Assignment : DT

Please check below video before attempting this assignment

TF-IDFW2V

Tfidf w2v (w1,w2..) = (tfidf(w1) *w2v(w1)* + *tfidf(w2)* w2v(w2) + …) / (tfidf(w1) + tfidf(w2) + …)

(Optional) Please check course video on AVgw2V and TF-IDFW2V (https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/2916/avg-word2vec-tf-idf-weighted-word2vec/3/module-3-foundations-of-natural-language-processing-and-machine-learning)for more details.

Glove vectors

In this assignment you will be working with glove vectors , please check this (https://en.wikipedia.org/wiki/GloVe_(machine_learning)) and this (https://en.wikipedia.org/wiki/GloVe_(machine_learning)) for more details.

Download glove vectors from this link (https://drive.google.com/file/d/1lDca_ge-GYO0iQ6_XDLWePQFMdAA2b8f/view?usp=sharing)

In [45]:

```
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())
```

or else , you can use below code

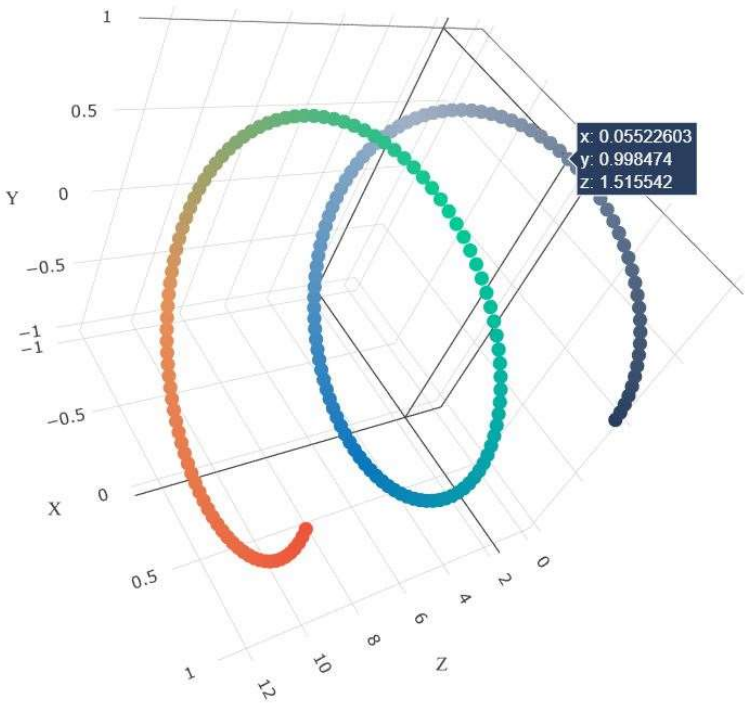
Task - 1

1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
    - Set 1: categorical, numerical features + preprocessed_essay (TFIDF) + Sentiment scores(preprocessed_essay)
    - Set 2: categorical, numerical features + preprocessed_essay (TFIDF W2V) + Sentiment scores(preprocessed_essay)
2. **The hyper paramter tuning (best `depth` in range [1, 3, 10, 30], and the best `min_samples_split` in range [5, 10, 100, 500])**
    - 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(use gridsearch cv or randomsearch cv)/simple cross validation data(you can write your own for loops refer sample solution)

3. **Representation of results**

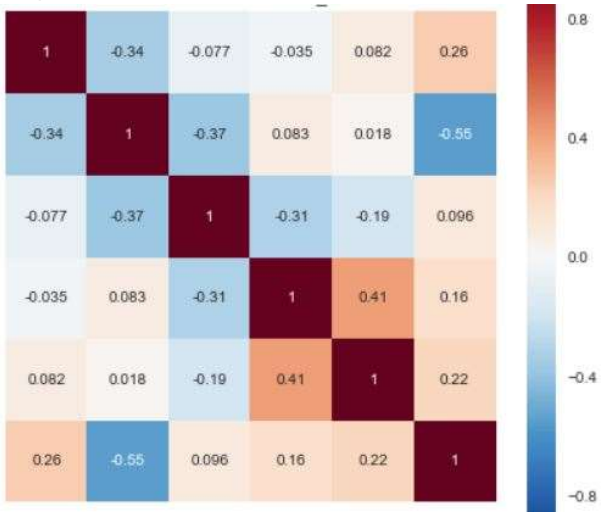
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



with X-axis as **min_sample_split**, 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

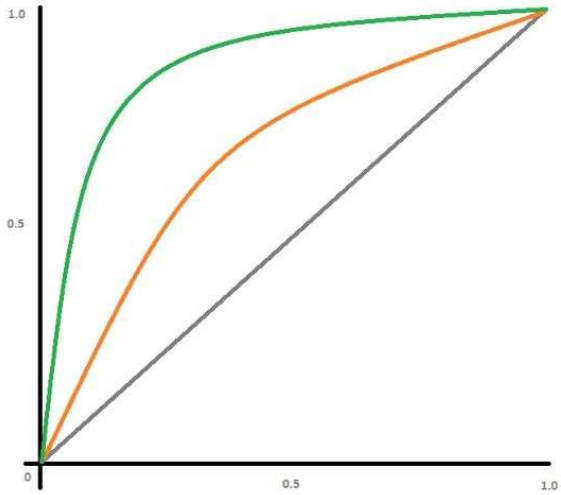
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **min_sample_split**, 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 = ??

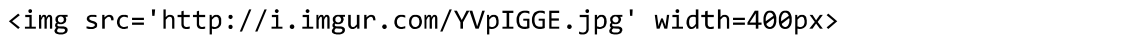
- Once after you plot the confusion matrix with the test data, get all the `false positive data points`
    - Plot the WordCloud(https://www.geeksforgeeks.org/generating-word-cloud-python/ (https://www.geeksforgeeks.org/generating-word-cloud-python/)) with the words of essay text of these `false positive data points`
    - Plot the box plot with the `price` of these `false positive data points`
    - Plot the pdf with the `teacher_number_of_previously_posted_projects` of these `false positive data points`

Task - 2

For this task consider **set-1** features.

- Select all the features which are having non-zero feature importance.You can get the feature importance using 'feature*importances*` (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)), discard the all other remaining features and then apply any of the model of you choice i.e. (Dession tree, Logistic Regression, Linear SVM).
- You need to do hyperparameter tuning corresponding to the model you selected and procedure in step 2 and step 3
  Note: when you want to find the feature importance make sure you don't use max_depth parameter keep it None.

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



Hint for calculating Sentiment scores

In [46]:

```
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[46]:

```
True
```

Decision Tree

Task - 1

1.1 Loading Data

In [2]:

```
#make sure you are loading atleast 50k datapoints
#you can work with features of preprocessed_data.csv for the assignment.
import pandas

data = pandas.read_csv('preprocessed_data.csv', nrows=50000)
```

In [ ]:

```
# write your code in following steps for task 1
# 1. calculate sentiment scores for the essay feature
# 2. Split your data.
# 3. perform tfidf vectorization of text data.
# 4. perform tfidf w2v vectorization of text data.
# 5. perform encoding of categorical features.
# 6. perform encoding of numerical features
# 7. For task 1 set 1 stack up all the features
# 8. For task 1 set 2 stack up all the features (for stacking dense features you can use np
# 9. Perform hyperparameter tuning and plot either heatmap or 3d plot.
# 10. Find the best parameters and fit the model. Plot ROC-AUC curve(using predict proba me
# 11. Plot confusion matrix based on best threshold value
# 12. Find all the false positive data points and plot wordcloud of essay text and pdf of t
# 13. Write your observations about the wordcloud and pdf.
```

In [3]:

```
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[3]:

```
((33500, 8), (16500, 8), (33500,), (16500,))
```

In [5]:

```
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: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:22: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:23: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:24: Sett
ingWithCopyWarning:
```

```
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:27: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:28: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:29: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:30: Sett
ingWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.or
g/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
```

In [6]:

```
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 [7]:

```
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%|████████████████████████████████████████████████████████████████
████| 33500/33500 [02:33<00:00, 217.54it/s]
100%|████████████████████████████████████████████████████████████████
████| 16500/16500 [01:11<00:00, 231.69it/s]
```

```
(33500, 300)
(16500, 300)
```

In [8]:

```
from sklearn.feature_extraction.text import CountVectorizer

# provided we did the cleaning
vectorizer = CountVectorizer(binary=True)
# school state
vectorizer.fit(X_train['school_state'].values)
Xtrn_school_state = vectorizer.transform(X_train['school_state'].values)
Xtst_school_state = vectorizer.transform(X_test['school_state'].values)
# teacher_prefix
vectorizer.fit(X_train['teacher_prefix'].values)
Xtrn_teacher_prefix = vectorizer.transform(X_train['teacher_prefix'].values)
Xtst_teacher_prefix = vectorizer.transform(X_test['teacher_prefix'].values)
# project_grade_category
vectorizer.fit(X_train['project_grade_category'].values)
Xtrn_pgc = vectorizer.transform(X_train['project_grade_category'].values)
Xtst_pgc = vectorizer.transform(X_test['project_grade_category'].values)
# clean_categories
vectorizer.fit(X_train['clean_categories'].values)
Xtrn_clean_categories = vectorizer.transform(X_train['clean_categories'].values)
Xtst_clean_categories = vectorizer.transform(X_test['clean_categories'].values)
# clean_subcategories
vectorizer.fit(X_train['clean_subcategories'].values)
Xtrn_clean_subcategories = vectorizer.transform(X_train['clean_subcategories'].values)
Xtst_clean_subcategories = vectorizer.transform(X_test['clean_subcategories'].values)
```

In [9]:

```
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 [12]:

```
X_train["sen_neg"].shape,X_train["sen_pos"].shape, X_train["sen_neu"].shape, X_train["sen_c
```

Out[12]:

```
((33500,), (33500,), (33500,), (33500,))
```

In [13]:

```
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
```

```
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: FutureW
arning: reshape is deprecated and will raise in a subsequent release. Please
use .values.reshape(...) instead
  This is separate from the ipykernel package so we can avoid doing imports
until
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureW
arning: reshape is deprecated and will raise in a subsequent release. Please
use .values.reshape(...) instead
  after removing the cwd from sys.path.
```

In [14]:

```
X_tr_set2 = hstack((Xtrn_tfidf_w2v, Xtrn_school_state, Xtrn_teacher_prefix, Xtrn_pgc, Xtrn_
X_tst_set2 = hstack((Xtst_tfidf_w2v, Xtst_school_state, Xtst_teacher_prefix, Xtst_pgc, Xtst
```

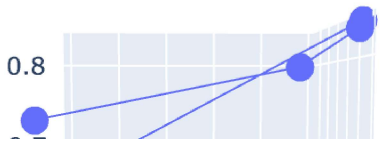
```
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureW
arning: reshape is deprecated and will raise in a subsequent release. Please
use .values.reshape(...) instead
  """Entry point for launching an IPython kernel.
C:\Users\PRASAD\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureW
arning: reshape is deprecated and will raise in a subsequent release. Please
use .values.reshape(...) instead
```

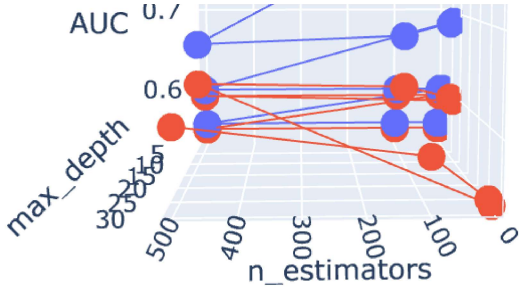
In [110]:

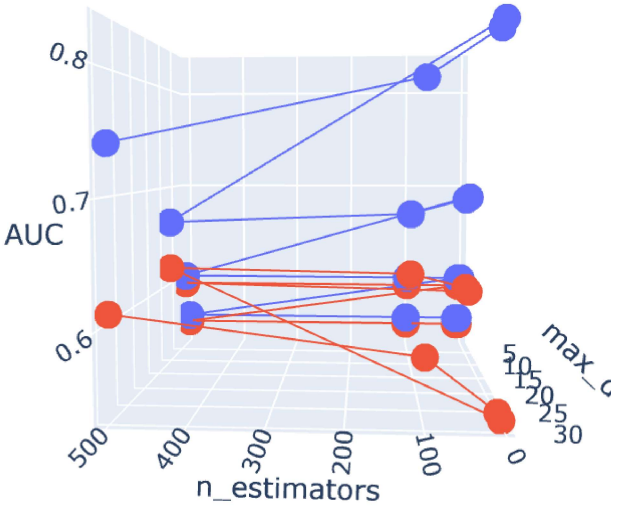
```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging 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.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 = {'max_depth':[1, 3, 10, 30], 'min_samples_split':[5, 10, 100, 500]}
DT = DecisionTreeClassifier(criterion='gini', splitter='best', min_samples_leaf=1, min_weig
clf = GridSearchCV(DT, 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_max_depth = results.sort_values(['param_max_depth'])
results_min_samples_split = results.sort_values(['param_min_samples_split'])

train_auc= results['mean_train_score']
cv_auc = results['mean_test_score']
max_depth =  np.array(results['param_max_depth'])
min_samples_split =  np.array(results['param_min_samples_split'])

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

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

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

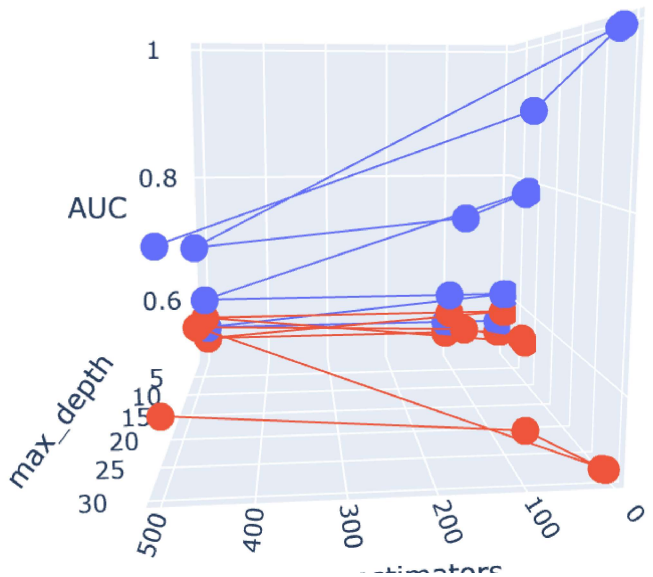


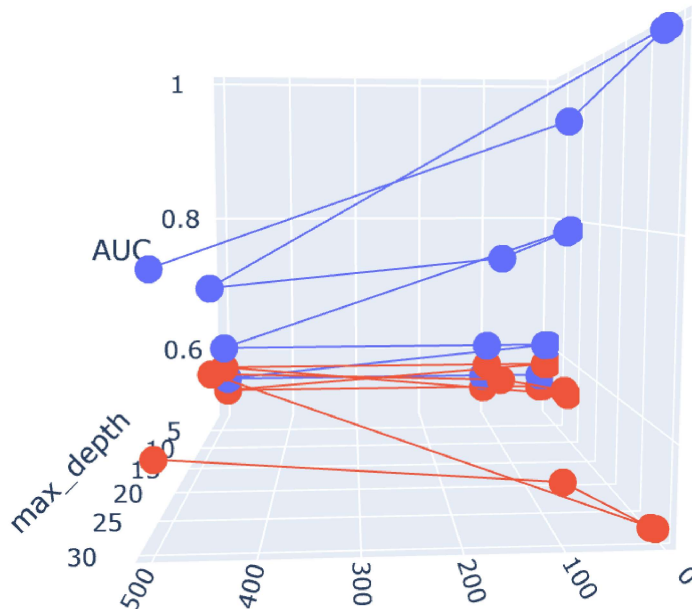

AUC
0.7
0.6
max_depth
15
30
500
400
300
200
100
0
n_estimators


0.8
0.7
AUC
0.6
5
10
15
20
25
30
500
400
300
200
100
0
n_estimators

In [16]:

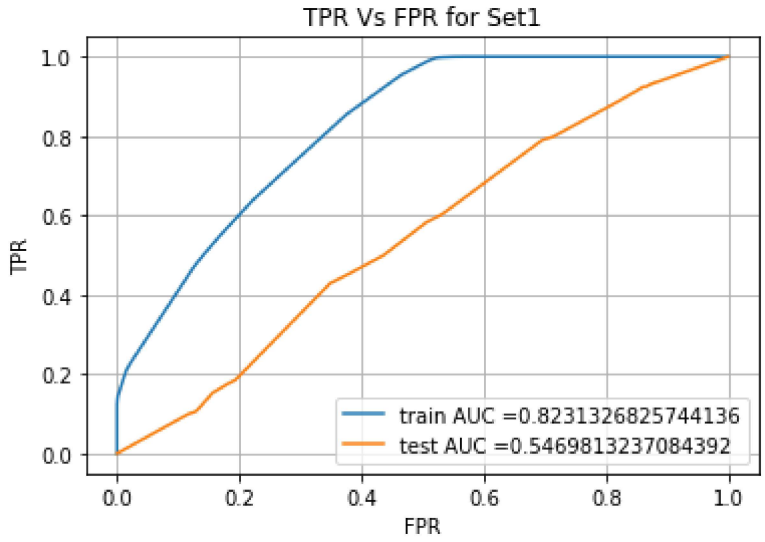
```
parameters = {'max_depth':[1, 3, 10, 30], 'min_samples_split':[5, 10, 100, 500]}
DT = DecisionTreeClassifier(criterion='gini', splitter='best', min_samples_leaf=1, min_weig
clf = GridSearchCV(DT, parameters, cv=3, scoring='roc_auc', return_train_score=True)
clf.fit(X_tr_set2, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results_max_depth = results.sort_values(['param_max_depth'])
results_min_samples_split = results.sort_values(['param_min_samples_split'])

train_auc= results['mean_train_score']
cv_auc = results['mean_test_score']
max_depth =  np.array(results['param_max_depth'])
min_samples_split =  np.array(results['param_min_samples_split'])

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

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

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

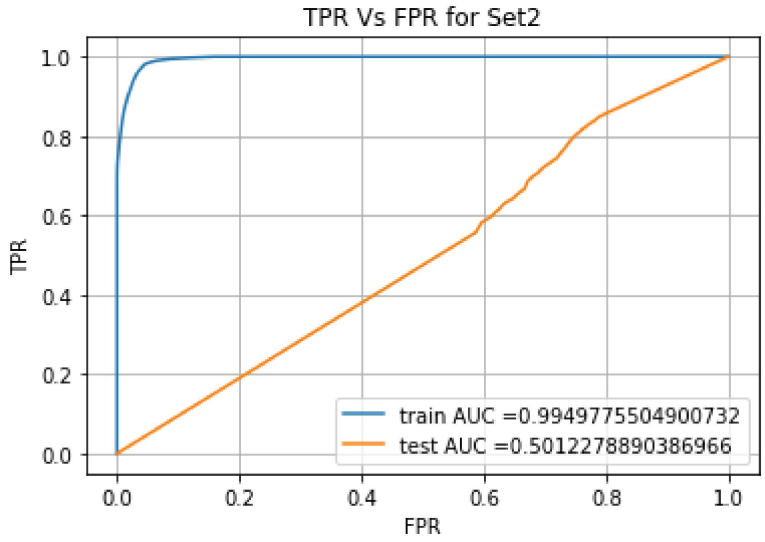



1
0.8
AUC
0.6
max_depth
20
25
30
500
400
300
200
100

In [26]:

```
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
#max_depth = 30
#min_samples_split = 5
clf_tfidf = DecisionTreeClassifier(max_depth = 30, min_samples_split = 5, criterion='gini',
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()

clf_tfidfW2v = DecisionTreeClassifier(max_depth = 30, min_samples_split = 5, criterion='gir
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()
```

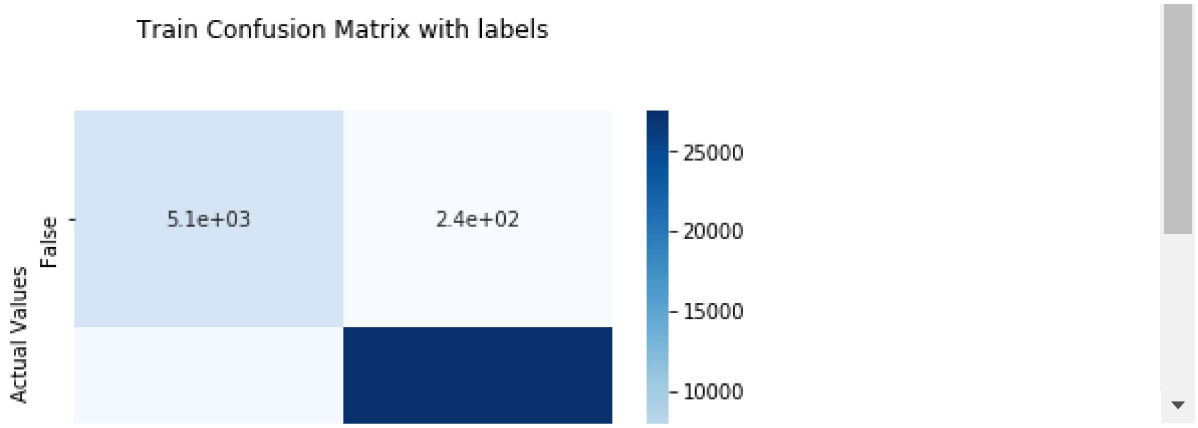

TPR Vs FPR for Set1
TPR
FPR
train AUC =0.8231326825744136
test AUC =0.5469813237084392

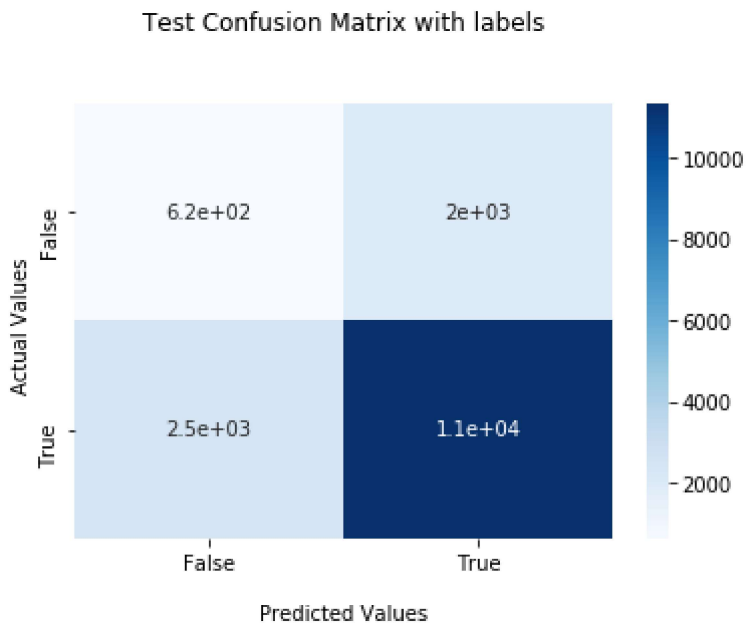

TPR Vs FPR for Set2
TPR
FPR
train AUC =0.9949775504900732
test AUC =0.5012278890386966

In [18]:

```
# 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()
```

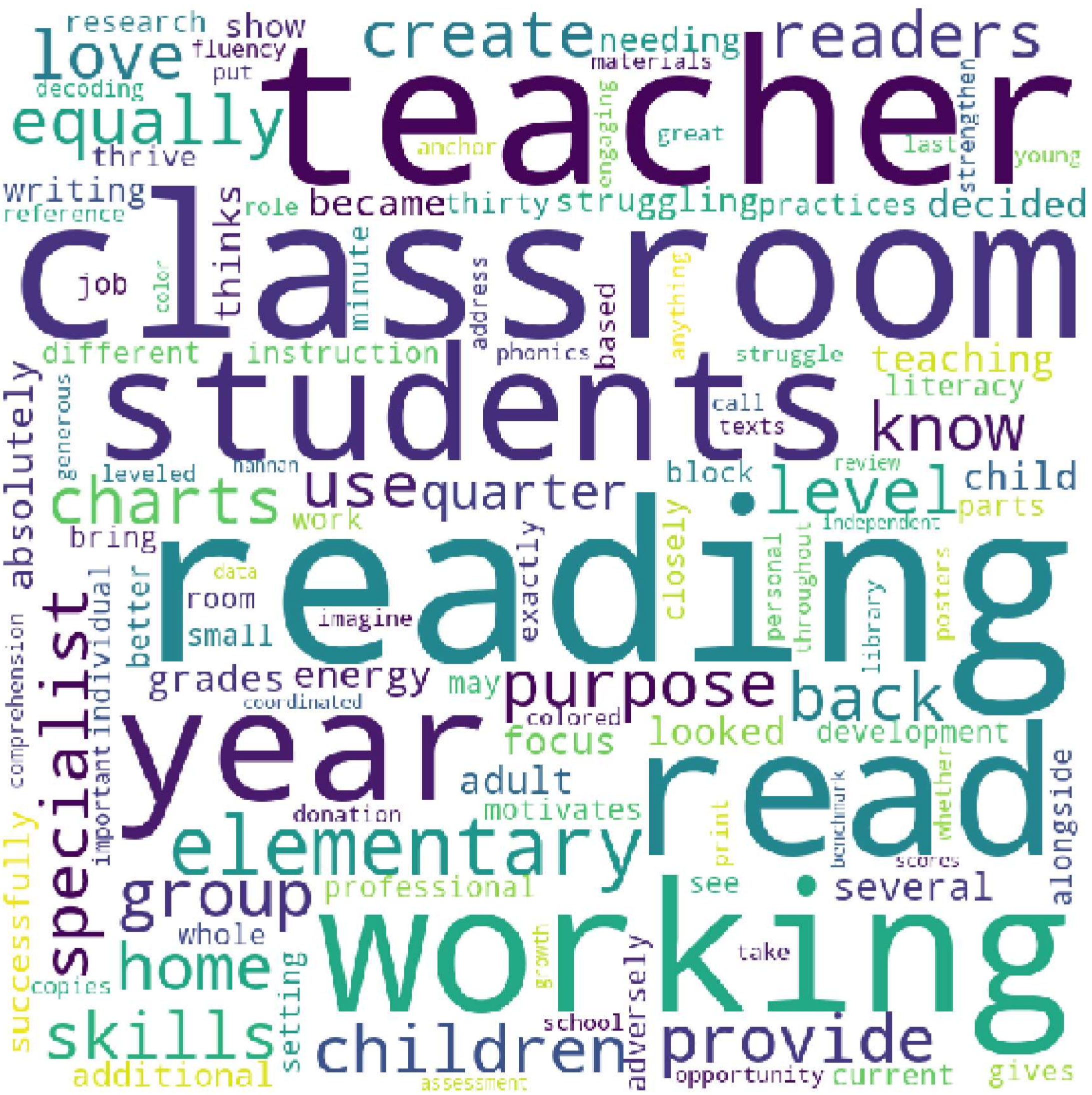
```
the maximum value of tpr*(1-fpr) 0.9353046347733294 for threshold 0.875
```


Train Confusion Matrix with labels
Actual Values
False
5.1e+03
2.4e+02
25000
20000
15000
10000


Test Confusion Matrix with labels
Actual Values
False
True
6.2e+02
2e+03
2.5e+03
1.1e+04
10000
8000
6000
4000
2000
False
True
Predicted Values

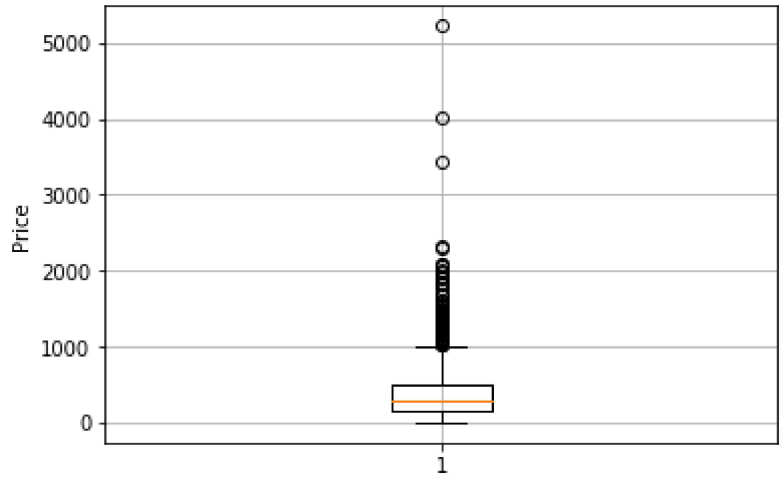
In [19]:

```
from wordcloud import WordCloud, STOPWORDS
from sklearn import preprocessing

fpi = []
for i in range(len(y_test)):
  if y_test[i]==0 and y_tst_predict[i] ==1:
    fpi.append(i)
fp_essay = []
for i in fpi:
    fp_essay.append(X_test['essay'].values[i])

#Reference from https://www.geeksforgeeks.org/generating-word-cloud-python/
# iterate through the csv file
for val in fp_essay:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()
    comment_words = ""
    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = STOPWORDS,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```


teacher
classroom
students
reading
year
read
working
elementary
specialist
purpose
children
provide
skills
home
group
charts
use
quarter
level
know
back
create
readers
love
equally
research
show
fluency
needing
materials
put
decoding
thrive
writing
reference
thinks
role
became
thirty
struggling
practices
decided
job
different
instruction
phonics
struggle
teaching
literacy
absolutely
leveled
block
child
parts
bring
work
closely
exactly
independent
room
small
imagine
grades
energy
may
looked
focus
development
adult
motivates
donation
comprehension
coordinated
colored
professional
see
several
alongside
whole
take
copies
setting
school
adversely
additional
assessment
opportunity
current
gives
successfully
important
individual
better
minute
address
based
anything
great
anchor
last
young
strengthen
engaging
color
generous
texts
call
review
personal
throughout
library
posters
data
print
benchmark
scores
whether
growth

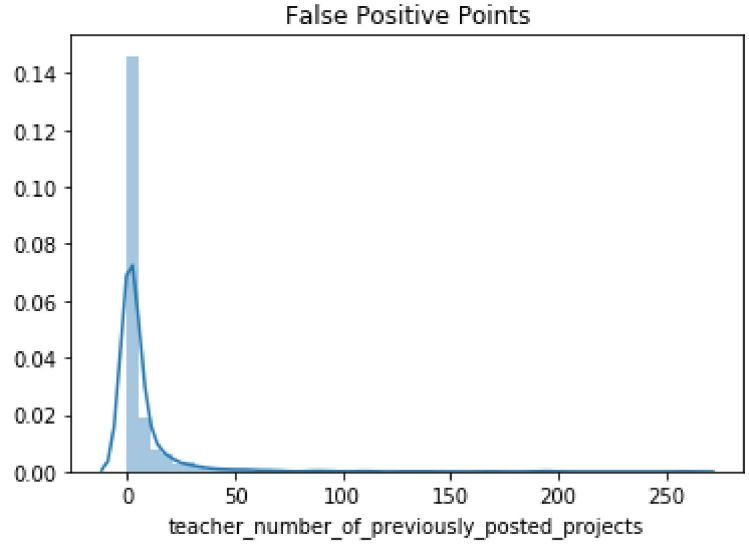
In [20]:

```
fp_price = []
for i in fpi:
    fp_price.append(X_test['price'].values[i])

# https://glowingpython.blogspot.com/2012/09/boxplot-with-matplotlib.html
plt.boxplot(fp_price)
plt.ylabel('Price')
plt.grid()
plt.show()
```



In [21]:

```
import seaborn as sns

fp_teacher_number_of_previously_posted_projects = []
for i in fpi:
    fp_teacher_number_of_previously_posted_projects.append(X_test['teacher_number_of_previc

sns.distplot(fp_teacher_number_of_previously_posted_projects)
plt.title('False Positive Points')
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



In [ ]:

```
Observations:
    1. Tfidf word2vec AUC score is better than TFIDF AUC score
```

Task - 2

In [ ]:

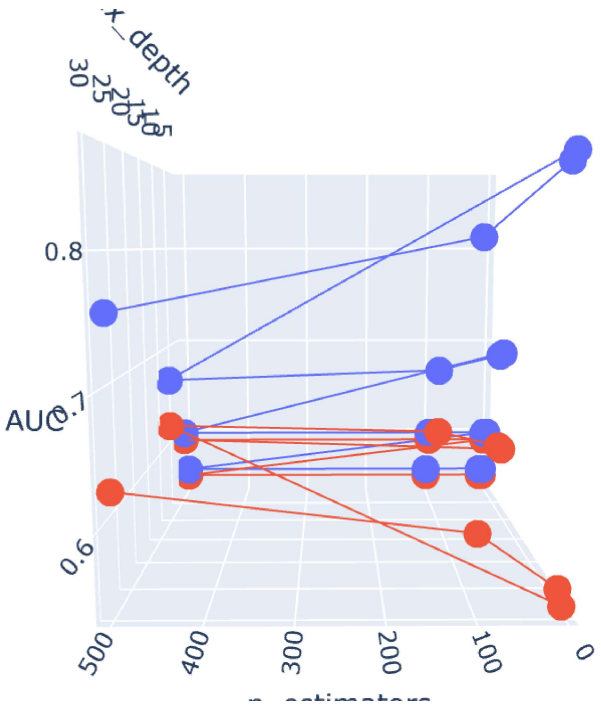
```
# 1. write your code in following steps for task 2
# 2. select all non zero features
# 3. Update your dataset i.e. X_train,X_test and X_cv so that it contains all rows and only
# 4. perform hyperparameter tuning and plot either heatmap or 3d plot.
# 5. Fit the best model. Plot ROC AUC curve and confusion matrix similar to model 1.
```

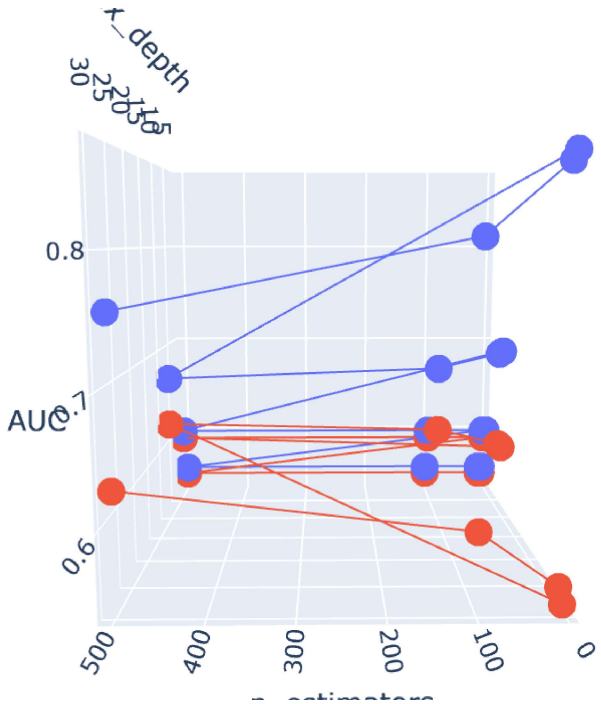
In [24]:

```
feature_id_lst =[]
for i in range(len(clf_tfidf.feature_importances_)):
    if clf_tfidf.feature_importances_[i]!=0:
        feature_id_lst.append(i)

to_select = np.ix_(range(X_tr_set1.shape[0]),feature_id_lst)
X_tr = X_tr_set1[to_select]

to_select = np.ix_(range(X_tst_set1.shape[0]),feature_id_lst)
X_tst = X_tst_set1[to_select]
```

In [23]:

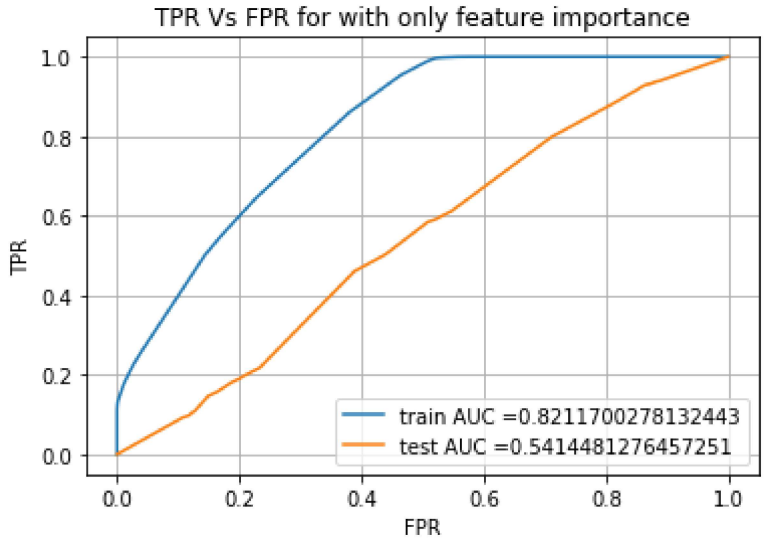
```
parameters = {'max_depth':[1, 3, 10, 30], 'min_samples_split':[5, 10, 100, 500]}
DT = DecisionTreeClassifier(criterion='gini', splitter='best', min_samples_leaf=1, min_weig
clf = GridSearchCV(DT, parameters, cv=3, scoring='roc_auc', return_train_score=True)
clf.fit(X_tr, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results_max_depth = results.sort_values(['param_max_depth'])
results_min_samples_split = results.sort_values(['param_min_samples_split'])

train_auc= results['mean_train_score']
cv_auc = results['mean_test_score']
max_depth =  np.array(results['param_max_depth'])
min_samples_split =  np.array(results['param_min_samples_split'])

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

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

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




depth
30
25
20
15
10
0.8
AUC
0.7
0.6
500
400
300
200
100
0

In [28]:

```
clf = DecisionTreeClassifier(max_depth = 30, min_samples_split = 5, criterion='gini', split
clf.fit(X_tr, y_train)
y_trn_predict = batch_predict(clf, X_tr)
y_tst_predict = batch_predict(clf, X_tst)
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_FI = auc(train_fpr, train_tpr)
set1_tst_FI = auc(test_fpr, test_tpr)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(set1_tr_FI))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(set1_tst_FI))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("TPR Vs FPR for with only feature importance")
plt.grid()
plt.show()
```



In [ ]:

```
# Tabulate your results
```

In [30]:

```
from tabulate import tabulate
data = np.array(["Set1",set1_tr_AUC,set1_tst_AUC,"Set2",set2_tr_AUC,set2_tst_AUC,"Set1_with
print(tabulate(data, headers=["Dataset", "Train_AUC", "Test_AUC"], tablefmt="grid", showinc
```

```
+----+-----------------------------+-------------+------------+
|    | Dataset                     |   Train_AUC |   Test_AUC |
+====+=============================+=============+============+
|  0 | Set1                        |    0.823133 |   0.546981 |
+----+-----------------------------+-------------+------------+
|  1 | Set2                        |    0.994978 |   0.501228 |
+----+-----------------------------+-------------+------------+
|  2 | Set1_with_featureImportance |    0.82117  |   0.541448 |
+----+-----------------------------+-------------+------------+
```