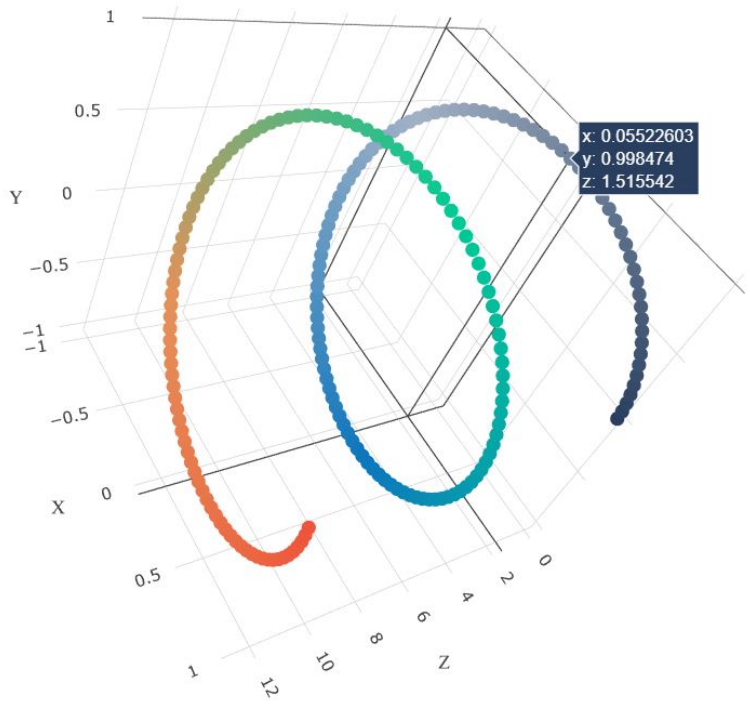
Assignment 8: DT

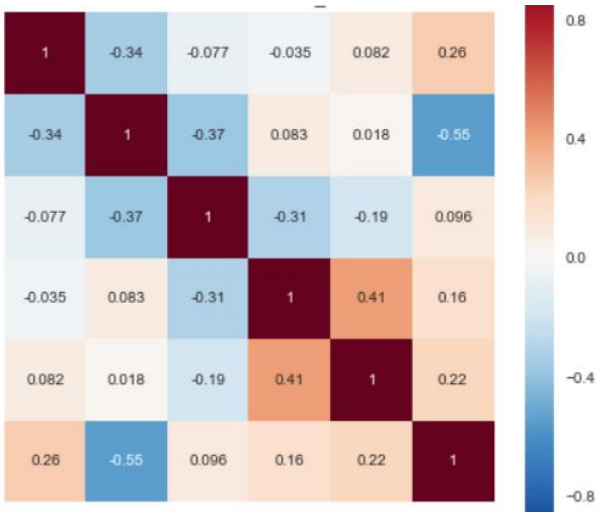
1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
    - Set 1: categorical, numerical features + preprocessed_eassay (TFIDF)
    - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF W2V)
2. **The hyper paramter tuning (best `depth` in range [1, 5, 10, 50], 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*

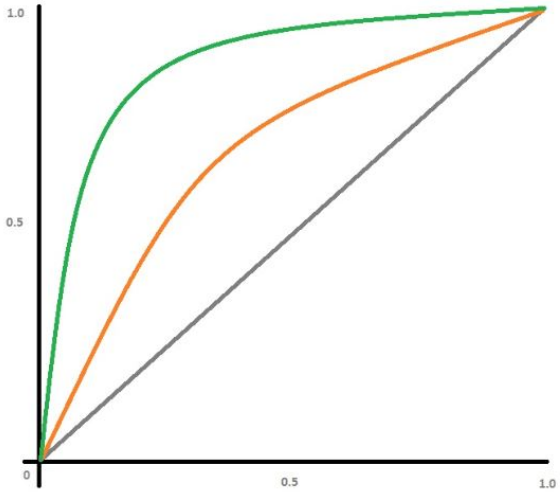
or

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

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



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

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

- 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`

4. **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.

5. 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

1. Decision Tree

1.1 Loading Data

In [1]:
```
! pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\aaa\anaconda3\lib\site-packag
es (4.6.0)
Requirement already satisfied: six in c:\users\aaa\anaconda3\lib\site-packages
(from plotly) (1.11.0)
Requirement already satisfied: retrying>=1.3.3 in c:\users\aaa\anaconda3\lib\si
te-packages (from plotly) (1.3.3)

distributed 1.21.8 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 20.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' com
mand.
```

In [2]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc

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

import pickle
from tqdm import tqdm
import os

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

In [3]:
```
import pandas as pd
data = pd.read_csv('preprocessed_data.csv',nrows=50000)
data.head(1)
```

Out[3]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projec
0	ca	mrs	grades_prek_2	

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

In [4]:
```
# please write all the code with proper documentation, and proper titles for
# 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 debu
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the re
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [5]:
```
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)
X.head(1)
```

Out[5]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projec
0	ca	mrs	grades_prek_2	

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, str
# X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size

print(X_train.shape)
print(X_test.shape)
print("**************************")
print(y_train.shape)
print(y_test.shape)
```

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

In [ ]:
```

```

In [ ]:
```

```

1.3 Make Data Model Ready: encoding eassay, and project_title

TFIDF Vectorization

In [7]:
```
# please write all the code with proper documentation, and proper titles for
# 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 debu
# 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 re
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

TFIDF Vectorization Eassy

In [8]:
```
vect= TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['essay'].values)
X_train_essay_tfidf = vect.transform(X_train['essay'].values)
# X_cv_essay_tfidf = vect.transform(X_cv['essay'].values)
X_test_essay_tfidf = vect.transform(X_test['essay'].values)
essay_tfidf_features=vect.get_feature_names()

print('X_train_essay_tfidf.shape',X_train_essay_tfidf.shape, y_train.shape)
print('X_test_essay_tfidf.shape',X_test_essay_tfidf.shape, y_test.shape)
# print('X_cv_essay_tfidf.shape',X_cv_essay_tfidf.shape, y_cv.shape)s
print(essay_tfidf_features[:10])
```

```
X_train_essay_tfidf.shape (33500, 50000) (33500,)
X_test_essay_tfidf.shape (16500, 50000) (16500,)
['00', '000', '000 steps', '000 students', '10', '10 000', '10 11', '10 12', '1
0 15', '10 chromebooks']
```

1.4 Make Data Model Ready: encoding numerical, categorical features

In [9]:
```
# please write all the code with proper documentation, and proper titles for
# 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 debu
# 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 re
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

school_state-OHE

In [10]:
```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['school_state'].values)
X_train_ohe_school_state= vect.transform(X_train['school_state'].values)
# X_cv_ohe_school_state= vect.transform(X_cv['school_state'].values)
X_test_ohe_school_state= vect.transform(X_test['school_state'].values)
school_state_features=vect.get_feature_names()

print('X_train_ohe_school_state',X_train_ohe_school_state.shape, y_train.shap
print('X_test_ohe_school_state',X_test_ohe_school_state.shape, y_test.shape)
# print('X_cv_ohe_school_state',X_cv_ohe_school_state.shape, y_cv.shape)
print(school_state_features[:10])
```

```
X_train_ohe_school_state (33500, 51) (33500,)
X_test_ohe_school_state (16500, 51) (16500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl']
```

teacher_prefix-OHE

In [11]:
```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['teacher_prefix'].values)
X_train_ohe_teacher_prefix= vect.transform(X_train['teacher_prefix'].values)
# X_cv_ohe_teacher_prefix= vect.transform(X_cv['teacher_prefix'].values)
X_test_ohe_teacher_prefix= vect.transform(X_test['teacher_prefix'].values)
teacher_prefix_features=vect.get_feature_names()

print('X_train_ohe_teacher_prefix',X_train_ohe_teacher_prefix.shape, y_train.
print('X_test_ohe_teacher_prefix',X_test_ohe_teacher_prefix.shape, y_test.sha
# print('X_cv_ohe_teacher_prefix',X_cv_ohe_teacher_prefix.shape, y_cv.shape)
print(teacher_prefix_features[:10])
```

```
X_train_ohe_teacher_prefix (33500, 4) (33500,)
X_test_ohe_teacher_prefix (16500, 4) (16500,)
['mr', 'mrs', 'ms', 'teacher']
```

project_grade_category-OHE

In [12]:
```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['project_grade_category'].values)
X_train_ohe_project_grade_category= vect.transform(X_train['project_grade_cat
# X_cv_ohe_project_grade_category= vect.transform(X_cv['project_grade_categor
X_test_ohe_project_grade_category= vect.transform(X_test['project_grade_categ
project_grade_category_features=vect.get_feature_names()

print('X_train_ohe_project_grade_category',X_train_ohe_project_grade_category
print('X_test_ohe_project_grade_category',X_test_ohe_project_grade_category.s
# print('X_cv_ohe_project_grade_category',X_cv_ohe_project_grade_category.sha
print(project_grade_category_features[:10])

```

```
X_train_ohe_project_grade_category (33500, 4) (33500,)
X_test_ohe_project_grade_category (16500, 4) (16500,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
```

clean_categories-OHE

In [13]:
```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['clean_categories'].values)
X_train_ohe_clean_categories= vect.transform(X_train['clean_categories'].valu
# X_cv_ohe_clean_categories= vect.transform(X_cv['clean_categories'].values)
X_test_ohe_clean_categories= vect.transform(X_test['clean_categories'].values
clean_categories_features=vect.get_feature_names()

print('X_train_ohe_clean_categories',X_train_ohe_clean_categories.shape, y_tr
print('X_test_ohe_clean_categories',X_test_ohe_clean_categories.shape, y_test
# print('X_cv_ohe_clean_categories',X_cv_ohe_clean_categories.shape, y_cv.sha
print(clean_categories_features[:10])

```

```
X_train_ohe_clean_categories (33500, 40) (33500,)
X_test_ohe_clean_categories (16500, 40) (16500,)
['appliedlearning', 'appliedlearning health_sports', 'appliedlearning history_c
ivics', 'appliedlearning literacy_language', 'appliedlearning math_science', 'a
ppliedlearning music_arts', 'appliedlearning specialneeds', 'health_sports', 'h
ealth_sports appliedlearning', 'health_sports history_civics']
```

clean_subcategories-OHE

In [14]:
```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['clean_subcategories'].values)
X_train_ohe_clean_subcategories= vect.transform(X_train['clean_subcategories'
# X_cv_ohe_clean_subcategories= vect.transform(X_cv['clean_subcategories'].va
X_test_ohe_clean_subcategories= vect.transform(X_test['clean_subcategories'].
clean_subcategories_features=vect.get_feature_names()

print('X_train_ohe_clean_subcategories',X_train_ohe_clean_subcategories.shape
print('X_test_ohe_clean_subcategories',X_test_ohe_clean_subcategories.shape,
# print('X_cv_ohe_clean_subcategories',X_cv_ohe_clean_subcategories.shape, y_
print(clean_subcategories_features[:10])
```

```
X_train_ohe_clean_subcategories (33500, 181) (33500,)
X_test_ohe_clean_subcategories (16500, 181) (16500,)
['appliedsciences', 'appliedsciences charactereducation', 'appliedsciences coll
ege_careerprep', 'appliedsciences earlydevelopment', 'appliedsciences environme
ntalscience', 'appliedsciences esl', 'appliedsciences extracurricular', 'applie
dsciences health_lifescience', 'appliedsciences health_wellness', 'appliedscien
ces history_geography']
```

Normalize numerical features

price

In [15]:
```
from sklearn.preprocessing import Normalizer
norml=Normalizer()
norml.fit(X_train['price'].values.reshape(1,-1))
X_train_norml_price=norml.transform(X_train['price'].values.reshape(1,-1))
X_test_norml_price=norml.transform(X_test['price'].values.reshape(1,-1))
# X_cv_norml_price=norml.transform(X_cv['price'].values.reshape(1,-1))

#X_cv_norm_price=norm.transform(xcv['price'].values.reshape(-1,1))
print('X_train_norml_price shape',X_train_norml_price.shape,y_train.shape)
print('X_test_norml_price shape',X_test_norml_price.shape,y_test.shape)
# print('X_cv_norml_price shape',X_cv_norml_price.shape,y_cv.shape)

```

```
X_train_norml_price shape (1, 33500) (33500,)
X_test_norml_price shape (1, 16500) (16500,)
```

teacher_number_of_previously_posted_projects

In [16]:
```
norml.fit(X_train['teacher_number_of_previously_posted_projects'].values.resh
X_train_norml_teacher_number_of_previously_posted_projects=norml.transform(X_
X_test_norml_teacher_number_of_previously_posted_projects=norml.transform(X_t
# X_cv_norml_teacher_number_of_previously_posted_projects=norml.transform(X_c


# X_cv_norm_price=norm.transform(xcv['price'].values.reshape(-1,1))
print('X_train_norml_teacher_number_of_previously_posted_projects shape',X_tr
print('X_test_norml_teacher_number_of_previously_posted_projects shape',X_tes

```

```
X_train_norml_teacher_number_of_previously_posted_projects shape (1, 33500) (33
500,)
X_test_norml_teacher_number_of_previously_posted_projects shape (1, 16500) (165
00,)
```

SET-1

In [17]:
```
#  https://stackoverflow.com/a/19710648/4084039
# combine all features into one single set

X_train_norml_tnoprepst_projects=X_train_norml_teacher_number_of_previously_p
X_test_norml_tnoprepst_projects=X_test_norml_teacher_number_of_previously_pos
# X_cv_norml_tnoprepst_projects=X_cv_norml_teacher_number_of_previously_poste

X_train_norml_price= X_train_norml_price.reshape(33500,1)
X_test_norml_price=X_test_norml_price.reshape(16500,1)
# X_cv_norml_price=X_cv_norml_price.reshape(11055,1)

from scipy.sparse import hstack
X_tr_set1 = hstack((X_train_essay_tfidf,X_train_norml_price,X_train_norml_tnc
X_tst_set1= hstack((X_test_essay_tfidf,X_test_norml_price,X_test_norml_tnopre
# X_cr_set1= hstack((X_cv_essay_tfidf,X_cv_norml_price,X_cv_norml_tnoprepst_p


print("Final Data matrix")
print(X_tr_set1.shape, y_train.shape)
# print(X_cr_set1.shape, y_cv.shape)
print(X_tst_set1.shape, y_test.shape)

```

```
Final Data matrix
(33500, 50282) (33500,)
(16500, 50282) (16500,)
```

TFIDF W2V - EASSY

In [ ]:
```

```

In [18]:
```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [19]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(data['essay'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)
tfidf_words = set(tfidf_model.get_feature_names())
```

In [20]:
```
from tqdm import tqdm
# compute average word2vec for each review.
tfidf_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stor
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/revi
    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
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spli
            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_xtrain.append(vector)

print(len(tfidf_w2v_vectors_xtrain))
print(len(tfidf_w2v_vectors_xtrain[0]))

```

```
100%|████████████████████████████████| 33500/33500 [02:09<00:00, 259.57it/s]

33500
300
```

In [21]:
```
tfidf_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is store
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/revi
    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
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spli
            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_xtest.append(vector)

print(len(tfidf_w2v_vectors_xtest))
X_test_tfidf_w2v=len(tfidf_w2v_vectors_xtest)
```

```
100%|████████████████████████████████| 16500/16500 [01:00<00:00, 274.06it/s]

16500
```

SET-2

In [22]:
```
X_train_norml_tnoprepst_projects=X_train_norml_teacher_number_of_previously_p
X_test_norml_tnoprepst_projects=X_test_norml_teacher_number_of_previously_pos
# X_cv_norml_tnoprepst_projects=X_cv_norml_teacher_number_of_previously_poste

X_train_norml_price= X_train_norml_price.reshape(33500,1)
X_test_norml_price=X_test_norml_price.reshape(16500,1)
# X_cv_norml_price=X_cv_norml_price.reshape(11055,1)

from scipy.sparse import hstack
X_tr_set2 = hstack((tfidf_w2v_vectors_xtrain,X_train_norml_price,X_train_norm
X_tst_set2= hstack((tfidf_w2v_vectors_xtest,X_test_norml_price,X_test_norml_t
# X_cr_set1= hstack((X_cv_essay_tfidf,X_cv_norml_price,X_cv_norml_tnoprepst_p


print("Final Data matrix")
print(X_tr_set1.shape, y_train.shape)
# print(X_cr_set1.shape, y_cv.shape)
print(X_tst_set1.shape, y_test.shape)
```

```
Final Data matrix
(33500, 50282) (33500,)
(16500, 50282) (16500,)
```

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

Apply Decision Tree 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

finding best hyperparameter for set1

In [23]:
```
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import roc_auc_score
from sklearn import cross_validation
import math
import warnings
warnings.filterwarnings("ignore")

model=DecisionTreeClassifier(random_state=0)
parameters = {'max_depth':[1,5,10,50],'min_samples_split':[10,50,100,500]}
clf = RandomizedSearchCV(estimator = model,param_distributions = parameters,
clf.fit(X_tr_set1, y_train)
# clf.estimator
```

```
C:\Users\aaa\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: Deprec
ationWarning: This module was deprecated in version 0.18 in favor of the model_
selection module into which all the refactored classes and functions are moved.
Also note that the interface of the new CV iterators are different from that of
this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Out[23]:
```
RandomizedSearchCV(cv=None, error_score='raise',
          estimator=DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=0,
            splitter='best'),
          fit_params=None, iid=True, n_iter=10, n_jobs=1,
          param_distributions={'max_depth': [1, 5, 10, 50], 'min_samples_spli
t': [10, 50, 100, 500]},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring='roc_auc', verbose=0)
```

In [24]:
```
clf.estimator
```

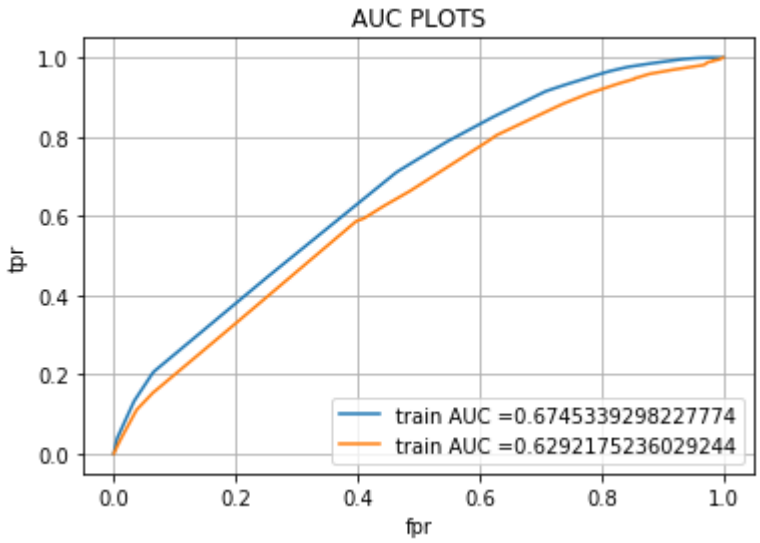
Out[24]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=0,
            splitter='best')
```

In [25]:
```
clf.best_params_
```

Out[25]:
```
{'min_samples_split': 500, 'max_depth': 10}
```

In [26]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
    # 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 - 490
    # 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

clf_set1=DecisionTreeClassifier(max_depth=clf.best_params_['max_depth'],min_s
clf_set1.fit(X_tr_set1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability esti
# not the predicted outputs

y_train_pred_set1 = batch_predict(clf_set1, X_tr_set1)
y_test_pred_set1 = batch_predict(clf_set1, X_tst_set1)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_set1)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_set1)

auc_set1=auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr))
plt.legend()
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title("AUC PLOTS")
plt.grid()
plt.show()


train_auc=roc_auc_score(y_train,y_train_pred_set1)
test_auc=roc_auc_score(y_test,y_test_pred_set1)
print("train_Auc:",train_auc)
print("test_Auc:",test_auc)
```




```
train_Auc: 0.6745339298227774
test_Auc: 0.6292175236029244
```

confusion matrix set1

In [27]:

```
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 h
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshol
    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)
# print("Train confusion matrix")
# print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_set1, best
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_set1, best_t))
```

```
the maximum value of tpr*(1-fpr) 0.38112492838830747 for threshold 0.868
Test confusion matrix
[[1591 1051]
 [5725 8133]]
```

finding best hyperparameter for set2

In [28]:
```
model=DecisionTreeClassifier(random_state=0)
parameters = {'max_depth':[1,5,10,50],'min_samples_split':[10,50,100,500]}
clf1 = RandomizedSearchCV(estimator = model,param_distributions = parameters,
clf1.fit(X_tr_set2, y_train)
# clf.estimator
```

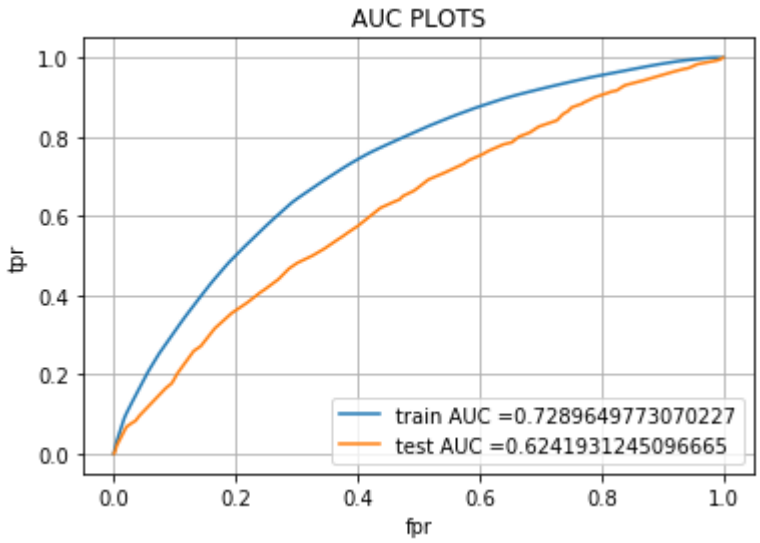
Out[28]:
```
RandomizedSearchCV(cv=None, error_score='raise',
          estimator=DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=0,
            splitter='best'),
          fit_params=None, iid=True, n_iter=10, n_jobs=1,
          param_distributions={'max_depth': [1, 5, 10, 50], 'min_samples_spli
t': [10, 50, 100, 500]},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring='roc_auc', verbose=0)
```

In [29]:
```
clf1.best_params_
```

Out[29]:
```
{'min_samples_split': 10, 'max_depth': 5}
```

In [30]:

```
def batch_predict(clf1, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
    # 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 - 490
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf1.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(clf1.predict_proba(data[tr_loop:])[:,1])


    return y_data_pred

clf_set2=DecisionTreeClassifier(max_depth=clf.best_params_['max_depth'],min_s
clf_set2.fit(X_tr_set2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability esti
# not the predicted outputs

y_train_pred_set2 = batch_predict(clf_set2, X_tr_set2)
y_test_pred_set2 = batch_predict(clf_set2, X_tst_set2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_set2)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_set2)

# auc_set1=auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title("AUC PLOTS")
plt.grid()
plt.show()

train_auc=roc_auc_score(y_train,y_train_pred_set2)
test_auc=roc_auc_score(y_test,y_test_pred_set2)
print("train_Auc:",train_auc)
print("test_Auc:",test_auc)
```




```
train_Auc: 0.7289649773070227
test_Auc: 0.6241931245096665
```

confusion matrix

In [31]:

```
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 h
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshol
    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)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_set2, best_t
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_set2, best_t))
```

```
the maximum value of tpr*(1-fpr) 0.45199795752439104 for threshold 0.846
Train confusion matrix
[[ 3576  1789]
 [ 9056 19079]]
Test confusion matrix
[[1524 1118]
 [5505 8353]]
```

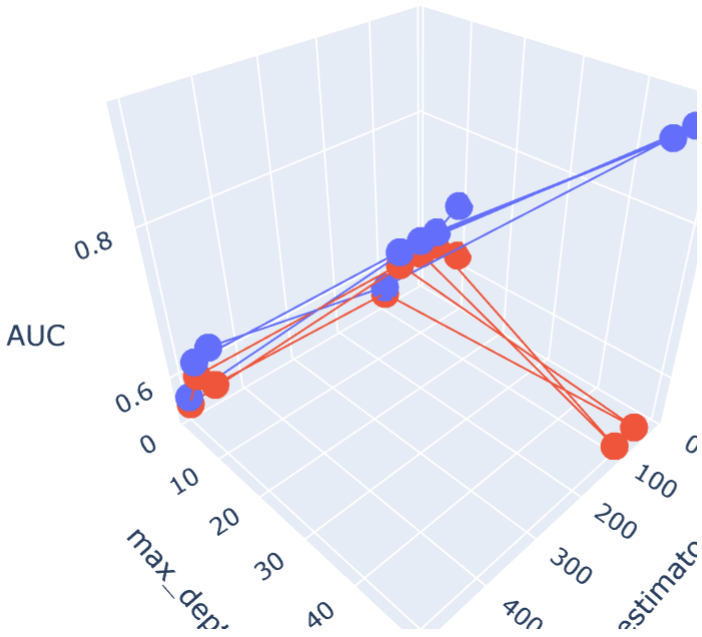
In [32]:
```
1 X_test.shape
```

Out[32]: (16500, 8)

3D map for set1

In [33]:

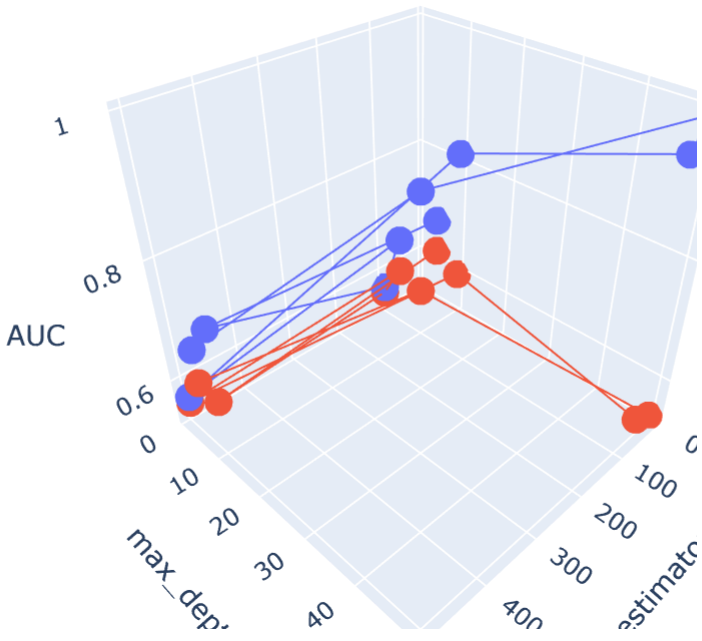
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

min_samples_split=clf.cv_results_['param_min_samples_split']
max_depth=clf.cv_results_['param_max_depth']
train_auc=clf.cv_results_['split0_train_score']
test_auc=clf.cv_results_['split0_test_score']

x = min_samples_split
y  = max_depth
z1 = train_auc
z2 = test_auc

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x,y=y,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x,y=y,z=z2, name = 'test')
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')

```



3D map for set2

In [34]:

```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

min_samples_split=clf1.cv_results_['param_min_samples_split']
max_depth=clf1.cv_results_['param_max_depth']
train_auc=clf1.cv_results_['split0_train_score']
test_auc=clf1.cv_results_['split0_test_score']

x = min_samples_split
y  = max_depth
z1 = train_auc
z2 = test_auc

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x,y=y,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x,y=y,z=z2, name = 'test')
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')

```



FALSE POSITIVE DATA POINTS SET 1

In [35]:
```
cloud1=X_test.copy(deep=True)
cloud2=X_test.copy(deep=True)
```

In [ ]:
```

```

In [36]:
```
ytp=[]
for i in y_test_pred_set1:
    if i<0.5:
        ytp.append(0)
    else:
        ytp.append(1)
cloud1['y_pred'] = ytp
cloud1['y_test'] = y_test
cloud1

# fp_list=[]
# for i in range(len(y_test)):
#     if y_test[i]==0 and y_test_pred_set1[i]==1:
#         fp_list.append(i)
# fp
```

Out[36]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_
36320	wa	mrs	grades_prek_2	
37485	ny	mrs	grades_prek_2	
41428	ca	mrs	grades_prek_2	
30726	va	mr	grades_3_5	

In [37]:
```
df1= cloud1[(cloud1['y_pred'] == 1) &  (cloud1['y_test']==0)]
df1
```

Out[37]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_
30726	va	mr	grades_3_5	
32090	ok	mrs	grades_9_12	
43254	ga	mr	grades_prek_2	
6686	ar	mrs	grades_prek_2	

In [38]:
```
df1.shape[0]
```

Out[38]: 2565

FALSE POSITIVE DATA POINTS SET 2

In [39]:
```
# cloud2=X_tst_set2.copy()
ytp=[]
for i in y_test_pred_set2:
    if i<0.5:
        ytp.append(0)
    else:
        ytp.append(1)
cloud2['y_pred'] = ytp
cloud2['y_test'] = y_test
cloud2

df2 = cloud2[(cloud2['y_pred'] == 1) &  (cloud2['y_test']==0)]
df2

```

Out[39]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_
30726	va	mr	grades_3_5	
32090	ok	mrs	grades_9_12	
43254	ga	mr	grades_prek_2	
4018	mo	mrs	grades_3_5	

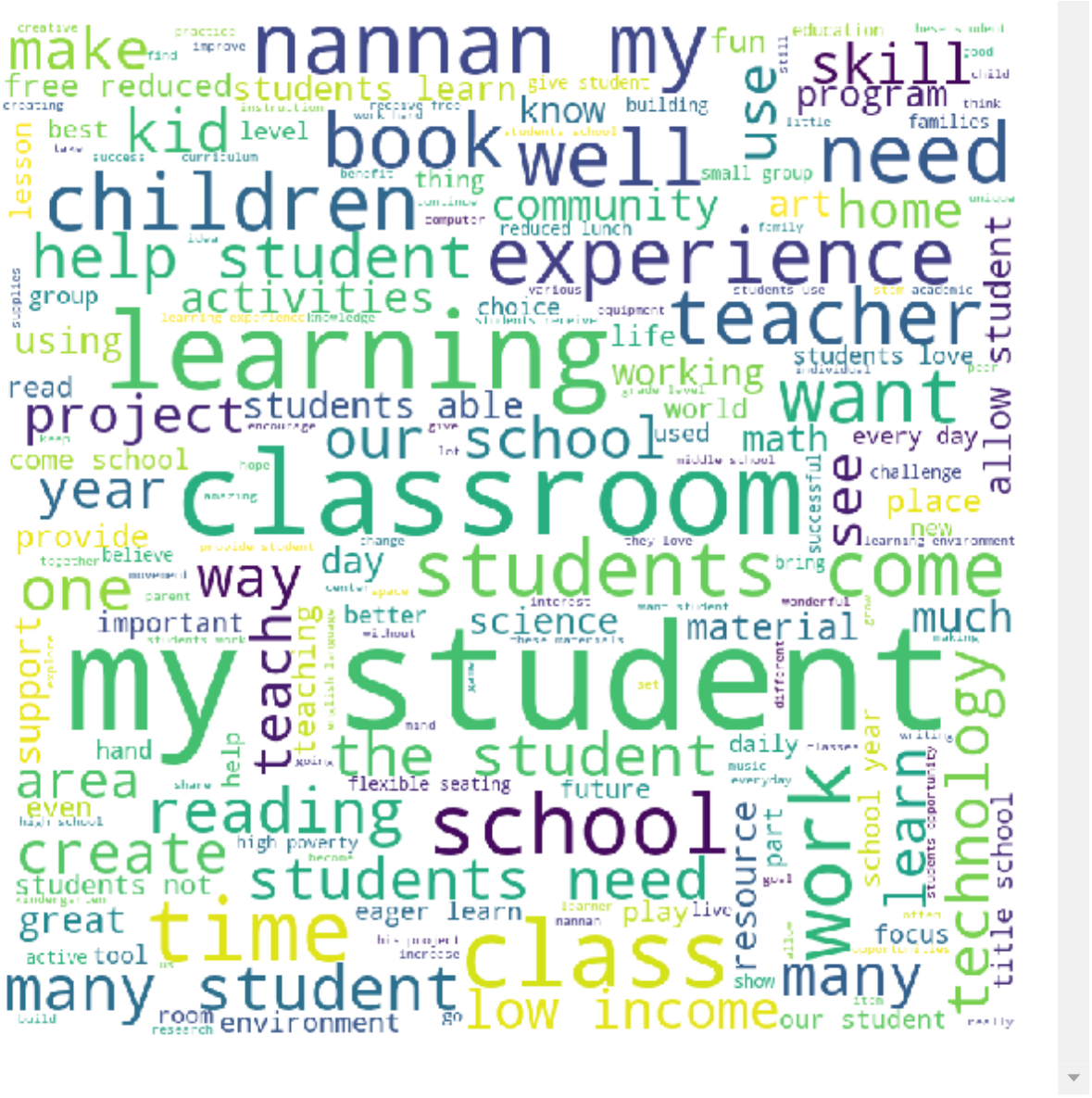
In [40]:
```
df2.shape[0]
```

Out[40]: 2525

plot wordcloud set 1

In [41]:

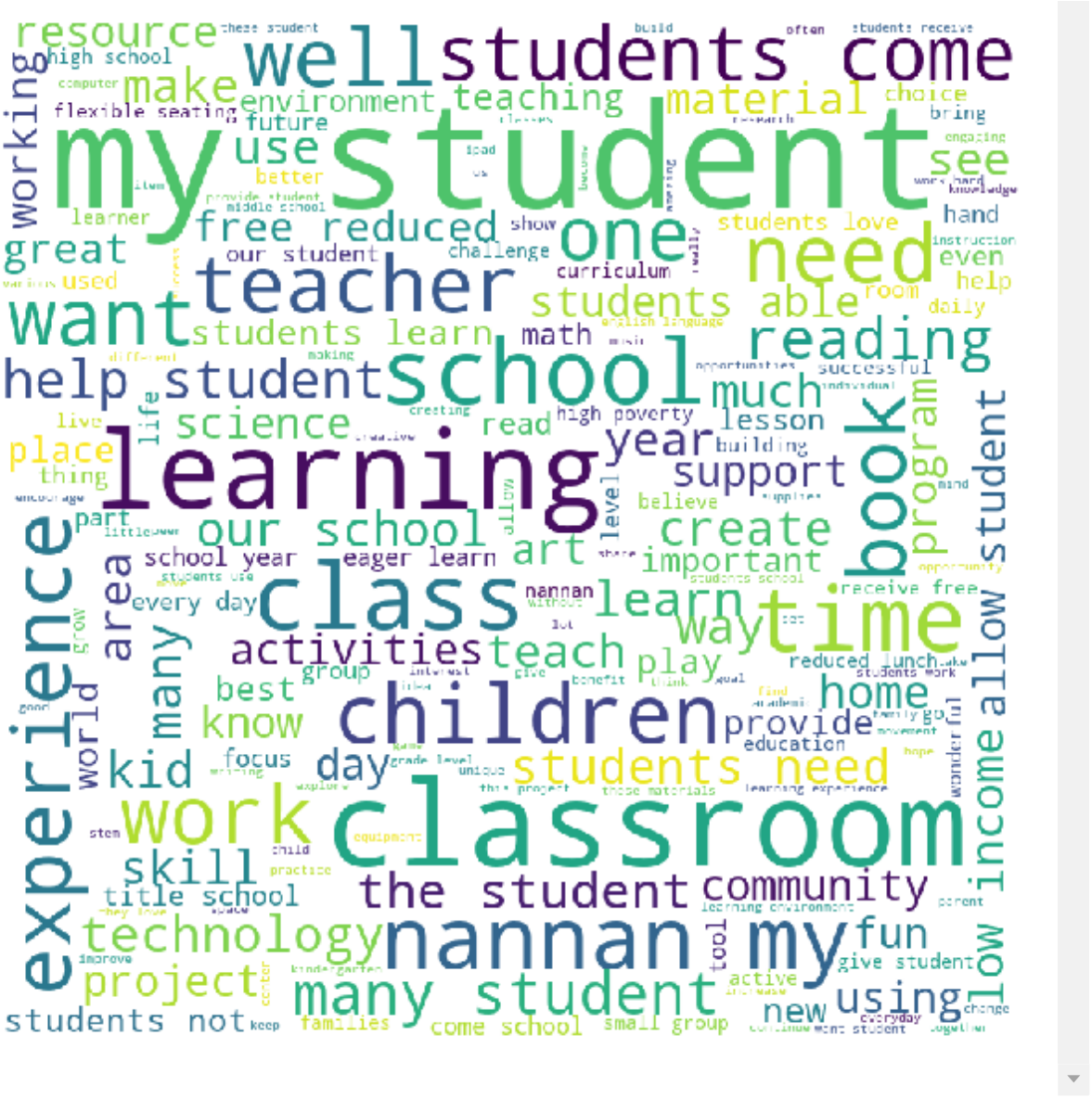
```
# https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in df1.essay:

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

    # split the value
    tokens = val.split()

    # 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()
```



plot wordcloud set2

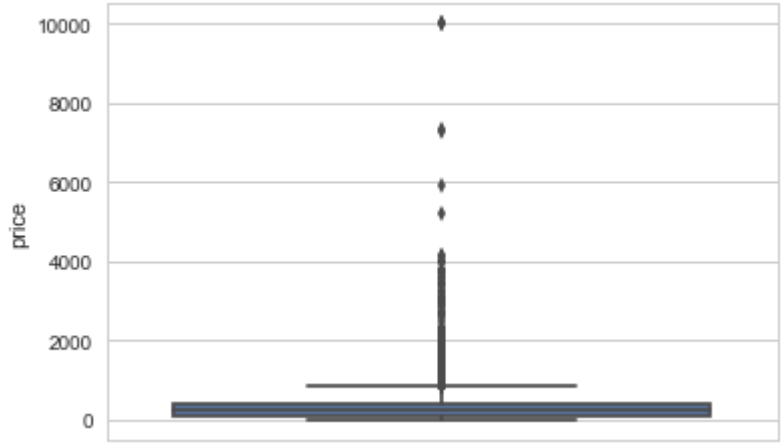
In [42]:

```
# https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in df2.essay:

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

    # split the value
    tokens = val.split()

    # 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()
```



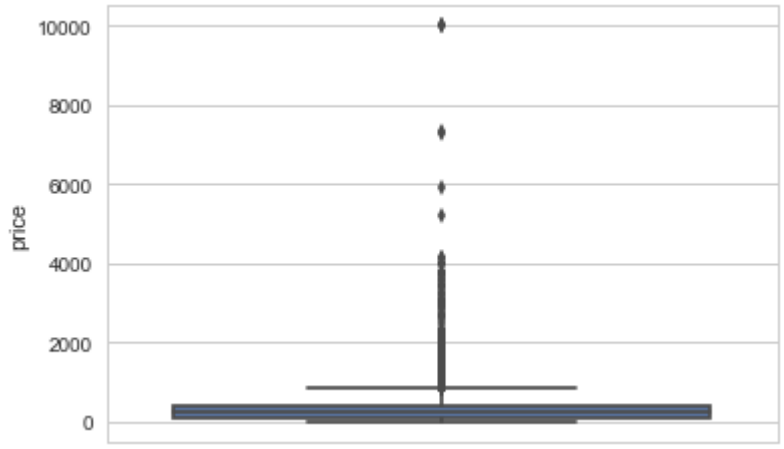
Plot the box plot with the `price` of these `false positive data points` set1

In [43]:
```
import seaborn as sns
sns.set(style='whitegrid')
ax = sns.boxplot(y=cloud1["price"])

```



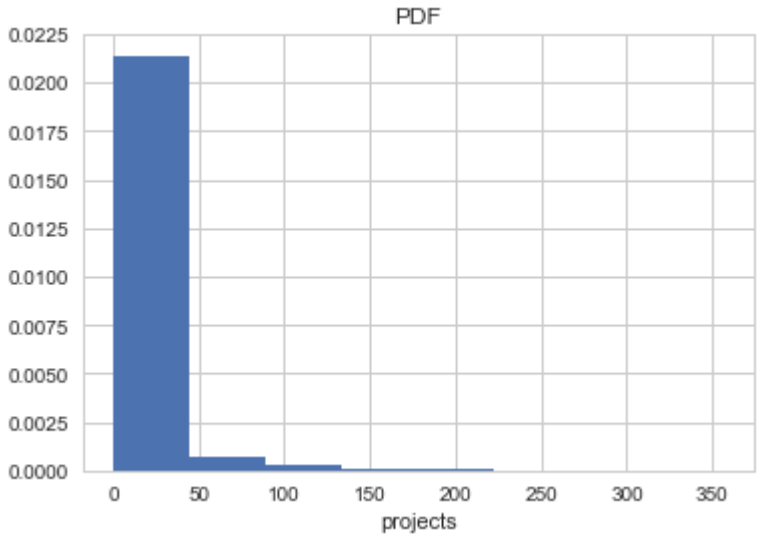
Plot the box plot with the price of these false positive data points set 2

In [44]:
```
import seaborn as sns
sns.set(style='whitegrid')
ax = sns.boxplot(y=cloud2["price"])
```



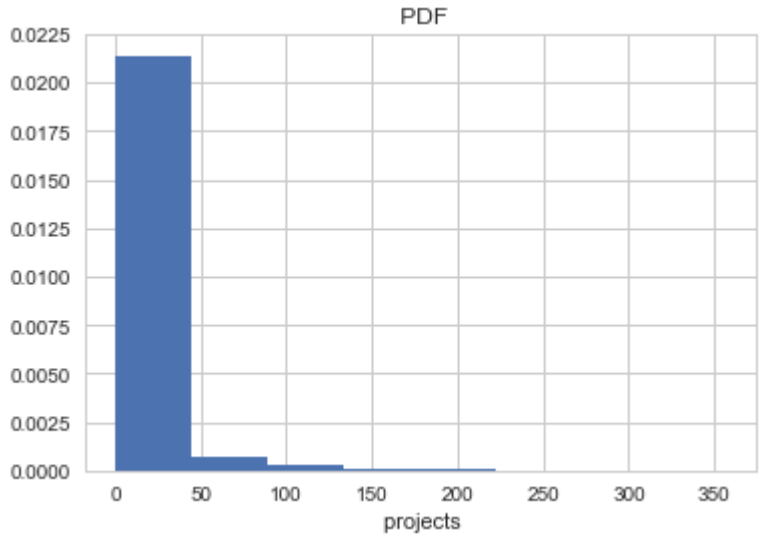
Plot the pdf set1

In [45]:
```
plt.hist(cloud1['teacher_number_of_previously_posted_projects'], bins=8, dens
# pdf = counts/(sum(counts))
# print(pdf)
plt.title('PDF')
plt.xlabel('projects')
plt.show()
# plt.plot(bin_edges[1:], cdf)
```



Plot the pdf set2

In [46]:
```
plt.hist(cloud2['teacher_number_of_previously_posted_projects'], bins=8, dens
# pdf = counts/(sum(counts))
# print(pdf)
plt.title(' PDF')
plt.xlabel(' projects')
plt.show()
# plt.plot(bin_edges[1:], cdf)
```



In [ ]:
```
1
```

1.6 Getting top features using `feature_importances_`

TASK 2

In [47]:
```
imp=[]
for i in clf_set1.feature_importances_:
    if i!=0.0:
        imp.append(i)
imp
len(imp)


X_train_task2=X_tr_set1[: ,imp]
X_test_task2=X_tst_set1[: ,imp]

print(X_train_task2.shape)
print(X_test_task2.shape)
print(y_train.shape)
```

```
(33500, 61)
(16500, 61)
(33500,)
```

In [48]:
```
model=DecisionTreeClassifier(random_state=0)
parameters = {'max_depth':[1,5,10,50],'min_samples_split':[10,50,100,500]}
clf_featureimp = RandomizedSearchCV(estimator = model,param_distributions = p
clf_featureimp.fit(X_train_task2,y_train)
```

Out[48]:
```
RandomizedSearchCV(cv=None, error_score='raise',
          estimator=DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=0,
            splitter='best'),
          fit_params=None, iid=True, n_iter=10, n_jobs=1,
          param_distributions={'max_depth': [1, 5, 10, 50], 'min_samples_spli
t': [10, 50, 100, 500]},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring='roc_auc', verbose=0)
```

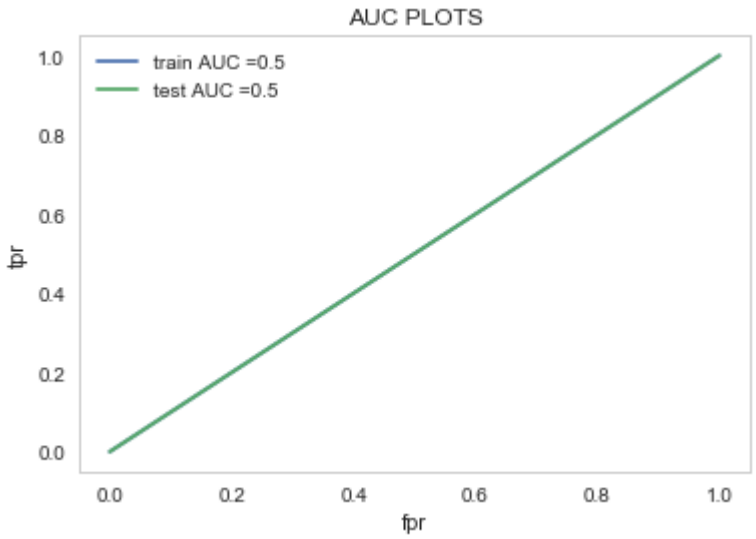
In [49]:
```
clf_featureimp.best_params_
```

Out[49]:
```
{'min_samples_split': 50, 'max_depth': 1}
```

Decision tree for hyperparameter tunning

In [50]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
    # 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 - 490
    # 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 auc

best_model=DecisionTreeClassifier(max_depth=clf_featureimp.best_params_['max_
best_model.fit(X_train_task2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability esti
# not the predicted outputs

y_train_pred = best_model.predict( X_train_task2)
y_test_pred = best_model.predict( X_test_task2)


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



# auc=auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title("AUC PLOTS")
plt.grid()


plt.show()


train_auc=roc_auc_score(y_train,y_train_pred)
test_auc=roc_auc_score(y_test,y_test_pred)
print("train_Auc:",train_auc)
print("test_Auc:",test_auc)
```




```
train_Auc: 0.5
test_Auc: 0.5
```

confusion matrix for task 2

In [51]:

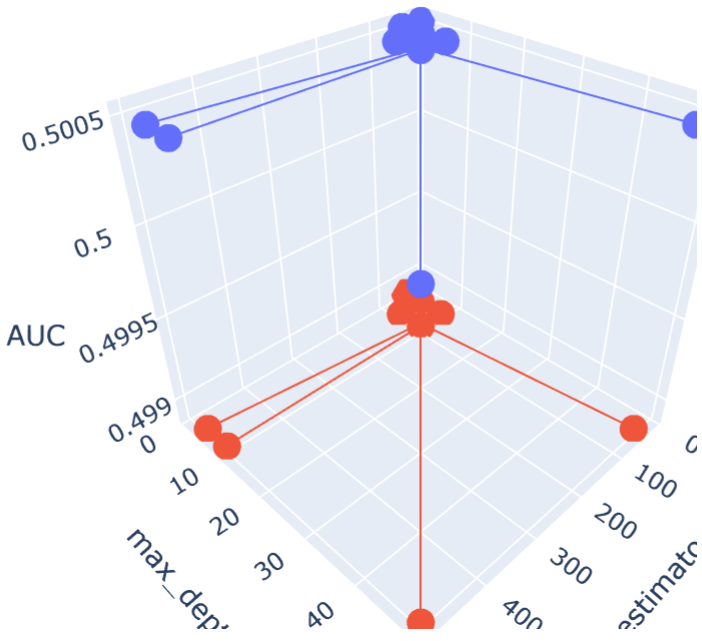
```
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very h
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshol
    return t

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)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_set2, best_t
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_set2, best_t))
```

```
the maximum value of tpr*(1-fpr) 0.0 for threshold 2
Train confusion matrix
[[ 5365     0]
 [28135     0]]
Test confusion matrix
[[ 2642     0]
 [13858     0]]
```

3D map for task 2

In [52]:

```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

min_samples_split=clf_featureimp.cv_results_['param_min_samples_split']
max_depth=clf_featureimp.cv_results_['param_max_depth']
train_auc=clf_featureimp.cv_results_['split0_train_score']
test_auc=clf_featureimp.cv_results_['split0_test_score']

x = min_samples_split
y  = max_depth
z1 = train_auc
z2 = test_auc

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x,y=y,z=z1, name = 'train')
trace2 = go.Scatter3d(x=x,y=y,z=z2, name = 'test')
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')

```



fp for task 2

In [53]:
```
1 X_test_task2.shape
```

Out[53]: (16500, 61)

In [54]:
```
1 cloud3=X_test.copy()
2 cloud3
```

Out[54]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_
36320	wa	mrs	grades_prek_2	
37485	ny	mrs	grades_prek_2	
41428	ca	mrs	grades_prek_2	
30726	va	mr	grades_3_5	

In [55]:
```
# cloud3=X_test_task2.copy()

ytp=[]
for i in y_test_pred:
    if i<0.5:
        ytp.append(0)
    else:
        ytp.append(1)
cloud3['y_pred'] = ytp
cloud3['y_test'] = y_test
cloud3

df3 = cloud3[(cloud3['y_pred'] == 1) &  (cloud3['y_test']==0)]
df3
```

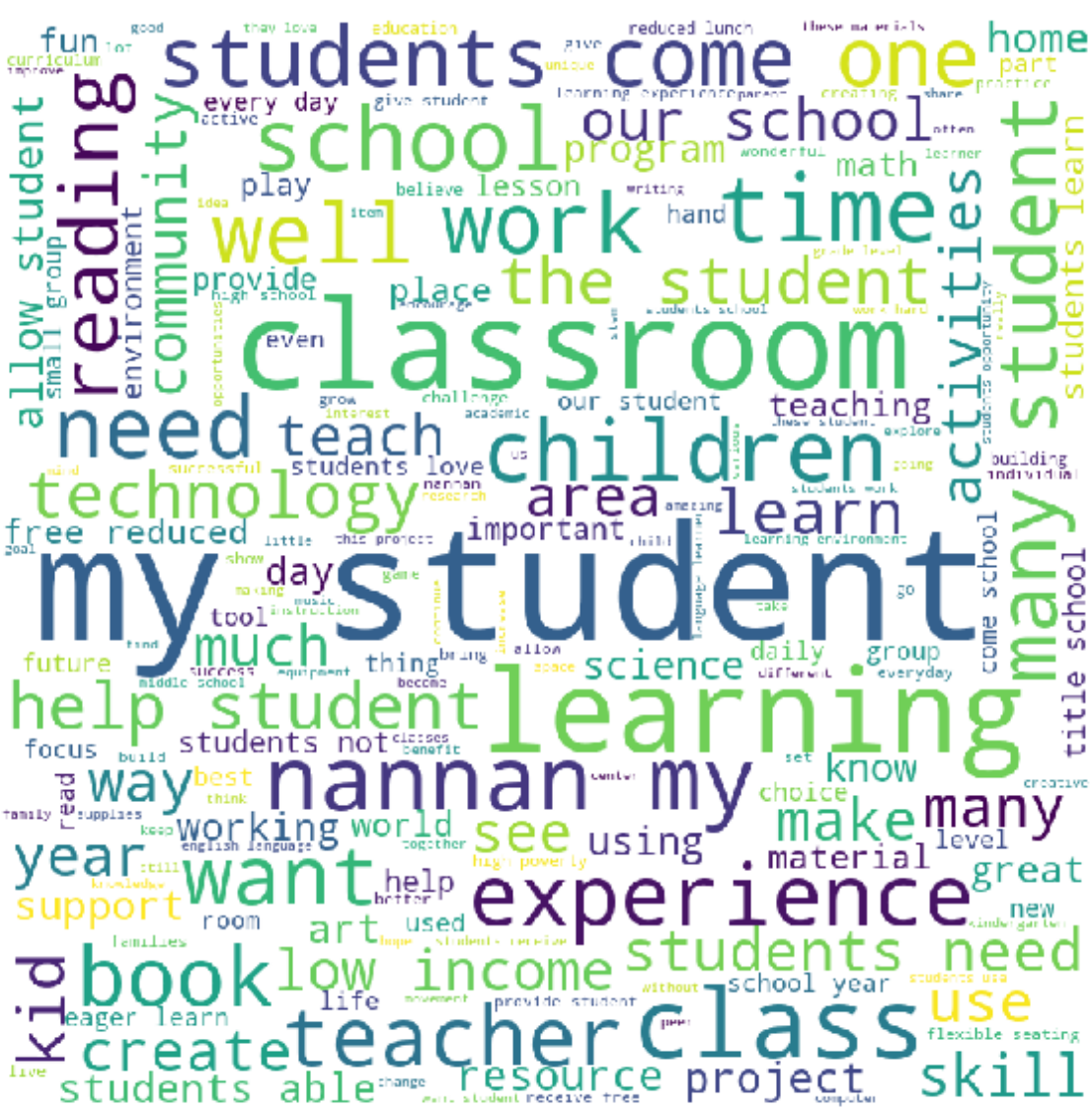
Out[55]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_
30726	va	mr	grades_3_5	
32090	ok	mrs	grades_9_12	
43254	ga	mr	grades_prek_2	
6686	ar	mrs	grades_prek_2	

wordcloud for task 2

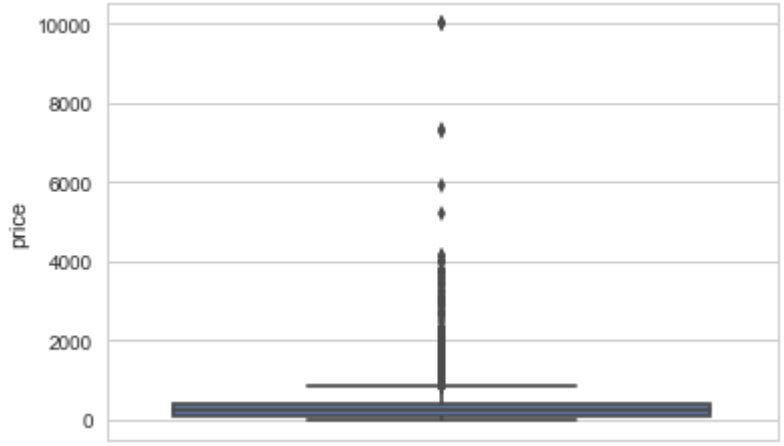
In [56]:

```
# https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in df3.essay:

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

    # split the value
    tokens = val.split()

    # 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()
```



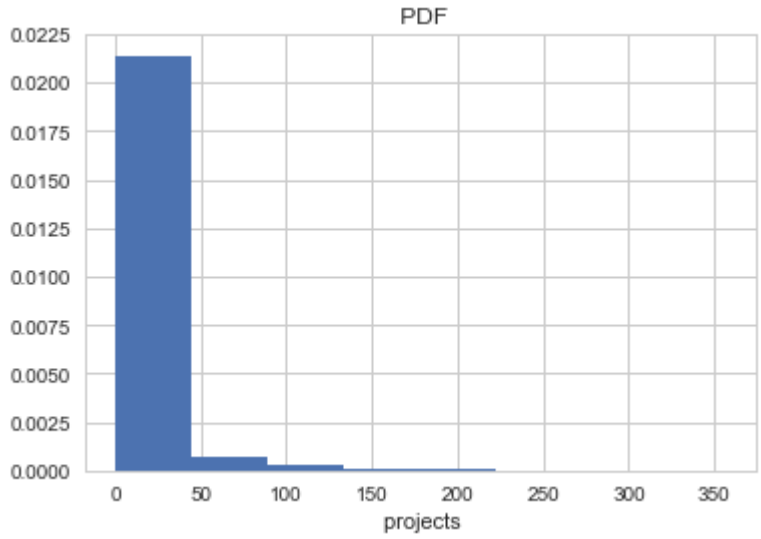
BOX plot for task 2

In [57]:
```
import seaborn as sns
sns.set(style='whitegrid')
ax = sns.boxplot(y=cloud3["price"])

```



PDF for task 2

In [58]:
```
plt.hist(cloud3['teacher_number_of_previously_posted_projects'], bins=8, dens
# pdf = counts/(sum(counts))
# print(pdf)
plt.title(' PDF')
plt.xlabel(' projects')
plt.show()
# plt.plot(bin_edges[1:], cdf)
```



2. Summary

In [59]:
```
from prettytable import PrettyTable
pryt=PrettyTable()
pryt.field_names=['vectorizer','model','best hyper params','train-AUC','test-
pryt.add_row(['TF-IDF','Decision Tree',clf.best_params_,train_auc,test_auc])
pryt.add_row(['TF-IDF W2V','Decision Tree',clf1.best_params_,train_auc,test_a
pryt.add_row(['TF-IDF','Decision Tree',clf_featureimp.best_params_,train_auc,
print(pryt)

```

```
+------------+---------------+-------------------------------------------+---
----------------------------------------------------------------+-------------
-------------------------------------------------------+
| vectorizer |     model     |              best hyper params              |
train-AUC                                |                                  test-AUC
|
+------------+---------------+-------------------------------------------+---
----------------------------------------------------------------+-------------
-------------------------------------------------------+
|   TF-IDF   | Decision Tree | {'min_samples_split': 500, 'max_depth': 10} |
[0.50050704 0.50050704 0.50050704 0.50050704 0.50050704 0.50050704 | [0.4988965
7 0.49889657 0.49889657 0.49889657 0.49889657 0.49889657 |
|            |               |                                                |
0.50050704 0.50050704 0.50050704 0.50050704]           |            0.49889657
0.49889657 0.49889657 0.49889657]            |
| TF-IDF W2V | Decision Tree |  {'min_samples_split': 10, 'max_depth': 5}  |
[0.50050704 0.50050704 0.50050704 0.50050704 0.50050704 0.50050704 | [0.4988965
7 0.49889657 0.49889657 0.49889657 0.49889657 0.49889657 |
|            |               |                                                |
0.50050704 0.50050704 0.50050704 0.50050704]           |            0.49889657
0.49889657 0.49889657 0.49889657]            |
|   TF-IDF   | Decision Tree |  {'min_samples_split': 50, 'max_depth': 1}  |
[0.50050704 0.50050704 0.50050704 0.50050704 0.50050704 0.50050704 | [0.4988965
7 0.49889657 0.49889657 0.49889657 0.49889657 0.49889657 |
|            |               |                                                |
0.50050704 0.50050704 0.50050704 0.50050704]           |            0.49889657
0.49889657 0.49889657 0.49889657]            |
+------------+---------------+-------------------------------------------+---
----------------------------------------------------------------+-------------
-------------------------------------------------------+
```

In [ ]:
```

```