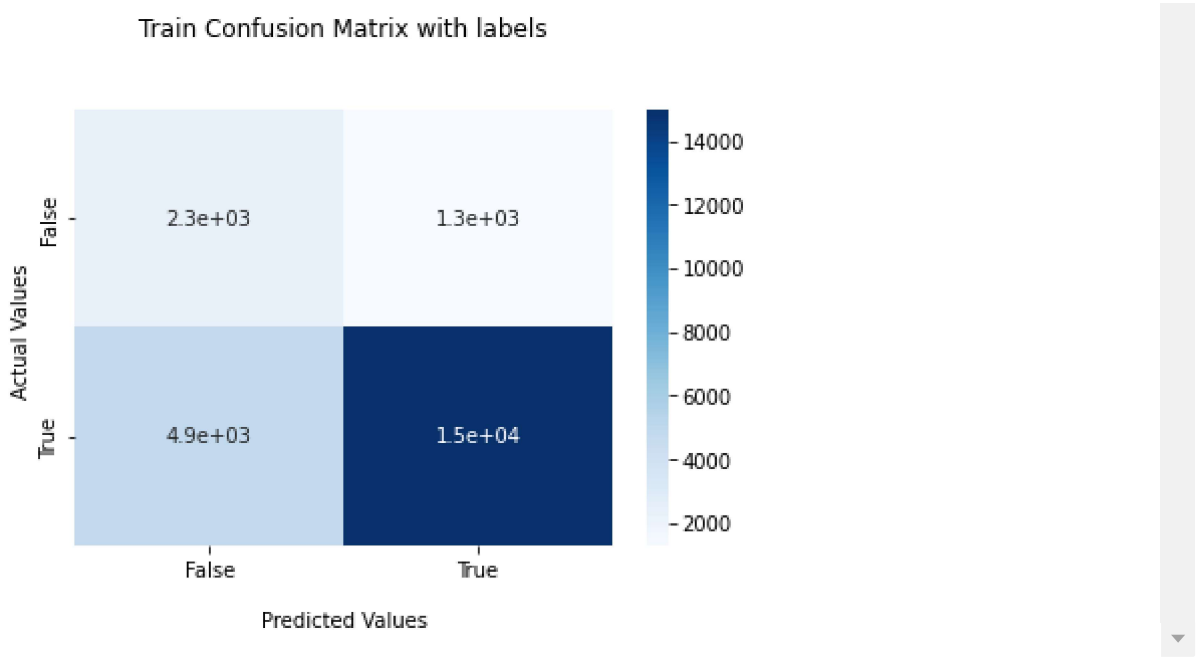
```
[07:42:28] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
```

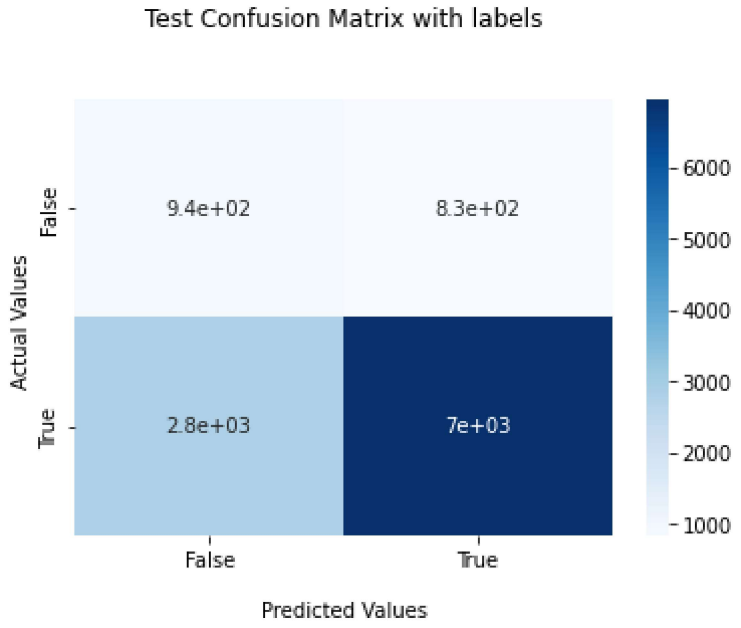


In [28]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[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
    return t

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

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)

# The below code is refered from the following link: https://www.stackvidhya.com/plot-confu

import seaborn as sns

ax = sns.heatmap(confusion_matrix(y_train, predict_with_best_t(y_trn_predict, best_t)), anr

ax.set_title('Train Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()

ax = sns.heatmap(confusion_matrix(y_test, predict_with_best_t(y_tst_predict, best_t)), annc

ax.set_title('Test Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.48540018525623563 for threshold 0.634
```





3. Summary

as mentioned in the step 4 of instructions

In [29]:

```
from tabulate import tabulate
data = np.array(["Set1",set1_tr_AUC,set1_tst_AUC,"Set2",set2_tr_AUC,set2_tst_AUC]).reshape(
print(tabulate(data, headers=["Dataset", "Train_AUC", "Test_AUC"], tablefmt="grid", showinc
```

```
+----+-----------+-------------+------------+
|    | Dataset   |   Train_AUC |   Test_AUC |
+====+===========+=============+============+
|  0 | Set1      |    0.712453 |   0.642795 |
+----+-----------+-------------+------------+
|  1 | Set2      |    0.761138 |   0.666903 |
+----+-----------+-------------+------------+
```