prabhudayala@gmail.com_8

May 9, 2019

1 DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

```
Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result

How to scale current manual processes and resources to screen 500,000 projects so that they ca
<li>How to increase the consistency of project vetting across different volunteers to improve
<li>How to focus volunteer time on the applications that need the most assistance</li>
</ul>
```

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

1.1 About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`

`project_title` | Title of the project. **Examples:**
Art Will Make You Happy!
First Grade Fun
`project_grade_category` | Grade level of students for which the project is targeted. One of the following enumerated values:
Grades PreK-2
Grades 3-5
Grades 6-8
Grades 9-12
`project_subject_categories` | One or more (comma-separated) subject categories for the project from the following enumerated list of values:

Applied Learning
Care & Hunger
Health & Sports
History & Civics
Literacy & Language
Math & Science
Music & The Arts
Special Needs
Warmth
Examples:
Music & The Arts
Literacy & Language, Math & Science

`school_state` | State where school is located (Two-letter U.S. postal code). **Example:** `WY`

`project_subject_subcategories` | One or more (comma-separated) subject subcategories for the project. **Examples:**

Literacy
Literature & Writing, Social Sciences

`project_resource_summary` | An explanation of the resources needed for the project. **Example:**

My students need hands on literacy materials to manage sensory needs!

`project_essay_1` | First application essay

`project_essay_2` | *Second application essay* `project_essay_3` | Third application essay ***`project_essay_4`*** | *Fourth application essay* `project_submitted_datetime` | Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`

`teacher_id` | A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`

`teacher_prefix` | Teacher's title. One of the following enumerated values:

nan
Dr.
Mr.
Mrs.
Ms.
Teacher.

`teacher_number_of_previously_posted_projects` | Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section Notes on the Essay Data for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`

Feature	Description
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

1.1.1 Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

project_essay_1: "Introduce us to your classroom"

project_essay_2: "Tell us more about your students"

project_essay_3: "Describe how your students will use the materials you're requesting"

project_essay_3: "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

project_essay_1: "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

project_essay_2: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

```
In [1]: %matplotlib inline
        import warnings
```

```
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
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
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
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
```

1.2 1.1 Reading Data

```
In [2]: project_data = pd.read_csv('train_data.csv')
        resource_data = pd.read_csv('resources.csv')
```

```
In [3]: print("Number of data points in train data", project_data.shape)
        print('-'*50)
        print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
```

```
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

```
In [4]: print("Number of data points in train data", resource_data.shape)
        print(resource_data.columns.values)
        resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

```
Out[4]:         id                                           description  quantity  \
        0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
        1  p069063         Bouncy Bands for Desks (Blue support pipes)         3

            price
        0  149.00
        1   14.95
```

```
In [5]: # join two dataframes in python:
        price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_
        price_data.head(2)
        project_data = pd.merge(project_data, price_data, on='id', how='left')
```

```
In [6]: project_data.columns
```

```
Out[6]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
               'project_submitted_datetime', 'project_grade_category',
               'project_subject_categories', 'project_subject_subcategories',
               'project_title', 'project_essay_1', 'project_essay_2',
               'project_essay_3', 'project_essay_4', 'project_resource_summary',
               'teacher_number_of_previously_posted_projects', 'project_is_approved',
               'price', 'quantity'],
              dtype='object')
```

1.3 1.2 preprocessing of `project_subject_categories`

```
In [7]: catogories = list(project_data['project_subject_categories'].values)
        # remove special characters from list of strings python: https://stackoverflow.com/a/4

        # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
        # https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-str
        # https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyt
        cat_list = []
```

```
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
        if 'The' in j.split(): # this will split each of the catogory based on space "
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing sp
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.4 1.3 preprocessing of `project_subject_subcategories`

In [8]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/4

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-str
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyt

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
        if 'The' in j.split(): # this will split each of the catogory based on space "
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing sp
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/408403
my_counter = Counter()
```

```
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.5 1.3 Text preprocessing

```
In [9]: # merge two column text dataframe:
        project_data["essay"] = project_data["project_essay_1"].map(str) +\
                                project_data["project_essay_2"].map(str) + \
                                project_data["project_essay_3"].map(str) + \
                                project_data["project_essay_4"].map(str)

In [10]: project_data.head(2)

Out[10]:    Unnamed: 0       id                         teacher_id teacher_prefix  \
         0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
         1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.

           school_state project_submitted_datetime project_grade_category  \
         0           IN        2016-12-05 13:43:57          Grades PreK-2
         1           FL        2016-10-25 09:22:10             Grades 6-8

                                          project_title  \
         0  Educational Support for English Learners at Home
         1            Wanted: Projector for Hungry Learners

                                         project_essay_1  \
         0  My students are English learners that are work...
         1  Our students arrive to our school eager to lea...

                                         project_essay_2 project_essay_3  \
         0  \"The limits of your language are the limits o...            NaN
         1  The projector we need for our school is very c...            NaN

           project_essay_4                         project_resource_summary  \
         0             NaN  My students need opportunities to practice beg...
         1             NaN  My students need a projector to help with view...

            teacher_number_of_previously_posted_projects  project_is_approved  price  \
         0                                             0                   0  154.6
         1                                             7                   1  299.0

            quantity             clean_categories           clean_subcategories  \
         0        23            Literacy_Language                   ESL Literacy
         1         1  History_Civics Health_Sports  Civics_Government TeamSports
```

```
                                                   essay
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...
```

In [11]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [12]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, a
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogn
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The g
==================================================
```

In [13]:
```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

```
In [14]: sent = decontracted(project_data['essay'].values[20000])
         print(sent)
         print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogn
==================================================
```

```
In [15]: # \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-p
         sent = sent.replace('\\r', ' ')
         sent = sent.replace('\\"', ' ')
         sent = sent.replace('\\n', ' ')
         print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogn
```

```
In [16]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
         sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
         print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cogni
```

```
In [17]: # https://gist.github.com/sebleier/554280
         # we are removing the words from the stop words list: 'no', 'nor', 'not'
         stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'r
                     "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him'
                     'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
                     'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "t
                     'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'h
                     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
                     'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'throug
                     'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'o
                     'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
                     'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
                     's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'n
                     've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't"
                     "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mig
                     "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
                     'won', "won't", 'wouldn', "wouldn't"]
```

```
In [18]: # Combining all the above stundents
         from tqdm import tqdm
         preprocessed_essays = []
         # tqdm is for printing the status bar
         for sentance in tqdm(project_data['essay'].values):
             sent = decontracted(sentance)
             sent = sent.replace('\\r', ' ')
```

```
            sent = sent.replace('\\"', ' ')
            sent = sent.replace('\\n', ' ')
            sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
            # https://gist.github.com/sebleier/554280
            sent = ' '.join(e for e in sent.split() if e not in stopwords)
            preprocessed_essays.append(sent.lower().strip())
```

```
100%|| 109248/109248 [00:43<00:00, 2493.63it/s]
```

```
In [19]: # after preprocesing
         preprocessed_essays[20000]
```

```
Out[19]: 'my kindergarten students varied disabilities ranging speech language delays cognitive
```

```
In [20]: project_data["essay"]=preprocessed_essays
```

1.4 Preprocessing of project_title

```
In [21]: from tqdm import tqdm
         preprocessed_project_title = []
         # tqdm is for printing the status bar
         for sentance in tqdm(project_data['project_title'].values):
             sent = decontracted(sentance)
             sent = sent.replace('\\r', ' ')
             sent = sent.replace('\\"', ' ')
             sent = sent.replace('\\n', ' ')
             sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
             # https://gist.github.com/sebleier/554280
             sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
             preprocessed_project_title.append(sent.lower().strip())
```

```
100%|| 109248/109248 [00:01<00:00, 56264.80it/s]
```

```
In [22]: print(project_data['project_title'].values[20000])
         project_data['project_title']=preprocessed_project_title
         print(project_data['project_title'].values[20000])
```

```
We Need To Move It While We Input It!
need move input
```

__ Computing Sentiment Scores__

```
In [23]: from nltk.sentiment import SentimentIntensityAnalyzer as SID
         #nltk.download('vader_lexicon')
         new_df_as_dictinary=[]
         sid=SID()
         for i in tqdm(project_data.essay.values):
             new_df_as_dictinary.append(sid.polarity_scores(i))
```

```
100%|| 109248/109248 [02:26<00:00, 746.33it/s]
```

```
In [24]: print(project_data.columns)
         print(project_data.shape)
         sentiment_score=pd.DataFrame(new_df_as_dictinary)
         print(sentiment_score.columns)
         print(sentiment_score.shape)
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'essay'],
      dtype='object')
(109248, 20)
Index(['compound', 'neg', 'neu', 'pos'], dtype='object')
(109248, 4)
```

```
In [25]: sentiment_score=pd.DataFrame(new_df_as_dictinary)
         project_data=pd.concat((project_data,sentiment_score),axis=1,ignore_index=True)
         print(project_data.shape)
```

```
(109248, 24)
```

```
In [26]: project_data.columns=['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_sta
                'project_submitted_datetime', 'project_grade_category', 'project_title',
                'project_essay_1', 'project_essay_2', 'project_essay_3',
                'project_essay_4', 'project_resource_summary',
                'teacher_number_of_previously_posted_projects', 'project_is_approved',
                'price', 'quantity', 'clean_categories', 'clean_subcategories',
                'essay','compound', 'neg', 'neu', 'pos']
```

2 Assignment 8: DT

```
<li><strong>Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets</stro
    <ul>
        <li><font color='red'>Set 1</font>: categorical, numerical features + project_title(BO
        <li><font color='red'>Set 2</font>: categorical, numerical features + project_title(TF
        <li><font color='red'>Set 3</font>: categorical, numerical features + project_title(AV
        <li><font color='red'>Set 4</font>: categorical, numerical features + project_title(TF
</li>
<br>
<li><strong>Hyper paramter tuning (best `depth` in range [1, 5, 10, 50, 100, 500, 100], and th
```

```
    <ul>
<li>Find the best hyper parameter which will give the maximum <a href='https://www.appliedaicou
<li>Find the best hyper paramter using k-fold cross validation or simple cross validation data
<li>Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this ta
    </ul>
</li>
<br>
<li><strong>Graphviz</strong>
    <ul>
<li>Visualize your decision tree with Graphviz. It helps you to understand how a decision is be
<li>Since feature names are not obtained from word2vec related models, visualize only BOW & TFI
<li>Make sure to print the words in each node of the decision tree instead of printing its inde
<li>Just for visualization purpose, limit max_depth to 2 or 3 and either embed the generated im
    </ul>
</li>
<br>
<li>
<strong>Representation of results</strong>
    <ul>
<li>You need to plot the performance of model both on train data and cross validation data for
<img src='train_cv_auc.JPG' width=300px></li>
<li>Once after you found the best hyper parameter, you need to train your model with it, and fi
<img src='train_test_auc.JPG' width=300px></li>
<li>Along with plotting ROC curve, you need to print the <a href='https://www.appliedaicourse.c
<img src='confusion_matrix.png' width=300px></li>
<li>Once after you plot the confusion matrix with the test data, get all the `false positive da
    <ul>
        <li> Plot the WordCloud <a href='https://www.geeksforgeeks.org/generating-word-cloud-py
        <li> Plot the box plot with the `price` of these `false positive data points`</li>
        <li> Plot the pdf with the `teacher_number_of_previously_posted_projects` of these `fal
    </ul>
    </ul>
</li>
<br>
<li><strong>[Task-2]</strong>
    <ul>
<li> Select 5k best features from features of <font color='red'>Set 2</font> using<a href='http
    </ul>
<br>
<li><strong>Conclusion</strong>
    <ul>
<li>You need to summarize the results at the end of the notebook, summarize it in the table for
    <img src='summary.JPG' width=400px>
</li>
    </ul>
```

2. Decision Tree

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

```
In [27]: project_data.columns
```

```
Out[27]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
                'project_submitted_datetime', 'project_grade_category', 'project_title',
                'project_essay_1', 'project_essay_2', 'project_essay_3',
                'project_essay_4', 'project_resource_summary',
                'teacher_number_of_previously_posted_projects', 'project_is_approved',
                'price', 'quantity', 'clean_categories', 'clean_subcategories', 'essay',
                'compound', 'neg', 'neu', 'pos'],
               dtype='object')
```

```
In [28]: sampling=False
         undersampling=True
         if (not sampling):
             print("Total data ",project_data.shape)

         else:
             if(sampling and undersampling):
                 print("Total data ",project_data.shape)
                 project_data_negative=project_data[project_data.project_is_approved==0]
                 project_data_positive=project_data[project_data.project_is_approved==1]
                 project_data_positive=project_data_positive.sample(n=project_data_negative.sha
                 print("Positive points: ",project_data_positive.shape[0])
                 print("Negaitive points: ",project_data_negative.shape[0])
                 project_data=pd.concat([project_data_positive,project_data_negative])
             else:
                 print("Total data ",project_data.shape)
                 project_data_negative=project_data[project_data.project_is_approved==0]
                 project_data_positive=project_data[project_data.project_is_approved==1]
                 project_data_negative=project_data_negative.sample(n=project_data_positive.sha
                 print("Positive points: ",project_data_positive.shape[0])
                 print("Negaitive points: ",project_data_negative.shape[0])
                 project_data=pd.concat([project_data_positive,project_data_negative])

         #data_point_size=50000
         #project_data=project_data.sample(n=data_point_size,random_state=42,replace=True)
         print("positive and negative counts")
         print(project_data.project_is_approved.value_counts())
         project_data_Y=project_data.project_is_approved
         project_data_X=project_data.drop(columns=['project_is_approved'])
         print("After sampling: ",project_data_X.shape)
```

```
Total data  (109248, 24)
positive and negative counts
1    92706
0    16542
Name: project_is_approved, dtype: int64
After sampling:  (109248, 23)
```

```
In [29]: from sklearn.model_selection import train_test_split
         project_data_X_train,project_data_X_test,project_data_Y_train,project_data_Y_test=tra
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 Categorical features

```
In [30]: from sklearn.feature_extraction.text import CountVectorizer
         vectorizer_clean_categories = CountVectorizer(vocabulary=list(sorted_cat_dict.keys())
         vectorizer_clean_categories.fit(project_data_X_train['clean_categories'].values)
         print(vectorizer_clean_categories.get_feature_names())

         #for train data
         categories_one_hot_train = vectorizer_clean_categories.transform(project_data_X_train
         print("Shape of matrix after one hot encodig ",categories_one_hot_train.shape)


         #for test
         categories_one_hot_test = vectorizer_clean_categories.transform(project_data_X_test['
         print("Shape of matrix after one hot encodig ",categories_one_hot_test.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', '
Shape of matrix after one hot encodig  (87398, 9)
Shape of matrix after one hot encodig  (21850, 9)
```

```
In [31]: vectorizer_clean_subcategories = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.
         vectorizer_clean_subcategories.fit(project_data_X_train['clean_subcategories'].values)
         print(vectorizer_clean_subcategories.get_feature_names())

         #for train data
         sub_categories_one_hot_train = vectorizer_clean_subcategories.transform(project_data_
         print("Shape of matrix after one hot encodig ",sub_categories_one_hot_train.shape)

         #for test
         sub_categories_one_hot_test = vectorizer_clean_subcategories.transform(project_data_X_
         print("Shape of matrix after one hot encodig ",sub_categories_one_hot_test.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
Shape of matrix after one hot encodig  (87398, 30)
Shape of matrix after one hot encodig  (21850, 30)
```

```
In [32]: project_data_X_train.teacher_prefix = project_data_X_train.teacher_prefix.replace(np.n
         print(project_data_X_train.teacher_prefix.value_counts())
         project_data_X_test.teacher_prefix = project_data_X_test.teacher_prefix.replace(np.nan
         print(project_data_X_test.teacher_prefix.value_counts())
```

```
Mrs.       45800
Ms.        31168
```

```
Mr.          8519
Teacher      1898
Dr.            11
                2
Name: teacher_prefix, dtype: int64
Mrs.        11469
Ms.          7787
Mr.          2129
Teacher       462
Dr.             2
                1
Name: teacher_prefix, dtype: int64
```

In [33]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_teacher_prefix = CountVectorizer(vocabulary=['Mrs.','Ms.','Mr.','Teacher',
vectorizer_teacher_prefix.fit(project_data_X_train['teacher_prefix'].values)
print(vectorizer_teacher_prefix.get_feature_names())

teacher_prefix_one_hot_train = vectorizer_teacher_prefix.transform(project_data_X_tra
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot_train.shape)

teacher_prefix_one_hot_test = vectorizer_teacher_prefix.transform(project_data_X_test
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot_test.shape)
```

```
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (87398, 5)
Shape of matrix after one hot encodig  (21850, 5)
```

In [34]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_project_grade_category = CountVectorizer(vocabulary=list(project_data_X_tra
vectorizer_project_grade_category.fit(project_data_X_train['project_grade_category'].
print(vectorizer_project_grade_category.get_feature_names())

project_grade_category_one_hot_train = vectorizer_project_grade_category.transform(pro
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_train.sh

project_grade_category_one_hot_test = vectorizer_project_grade_category.transform(proj
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_test.sha
```

```
['Grades PreK-2', 'Grades 3-5', 'Grades 6-8', 'Grades 9-12']
Shape of matrix after one hot encodig  (87398, 4)
Shape of matrix after one hot encodig  (21850, 4)
```

In [35]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_school_state = CountVectorizer(vocabulary=list(project_data_X_train['schoo
vectorizer_school_state.fit(project_data_X_train['school_state'].values)
```

```
        print(vectorizer_school_state.get_feature_names())


        school_state_one_hot_train = vectorizer_school_state.transform(project_data_X_train['s
        print("Shape of matrix after one hot encodig ",school_state_one_hot_train.shape)

        school_state_one_hot_test = vectorizer_school_state.transform(project_data_X_test['sch
        print("Shape of matrix after one hot encodig ",school_state_one_hot_test.shape)
```

```
['NY', 'MD', 'OK', 'MA', 'CA', 'AR', 'FL', 'PA', 'SC', 'NC', 'AZ', 'MI', 'AL', 'WI', 'NV', 'UT
Shape of matrix after one hot encodig  (87398, 51)
Shape of matrix after one hot encodig  (21850, 51)
```

2.2.2 Numerical features

```
In [36]: # check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
         # standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.p
         from sklearn.preprocessing import StandardScaler

         # price_standardized = standardScalar.fit(project_data['price'].values)
         # this will rise the error
         # ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ..
         # Reshape your data either using array.reshape(-1, 1)

         price_scalar = StandardScaler()
         price_scalar.fit(project_data_X_train['price'].values.reshape(-1,1)) # finding the me
         print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.va

         # Now standardize the data with above maen and variance.
         price_standardized_train = project_data_X_train['price'].values#price_scalar.transfor
         # Now standardize the data with above maen and variance.
         price_standardized_test = project_data_X_test['price'].values#price_scalar.transform(
```

```
Mean : 298.64356770177807, Standard deviation : 368.42853396795914
```

```
In [37]: # check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
         # standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.p
         from sklearn.preprocessing import StandardScaler,normalize

         # price_standardized = standardScalar.fit(project_data['price'].values)
         # this will rise the error
         # ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ..
         # Reshape your data either using array.reshape(-1, 1)

         price_scalar = StandardScaler()
         price_scalar.fit(project_data_X_train['teacher_number_of_previously_posted_projects']
         print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.va
```

```
        # Now standardize the data with above maen and variance.
        teacher_number_of_previously_posted_projects_standardized_train = project_data_X_trai

        # Now standardize the data with above maen and variance.
        teacher_number_of_previously_posted_projects_standardized_test = project_data_X_test[
```

```
Mean : 11.102897091466623, Standard deviation : 27.572082372998246
```

2.3 Make Data Model Ready: encoding eassay, and project_title

```
In [38]: vectorizer_essay_bow = CountVectorizer(min_df=10)
         vectorizer_essay_bow.fit(project_data_X_train.essay.values)

         text_bow_train=vectorizer_essay_bow.fit_transform(project_data_X_train.essay.values)
         print(text_bow_train.shape)

         text_bow_test=vectorizer_essay_bow.transform(project_data_X_test.essay.values)
         print(text_bow_test.shape)
```

```
(87398, 15254)
(21850, 15254)
```

```
In [39]: # Similarly you can vectorize for title also
         vectorizer_project_title_bow = CountVectorizer(min_df=10)
         vectorizer_project_title_bow.fit(project_data_X_train.project_title.values)

         title_text_bow_train=vectorizer_project_title_bow.fit_transform(project_data_X_train.p
         print(title_text_bow_train.shape)

         title_text_bow_test=vectorizer_project_title_bow.transform(project_data_X_test.projec
         print(title_text_bow_test.shape)
```

```
(87398, 2803)
(21850, 2803)
```

```
In [40]: from sklearn.feature_extraction.text import TfidfVectorizer
         vectorizer_essay_tfidf = TfidfVectorizer(min_df=10)
         vectorizer_essay_tfidf.fit(project_data_X_train.essay.values)

         text_tfidf_train=vectorizer_essay_tfidf.fit_transform(project_data_X_train.essay.valu
         print(text_tfidf_train.shape)

         text_tfidf_test=vectorizer_essay_tfidf.transform(project_data_X_test.essay.values)
         print(text_tfidf_test.shape)
```

```
(87398, 15254)
(21850, 15254)
```

In [41]:
```
# Similarly you can vectorize for title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_project_title_tfidf = TfidfVectorizer(min_df=10)
vectorizer_project_title_tfidf.fit(project_data_X_train.project_title.values)

title_text_tfidf_train=vectorizer_project_title_tfidf.fit_transform(project_data_X_tra
print(title_text_tfidf_train.shape)

title_text_tfidf_test=vectorizer_project_title_tfidf.transform(project_data_X_test.pro
print(title_text_tfidf_test.shape)
```

```
(87398, 2803)
(21850, 2803)
```

In [42]:
```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
# borrowed from https://therenegadecoder.com/code/how-to-check-if-a-file-exists-in-py
import os
exists = os.path.isfile('./glove_vectors')
if(not exists):
    model = loadGloveModel('glove.42B.300d.txt')

    '''# ============================
    Output:

    Loading Glove Model
    1917495it [06:32, 4879.69it/s]
    Done. 1917495  words loaded!

    # ============================'''

    words = []
    for i in preproced_texts:
```

```
            words.extend(i.split(' '))

        for i in preproced_titles:
            words.extend(i.split(' '))
        print("all the words in the coupus", len(words))
        words = set(words)
        print("the unique words in the coupus", len(words))

        inter_words = set(model.keys()).intersection(words)
        print("The number of words that are present in both glove vectors and our coupus"
              len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

        words_courpus = {}
        words_glove = set(model.keys())
        for i in words:
            if i in words_glove:
                words_courpus[i] = model[i]
        print("word 2 vec length", len(words_courpus))


        # stronging variables into pickle files python: http://www.jessicayung.com/how-to

        import pickle
        with open('glove_vectors', 'wb') as f:
            pickle.dump(words_courpus, f)
    else:
        print("glove already exists. No need to compute")
```

```
glove already exists. No need to compute
```

```
In [43]: with open('glove_vectors', 'rb') as f:
             model = pickle.load(f)
             glove_words =  set(model.keys())
```

```
In [44]: # average Word2Vec
         # compute average word2vec for each review.
         avg_w2v_vectors_essay_train = []; # the avg-w2v for each sentence/review is stored in
         for sentence in tqdm(project_data_X_train.essay.values): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             cnt_words =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:
                     vector += model[word]
                     cnt_words += 1
             if cnt_words != 0:
                 vector /= cnt_words
             avg_w2v_vectors_essay_train.append(vector)
```

```
        print(len(avg_w2v_vectors_essay_train))
        print(len(avg_w2v_vectors_essay_train[0]))
```

```
100%|| 87398/87398 [00:18<00:00, 4741.73it/s]


87398
300
```

```
In [45]: # average Word2Vec
         # compute average word2vec for each review.
         avg_w2v_vectors_essay_test = []; # the avg-w2v for each sentence/review is stored in
         for sentence in tqdm(project_data_X_test.essay.values): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             cnt_words =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:
                     vector += model[word]
                     cnt_words += 1
             if cnt_words != 0:
                 vector /= cnt_words
             avg_w2v_vectors_essay_test.append(vector)

         print(len(avg_w2v_vectors_essay_test))
         print(len(avg_w2v_vectors_essay_test[0]))
```

```
100%|| 21850/21850 [00:04<00:00, 4747.02it/s]


21850
300
```

```
In [46]: # average Word2Vec
         # compute average word2vec for each title.
         title_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in
         for sentence in tqdm(project_data_X_train.project_title.values): # for each review/se
             vector = np.zeros(300) # as word vectors are of zero length
             cnt_words =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:
                     vector += model[word]
                     cnt_words += 1
             if cnt_words != 0:
                 vector /= cnt_words
             title_avg_w2v_vectors_train.append(vector)
```

```
        print(len(title_avg_w2v_vectors_train))
        print(len(title_avg_w2v_vectors_train[0]))
```

```
100%|| 87398/87398 [00:00<00:00, 100948.72it/s]


87398
300
```

In [47]:
```
        # average Word2Vec
        # compute average word2vec for each title.
        title_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in
        for sentence in tqdm(project_data_X_test.project_title.values): # for each review/sen
            vector = np.zeros(300) # as word vectors are of zero length
            cnt_words =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:
                    vector += model[word]
                    cnt_words += 1
            if cnt_words != 0:
                vector /= cnt_words
            title_avg_w2v_vectors_test.append(vector)

        print(len(title_avg_w2v_vectors_test))
        print(len(title_avg_w2v_vectors_test[0]))
```

```
100%|| 21850/21850 [00:00<00:00, 107397.29it/s]


21850
300
```

In [48]:
```
        # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
        tfidf_model = TfidfVectorizer()
        tfidf_model.fit(project_data_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_)))
        essay_tfidf_words = set(tfidf_model.get_feature_names())
```

In [49]:
```
        # average Word2Vec
        # compute average word2vec for each review.
        essay_tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored
        for sentence in tqdm(project_data_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 essay_tfidf_words):
```

```
                    vec = model[word] # getting the vector for each word
                    # here we are multiplying idf value(dictionary[word]) and the tf value((s
                    tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
                    vector += (vec * tf_idf) # calculating tfidf weighted w2v
                    tf_idf_weight += tf_idf
            if tf_idf_weight != 0:
                vector /= tf_idf_weight
            essay_tfidf_w2v_vectors_train.append(vector)

        print(len(essay_tfidf_w2v_vectors_train))
        print(len(essay_tfidf_w2v_vectors_train[0]))
```

```
100%|| 87398/87398 [02:16<00:00, 639.48it/s]


87398
300
```

In [50]:
```
# average Word2Vec
# compute average word2vec for each review.
essay_tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored i
for sentence in tqdm(project_data_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 essay_tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((s
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors_test.append(vector)

print(len(essay_tfidf_w2v_vectors_test))
print(len(essay_tfidf_w2v_vectors_test[0]))
```

```
100%|| 21850/21850 [00:34<00:00, 641.95it/s]


21850
300
```

In [51]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_X_train.project_title.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_)))
        essay_tfidf_words = set(tfidf_model.get_feature_names())
```

In [52]:
```
        # average Word2Vec
        # compute average word2vec for each review.
        title_tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored
        for sentence in tqdm(project_data_X_train.project_title.values): # for each review/se
            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 essay_tfidf_words):
                    vec = model[word] # getting the vector for each word
                    # here we are multiplying idf value(dictionary[word]) and the tf value((s
                    tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
                    vector += (vec * tf_idf) # calculating tfidf weighted w2v
                    tf_idf_weight += tf_idf
            if tf_idf_weight != 0:
                vector /= tf_idf_weight
            title_tfidf_w2v_vectors_train.append(vector)

        print(len(title_tfidf_w2v_vectors_train))
        print(len(title_tfidf_w2v_vectors_train[0]))
```

```
100%|| 87398/87398 [00:01<00:00, 46535.63it/s]


87398
300
```

In [53]:
```
        # average Word2Vec
        # compute average word2vec for each review.
        title_tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored i
        for sentence in tqdm(project_data_X_test.project_title.values): # for each review/sen
            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 essay_tfidf_words):
                    vec = model[word] # getting the vector for each word
                    # here we are multiplying idf value(dictionary[word]) and the tf value((s
                    tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) #
                    vector += (vec * tf_idf) # calculating tfidf weighted w2v
                    tf_idf_weight += tf_idf
            if tf_idf_weight != 0:
                vector /= tf_idf_weight
            title_tfidf_w2v_vectors_test.append(vector)
```

```
        print(len(title_tfidf_w2v_vectors_test))
        print(len(title_tfidf_w2v_vectors_test[0]))
```

```
100%|| 21850/21850 [00:00<00:00, 45870.29it/s]


21850
300
```

2.4 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

2.0.1 2.4.1 Applying Decision Trees on BOW, SET 1

```
In [54]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
         from scipy.sparse import hstack
         # with the same hstack function we are concatinating a sparse matrix and a dense mati
         BOW = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_state_one
         print(BOW.shape)
         BOW_test= hstack((categories_one_hot_test, sub_categories_one_hot_test,school_state_o
         print(BOW_test.shape)
```

```
(87398, 18154)
(21850, 18154)
```

```
In [55]: feature_names_bow=vectorizer_clean_categories.get_feature_names()+vectorizer_clean_sub
         print(len(feature_names_bow))
```

```
18154
```

```
In [56]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import GridSearchCV
         model=DecisionTreeClassifier(class_weight='balanced')
         depth=[1,5,10,50,100,1000]
         split=[5,10,100,500]
         parameters = {'min_samples_split': split, 'max_depth':depth}
         clf = GridSearchCV(model, parameters,scoring='roc_auc',n_jobs=4,verbose=10)
         clf.fit(BOW,project_data_Y_train)
```

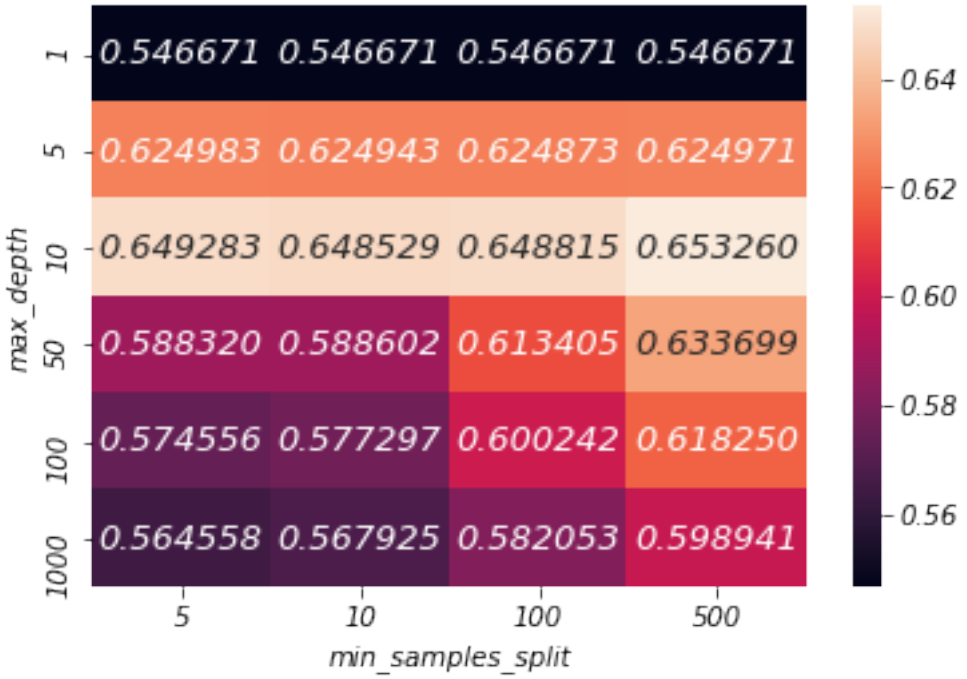
```
Fitting 3 folds for each of 24 candidates, totalling 72 fits


[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   5 tasks      | elapsed:    3.1s
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:    4.1s
```

```
[Parallel(n_jobs=4)]: Done  17 tasks      | elapsed:    8.1s
[Parallel(n_jobs=4)]: Done  24 tasks      | elapsed:   10.3s
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:   27.4s
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:  2.2min
[Parallel(n_jobs=4)]: Done  53 tasks      | elapsed:  5.2min
[Parallel(n_jobs=4)]: Done  64 tasks      | elapsed:  8.4min
[Parallel(n_jobs=4)]: Done  72 out of  72 | elapsed: 11.5min finished
```

Out[56]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight='balanced', 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=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=4,
       param_grid={'min_samples_split': [5, 10, 100, 500], 'max_depth': [1, 5, 10, 50
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [57]:
```
#https://stackoverflow.com/questions/30522724/take-multiple-lists-into-dataframe
#https://seaborn.pydata.org/generated/seaborn.heatmap.html
max_depth_all=[]
min_samples_split_all=[]
for i in range (0,len(clf.cv_results_['params'])):
    max_depth_all.append(clf.cv_results_['params'][i]['max_depth'])
    min_samples_split_all.append(clf.cv_results_['params'][i]['min_samples_split'])
#print(max_depth_all)
#print(min_samples_split_all)
score_all=clf.cv_results_['mean_test_score']
#print(score_all)
data=pd.DataFrame(
    {'max_depth': max_depth_all,
     'min_samples_split': min_samples_split_all,
     'auc': score_all
    })
data=data.pivot('max_depth','min_samples_split','auc')
sns.heatmap(data, annot=True,annot_kws={"size": 13}, fmt="f")
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27c801f6e48>
```



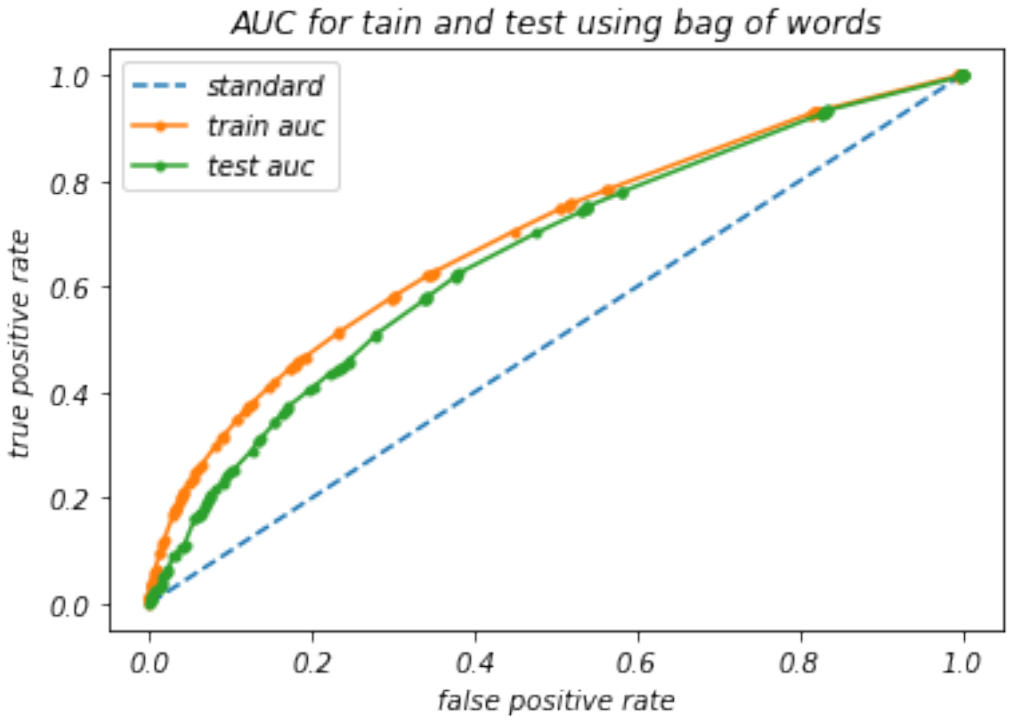
In [58]:
```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(class_weight='balanced',max_depth=10,min_samples_split=5
model.fit(BOW,project_data_Y_train)
```

Out[58]:
```
DecisionTreeClassifier(class_weight='balanced', criterion='gini',
            max_depth=10, max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=500,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [59]:
```
#https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classi
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from tqdm import tqdm

probs_test = model.predict_proba(BOW_test)
# keep probabilities for the positive outcome only
probs_test = probs_test[:, 1]
auc_test = roc_auc_score(project_data_Y_test, probs_test)
print('AUC: %.3f' % auc_test)
fpr, tpr, thresholds = roc_curve(project_data_Y_test, probs_test)

probs_train = model.predict_proba(BOW)
# keep probabilities for the positive outcome only
```

```
probs_train = probs_train[:, 1]
auc_train = roc_auc_score(project_data_Y_train, probs_train)
print('AUC: %.3f' % auc_train)
fpr1, tpr1, thresholds1 = roc_curve(project_data_Y_train, probs_train)


plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr1, tpr1, marker='.')
plt.plot(fpr, tpr, marker='.')

plt.legend({"standard":"","train auc":"","test auc":""})
plt.title("AUC for tain and test using bag of words")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.show()
```

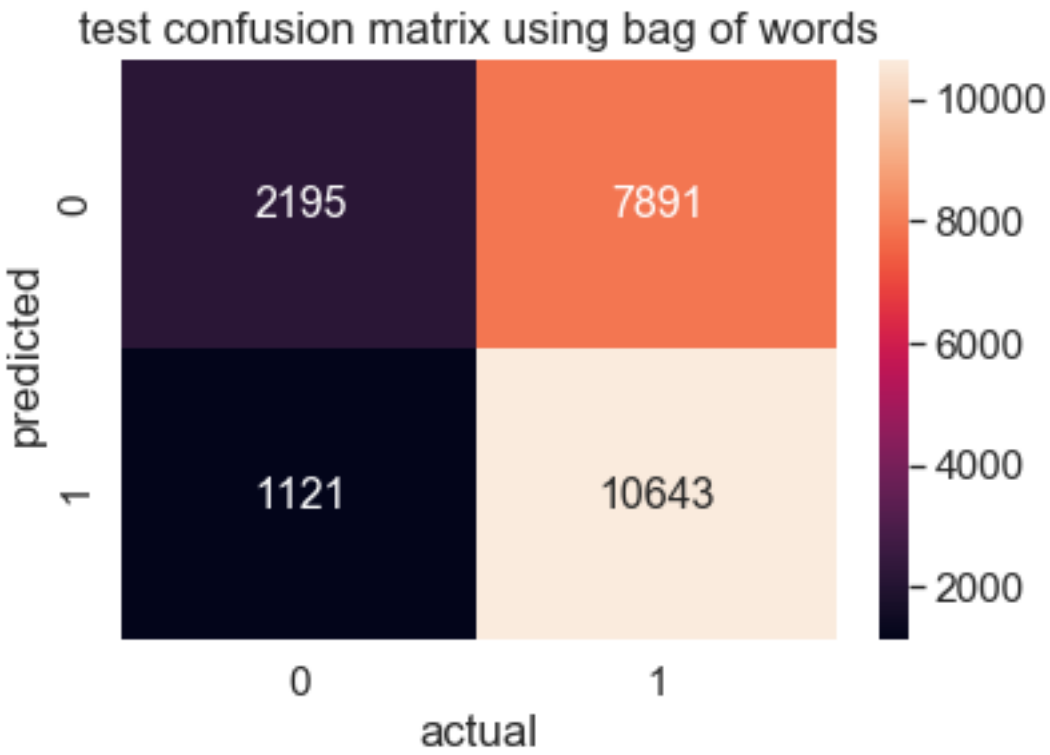
```
AUC: 0.658
AUC: 0.691
```



In [60]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#compute confudion matrix values and plot
from sklearn.metrics import confusion_matrix
predicted_bow_test=model.predict(BOW_test)
```

```
tn, fp, fn, tp = confusion_matrix(project_data_Y_test,predicted_bow_test).ravel()
print(tn, fp, fn, tp)
print("true positive rate",(tp/(tp+fn)))
print("true negaitive rate",(tn/(tn+fp)))
matrix=[[tn,fn],[fp,tp]]
print(matrix)
df_cm = pd.DataFrame(matrix, range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')# font size
plt.title("test confusion matrix using bag of words")
plt.xlabel("actual")
plt.ylabel("predicted")
plt.show()
```

```
2195 1121 7891 10643
true positive rate 0.5742419337433905
true negaitive rate 0.6619420989143546
[[2195, 7891], [1121, 10643]]
```



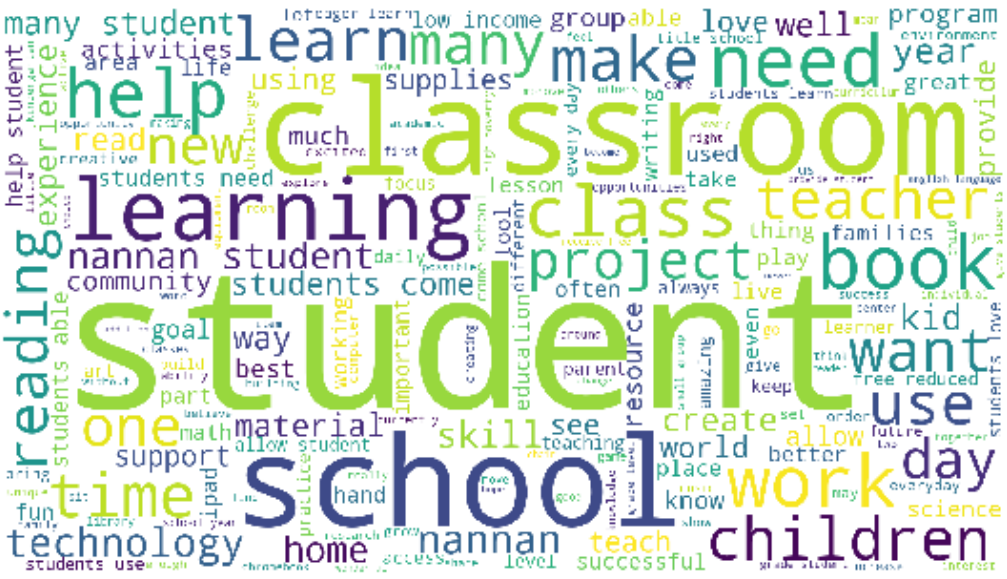
In [63]:
```
#collect ppredicted value and actual value from model and available labels respective
predicted_bow_test=model.predict(BOW_test)
```

```
        #create mask as your wish
        mask1 = predicted_bow_test > 0.5
        mask2 = (project_data_Y_test.values == 0)

        #combine the mask and select the index of mask
        special_mask=[]
        count=0
        for i in range(len(mask1)):
            #print(i)
            if(mask1[i] and mask2[i]):
                #print(mask1[i] and mask2[i])
                special_mask.append(i)
                count+=1

        #get the copy of data frame
        data_fp_whole_project_data = project_data_X_test
        print("Whole test data shape",data_fp_whole_project_data.shape)

        #reset the index
        a = np.arange(0,data_fp_whole_project_data.shape[0])
        data_fp_whole_project_data.index=a

        #apply same mask
        data_fp_whole_project_data=data_fp_whole_project_data.iloc[special_mask,]
        print("False positive from test data shape",data_fp_whole_project_data.shape)
```

```
Whole test data shape (21850, 23)
False positive from test data shape (1121, 23)
```

```
In [64]: from wordcloud import WordCloud, STOPWORDS
         import matplotlib.pyplot as plt
         import pandas as pd

         comment_words = ' '
         stopwords = set(STOPWORDS)

         # iterate through the csv file
         for val in data_fp_whole_project_data.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)):
```

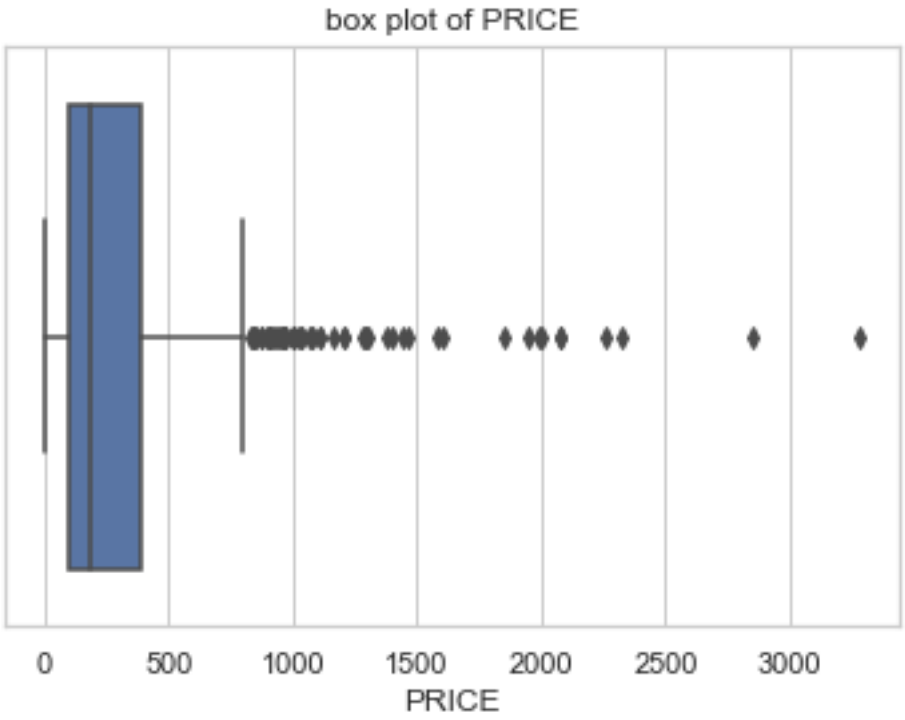
```
            tokens[i] = tokens[i].lower()

        for words in tokens:
            comment_words = comment_words + words + ' '


    wordcloud = WordCloud(width = 1920, height = 1080,
                    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()
```



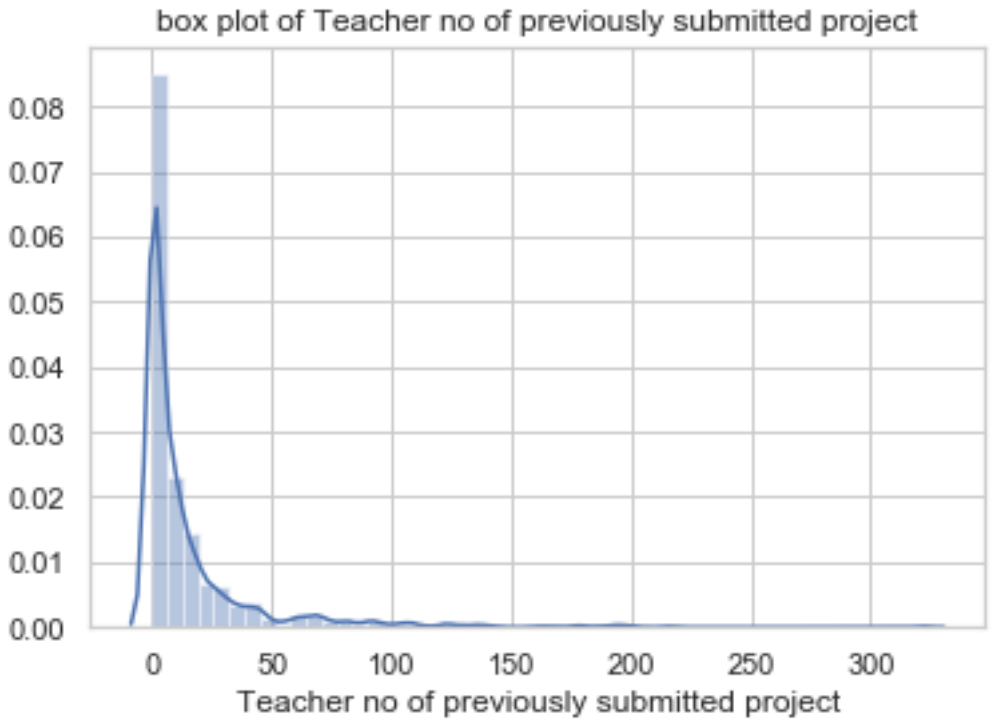
Observation: As Student, School, Classroom, Learning words are present, these points are being falsely predicted as positive.

```
In [65]: import seaborn as sns
         sns.set(style="whitegrid")
         tips = sns.load_dataset("tips")
         ax = sns.boxplot(x=data_fp_whole_project_data.price)
```

```
plt.xlabel("PRICE")
plt.title("box plot of PRICE")
plt.show()
```



In [66]:
```
import seaborn as sns
sns.set(style="whitegrid")
tips = sns.load_dataset("tips")
ax = sns.distplot(data_fp_whole_project_data.teacher_number_of_previously_posted_proj
plt.xlabel("Teacher no of previously submitted project")
plt.title("box plot of Teacher no of previously submitted project")
plt.show()
```



2.4.1.1 Graphviz visualization of Decision Tree on BOW, SET 1

```
In [67]: from sklearn.tree import DecisionTreeClassifier
         model=DecisionTreeClassifier(class_weight='balanced',max_depth=4,min_samples_split=500
         model.fit(BOW,project_data_Y_train)
```

```
Out[67]: DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=4,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=500,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')
```

```
In [68]: from sklearn.tree import export_graphviz
         export_graphviz(model,out_file='tree_bow.dot',feature_names=feature_names_bow)
```

2.0.2 2.4.2 Applying Decision Trees on TFIDF, SET 2

```
In [69]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
         from scipy.sparse import hstack
         # with the same hstack function we are concatinating a sparse matrix and a dense mati
         TFIDF = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_state_or
         print(TFIDF.shape)
         TFIDF_test = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_state
         print(TFIDF_test.shape)
```

```
(87398, 18154)
(21850, 18154)
```

```
In [70]: feature_names_bow=vectorizer_clean_categories.get_feature_names()+vectorizer_clean_sub
         print(len(feature_names_bow))
```

```
18154
```

```
In [71]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import GridSearchCV
         model=DecisionTreeClassifier(class_weight='balanced')
         depth=[1,5,10,50,100,1000]
         split=[5,10,100,500]
         parameters = {'min_samples_split': split, 'max_depth':depth}
         clf = GridSearchCV(model, parameters,scoring='roc_auc',n_jobs=4,verbose=10)
         clf.fit(TFIDF,project_data_Y_train)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   5 tasks      | elapsed:    4.1s
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:    5.6s
[Parallel(n_jobs=4)]: Done  17 tasks      | elapsed:   14.7s
[Parallel(n_jobs=4)]: Done  24 tasks      | elapsed:   19.1s
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:   49.9s
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:  2.9min
[Parallel(n_jobs=4)]: Done  53 tasks      | elapsed:  6.3min
[Parallel(n_jobs=4)]: Done  64 tasks      | elapsed:  9.3min
[Parallel(n_jobs=4)]: Done  72 out of  72 | elapsed: 11.7min finished
```

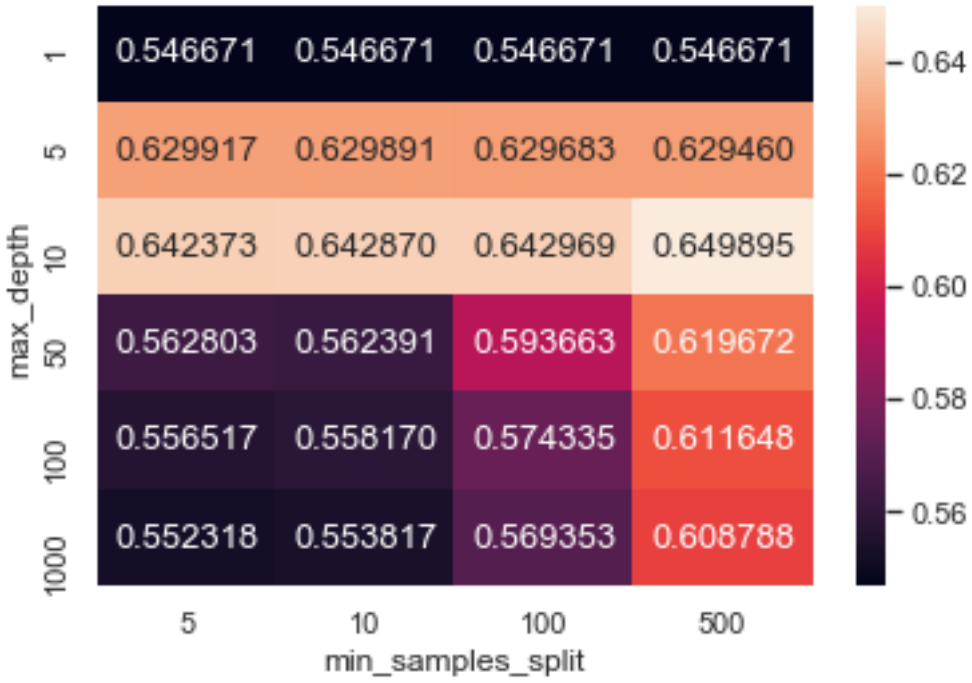
```
Out[71]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                estimator=DecisionTreeClassifier(class_weight='balanced', criterion='gini',
                     max_depth=None, max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best'),
                fit_params=None, iid='warn', n_jobs=4,
                param_grid={'min_samples_split': [5, 10, 100, 500], 'max_depth': [1, 5, 10, 50
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='roc_auc', verbose=10)
```

```
In [72]: clf.best_params_
```

```
Out[72]: {'max_depth': 10, 'min_samples_split': 500}
```

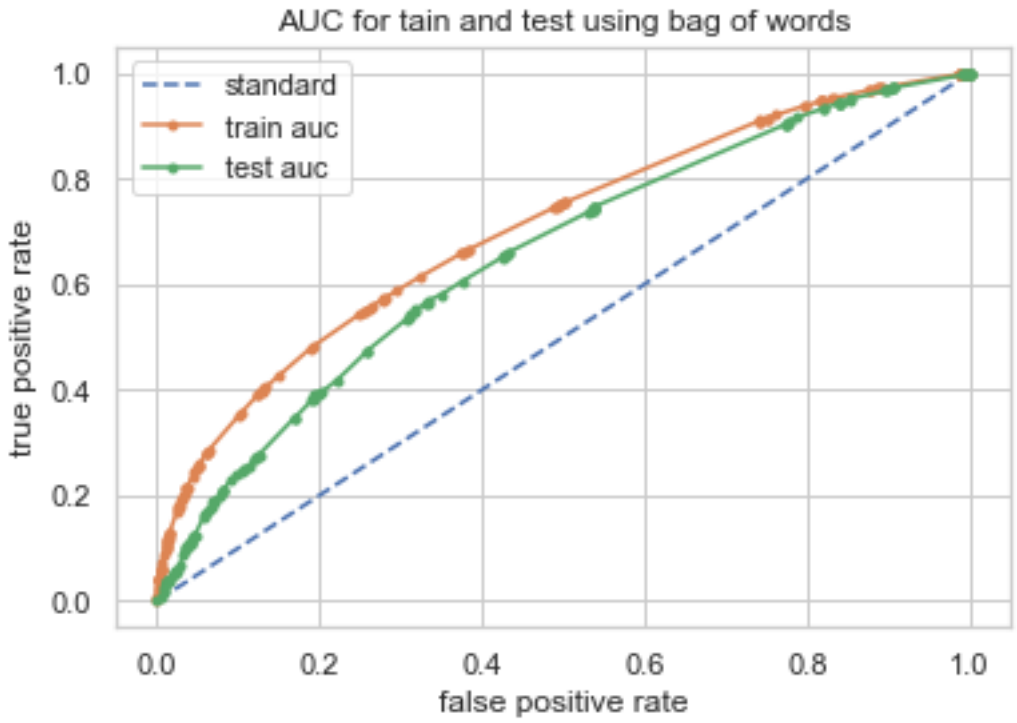
In [73]:
```
#https://stackoverflow.com/questions/30522724/take-multiple-lists-into-dataframe
#https://seaborn.pydata.org/generated/seaborn.heatmap.html
max_depth_all=[]
min_samples_split_all=[]
for i in range (0,len(clf.cv_results_['params'])):
    max_depth_all.append(clf.cv_results_['params'][i]['max_depth'])
    min_samples_split_all.append(clf.cv_results_['params'][i]['min_samples_split'])
#print(max_depth_all)
#print(min_samples_split_all)
score_all=clf.cv_results_['mean_test_score']
#print(score_all)
data=pd.DataFrame(
    {'max_depth': max_depth_all,
     'min_samples_split': min_samples_split_all,
     'auc': score_all
    })
data=data.pivot('max_depth','min_samples_split','auc')
sns.heatmap(data, annot=True,annot_kws={"size": 13}, fmt="f")
```

Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x27c82e91ba8>



In [74]:
```
model=DecisionTreeClassifier(class_weight='balanced',max_depth=10,min_samples_split=5
model.fit(TFIDF,project_data_Y_train)
```

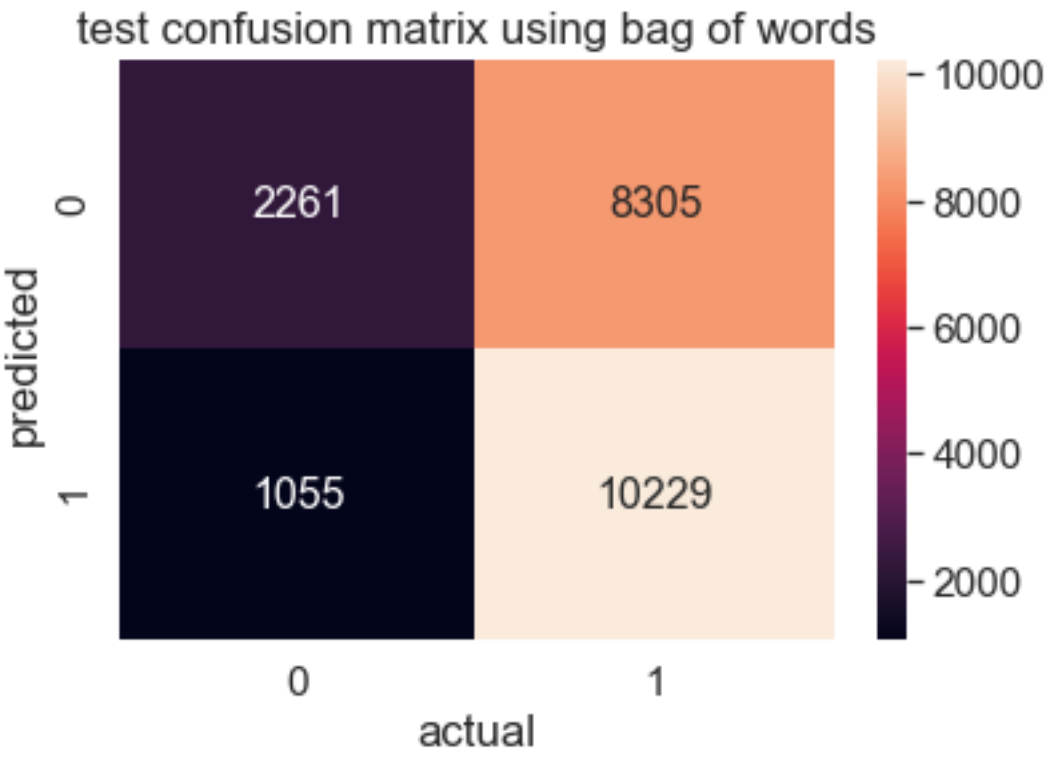
```
Out[74]: DecisionTreeClassifier(class_weight='balanced', criterion='gini',
                     max_depth=10, max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=500,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')

In [75]: #https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classi
         from sklearn.metrics import roc_curve
         from sklearn.metrics import roc_auc_score
         from tqdm import tqdm

         probs_test = model.predict_proba(TFIDF_test)
         # keep probabilities for the positive outcome only
         probs_test = probs_test[:, 1]
         auc_test = roc_auc_score(project_data_Y_test, probs_test)
         print('AUC: %.3f' % auc_test)
         fpr, tpr, thresholds = roc_curve(project_data_Y_test, probs_test)

         probs_train = model.predict_proba(TFIDF)
         # keep probabilities for the positive outcome only
         probs_train = probs_train[:, 1]
         auc_train = roc_auc_score(project_data_Y_train, probs_train)
         print('AUC: %.3f' % auc_train)
         fpr1, tpr1, thresholds1 = roc_curve(project_data_Y_train, probs_train)


         plt.plot([0, 1], [0, 1], linestyle='--')
         plt.plot(fpr1, tpr1, marker='.')
         plt.plot(fpr, tpr, marker='.')

         plt.legend({"standard":"","train auc":"","test auc":""})
         plt.title("AUC for tain and test using bag of words")
         plt.xlabel("false positive rate")
         plt.ylabel("true positive rate")
         plt.show()

AUC: 0.655
AUC: 0.705
```



In [76]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#compute confudion matrix values and plot
from sklearn.metrics import confusion_matrix
predicted_bow_test=model.predict(TFIDF_test)
tn, fp, fn, tp = confusion_matrix(project_data_Y_test,predicted_bow_test).ravel()
print(tn, fp, fn, tp)
print("true positive rate",(tp/(tp+fn)))
print("true negaitive rate",(tn/(tn+fp)))
matrix=[[tn,fn],[fp,tp]]
print(matrix)
df_cm = pd.DataFrame(matrix, range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')# font size
plt.title("test confusion matrix using bag of words")
plt.xlabel("actual")
plt.ylabel("predicted")
plt.show()
```

```
2261 1055 8305 10229
true positive rate 0.5519046077479227
true negaitive rate 0.6818455971049457
```

```
[[2261, 8305], [1055, 10229]]
```

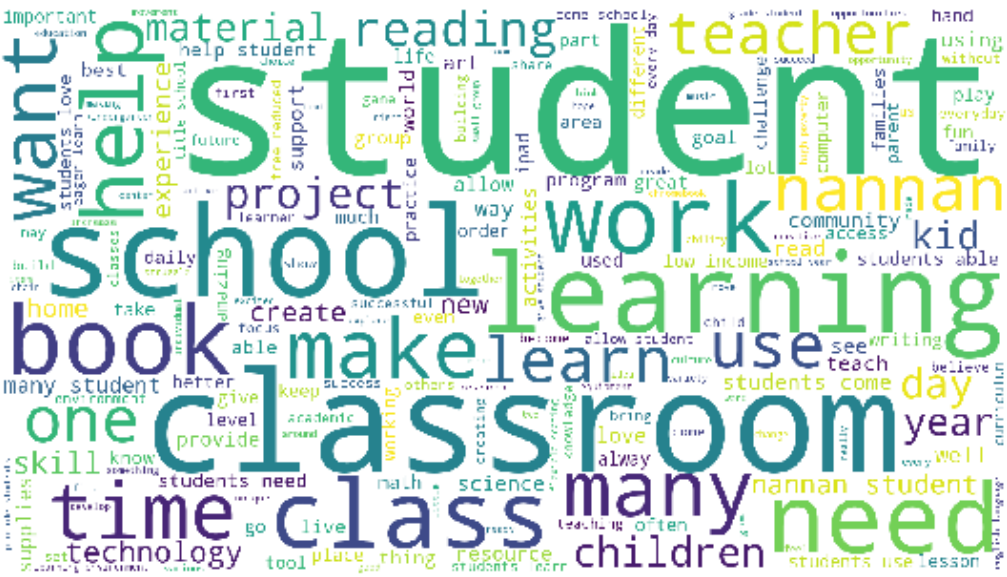


In [77]:
```
#collect ppredicted value and actual value from model and available labels respective
predicted_bow_test=model.predict(TFIDF_test)

#create mask as your wish
mask1 = predicted_bow_test > 0.5
mask2 = (project_data_Y_test.values == 0)

#combine the mask and select the index of mask
special_mask=[]
count=0
for i in range(len(mask1)):
    #print(i)
    if(mask1[i] and mask2[i]):
        #print(mask1[i] and mask2[i])
        special_mask.append(i)
        count+=1

#get the copy of data frame
data_fp_whole_project_data = project_data_X_test
print("Whole test data shape",data_fp_whole_project_data.shape)
```

```
        #reset the index
        a = np.arange(0,data_fp_whole_project_data.shape[0])
        data_fp_whole_project_data.index=a

        #apply same mask
        data_fp_whole_project_data=data_fp_whole_project_data.iloc[special_mask,]
        print("False positive from test data shape",data_fp_whole_project_data.shape)
```

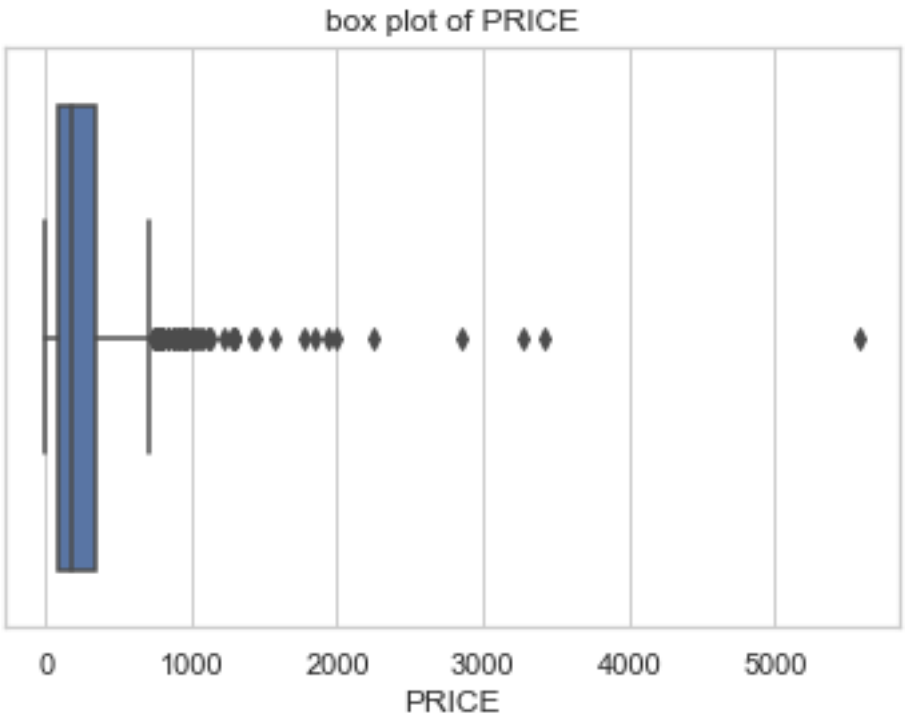
```
Whole test data shape (21850, 23)
False positive from test data shape (1055, 23)
```

```
In [78]: from wordcloud import WordCloud, STOPWORDS
         import matplotlib.pyplot as plt
         import pandas as pd

         comment_words = ' '
         stopwords = set(STOPWORDS)

         # iterate through the csv file
         for val in data_fp_whole_project_data.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()

             for words in tokens:
                 comment_words = comment_words + words + ' '


         wordcloud = WordCloud(width = 1920, height = 1080,
                         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()
```

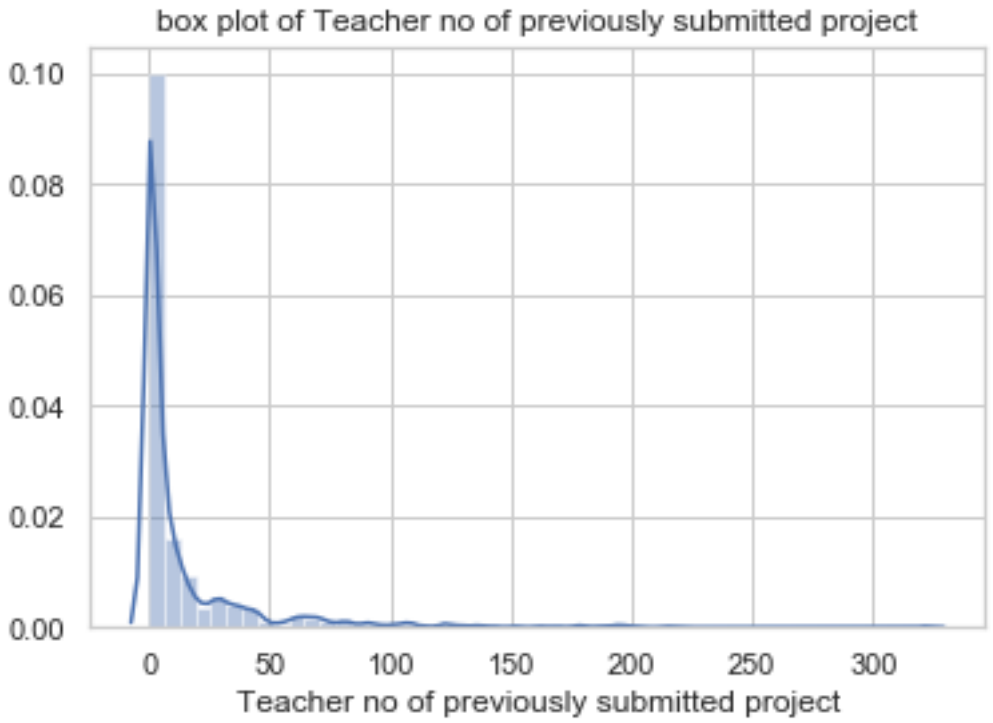


Observation: As Student, School, Classroom, Learning words are present, these points are being falsely predicted as positive.

```
In [79]: import seaborn as sns
         sns.set(style="whitegrid")
         tips = sns.load_dataset("tips")
         ax = sns.boxplot(x=data_fp_whole_project_data.price)
         plt.xlabel("PRICE")
         plt.title("box plot of PRICE")
         plt.show()
```




```
In [80]: import seaborn as sns
         sns.set(style="whitegrid")
         tips = sns.load_dataset("tips")
         ax = sns.distplot(data_fp_whole_project_data.teacher_number_of_previously_posted_proje
         plt.xlabel("Teacher no of previously submitted project")
         plt.title("box plot of Teacher no of previously submitted project")
         plt.show()
```



2.4.2.1 Graphviz visualization of Decision Tree on TFIDF, SET 2

```
In [81]: from sklearn.tree import DecisionTreeClassifier
         model=DecisionTreeClassifier(class_weight='balanced',max_depth=4,min_samples_split=500
         model.fit(TFIDF,project_data_Y_train)
```

```
Out[81]: DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=4,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=500,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')
```

```
In [82]: from sklearn.tree import export_graphviz
         export_graphviz(model,out_file='tree_tfidf.dot',feature_names=feature_names_bow)
```

2.0.3 2.4.3 Applying Decision Trees on AVG W2V, SET 3

```
In [83]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
         from scipy.sparse import hstack
         # with the same hstack function we are concatinating a sparse matrix and a dense mati
         AVG_W2V = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_state_
         print(AVG_W2V.shape)
         AVG_W2V_test = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_sta
         print(AVG_W2V_test.shape)
```

```
(87398, 697)
(21850, 697)
```

```
In [84]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import GridSearchCV
         model=DecisionTreeClassifier(class_weight='balanced')
         depth=[1,5,10,50,100,1000]
         split=[5,10,100,500]
         parameters = {'min_samples_split': split, 'max_depth':depth}
         clf = GridSearchCV(model, parameters,scoring='roc_auc',n_jobs=4,verbose=10)
         clf.fit(AVG_W2V,project_data_Y_train)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   5 tasks      | elapsed:   15.7s
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:   23.7s
[Parallel(n_jobs=4)]: Done  17 tasks      | elapsed:  1.2min
[Parallel(n_jobs=4)]: Done  24 tasks      | elapsed:  1.6min
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:  4.8min
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed: 11.4min
[Parallel(n_jobs=4)]: Done  53 tasks      | elapsed: 19.9min
[Parallel(n_jobs=4)]: Done  64 tasks      | elapsed: 26.6min
[Parallel(n_jobs=4)]: Done  72 out of  72 | elapsed: 31.7min finished
```

```
Out[84]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                estimator=DecisionTreeClassifier(class_weight='balanced', 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=None,
                     splitter='best'),
                fit_params=None, iid='warn', n_jobs=4,
                param_grid={'min_samples_split': [5, 10, 100, 500], 'max_depth': [1, 5, 10, 50
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='roc_auc', verbose=10)
```

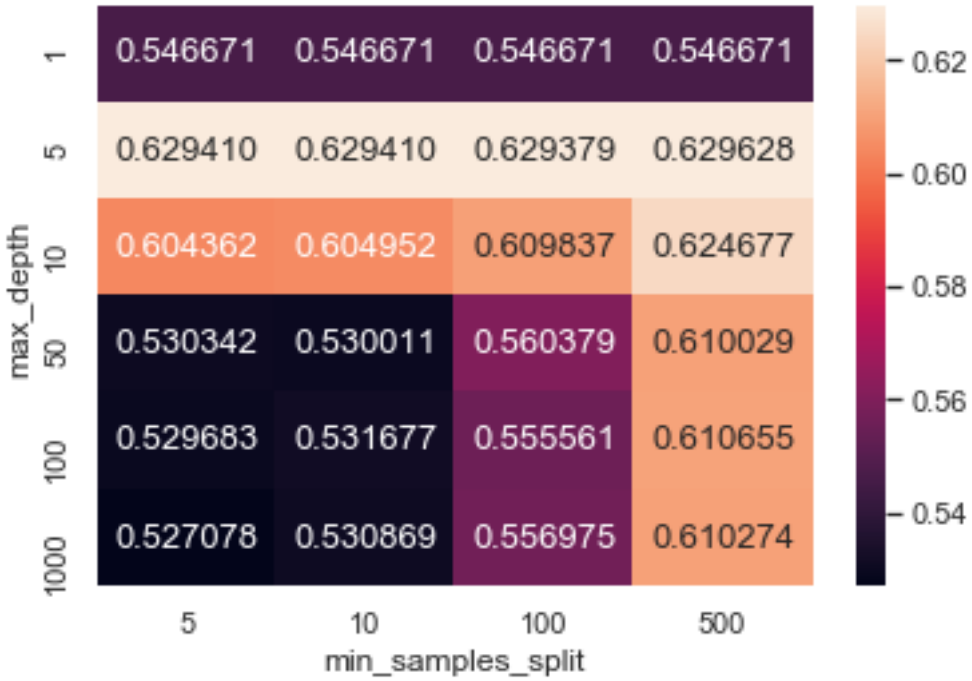
```
In [85]: clf.best_params_
```

```
Out[85]: {'max_depth': 5, 'min_samples_split': 500}
```

```
In [86]: #https://stackoverflow.com/questions/30522724/take-multiple-lists-into-dataframe
         #https://seaborn.pydata.org/generated/seaborn.heatmap.html
         max_depth_all=[]
         min_samples_split_all=[]
         for i in range (0,len(clf.cv_results_['params'])):
```

```
            max_depth_all.append(clf.cv_results_['params'][i]['max_depth'])
            min_samples_split_all.append(clf.cv_results_['params'][i]['min_samples_split'])
        #print(max_depth_all)
        #print(min_samples_split_all)
        score_all=clf.cv_results_['mean_test_score']
        #print(score_all)
        data=pd.DataFrame(
            {'max_depth': max_depth_all,
             'min_samples_split': min_samples_split_all,
             'auc': score_all
            })
        data=data.pivot('max_depth','min_samples_split','auc')
        sns.heatmap(data, annot=True,annot_kws={"size": 13}, fmt="f")
```

Out[86]: <matplotlib.axes._subplots.AxesSubplot at 0x27c8300af28>

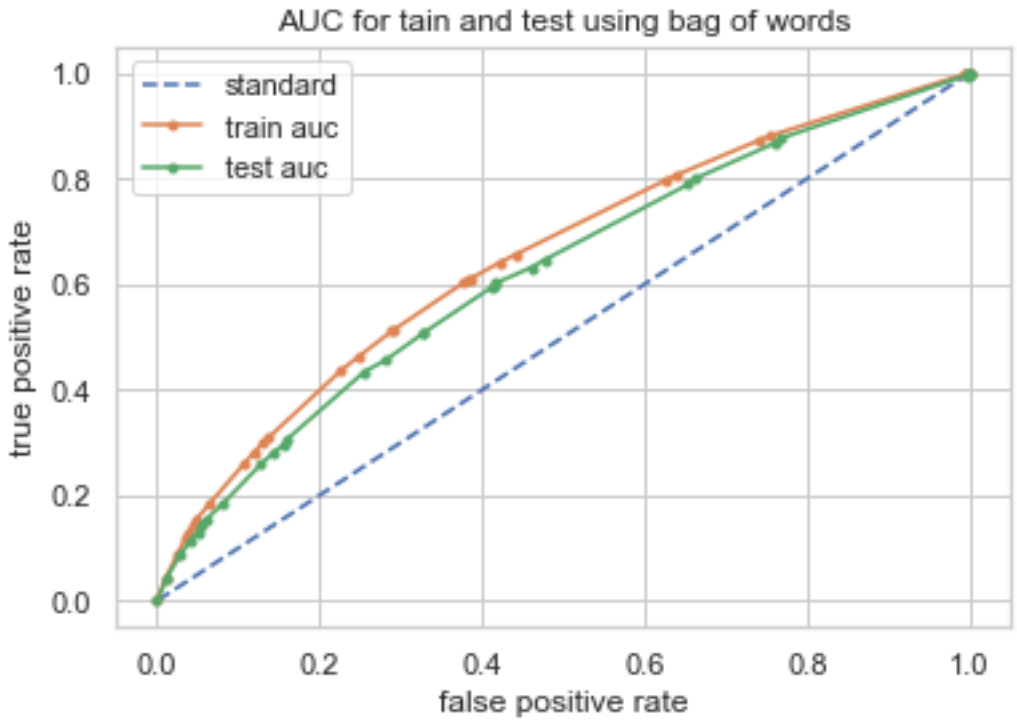



```
In [87]: model=DecisionTreeClassifier(class_weight='balanced',max_depth=5,min_samples_split=500
         model.fit(AVG_W2V,project_data_Y_train)
```

Out[87]:
```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=5,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                       splitter='best')
```

In [88]:
```
#https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classi
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from tqdm import tqdm

probs_test = model.predict_proba(AVG_W2V_test)
# keep probabilities for the positive outcome only
probs_test = probs_test[:, 1]
auc_test = roc_auc_score(project_data_Y_test, probs_test)
print('AUC: %.3f' % auc_test)
fpr, tpr, thresholds = roc_curve(project_data_Y_test, probs_test)

probs_train = model.predict_proba(AVG_W2V)
# keep probabilities for the positive outcome only
probs_train = probs_train[:, 1]
auc_train = roc_auc_score(project_data_Y_train, probs_train)
print('AUC: %.3f' % auc_train)
fpr1, tpr1, thresholds1 = roc_curve(project_data_Y_train, probs_train)


plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr1, tpr1, marker='.')
plt.plot(fpr, tpr, marker='.')

plt.legend({"standard":"","train auc":"","test auc":""})
plt.title("AUC for tain and test using bag of words")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.show()
```

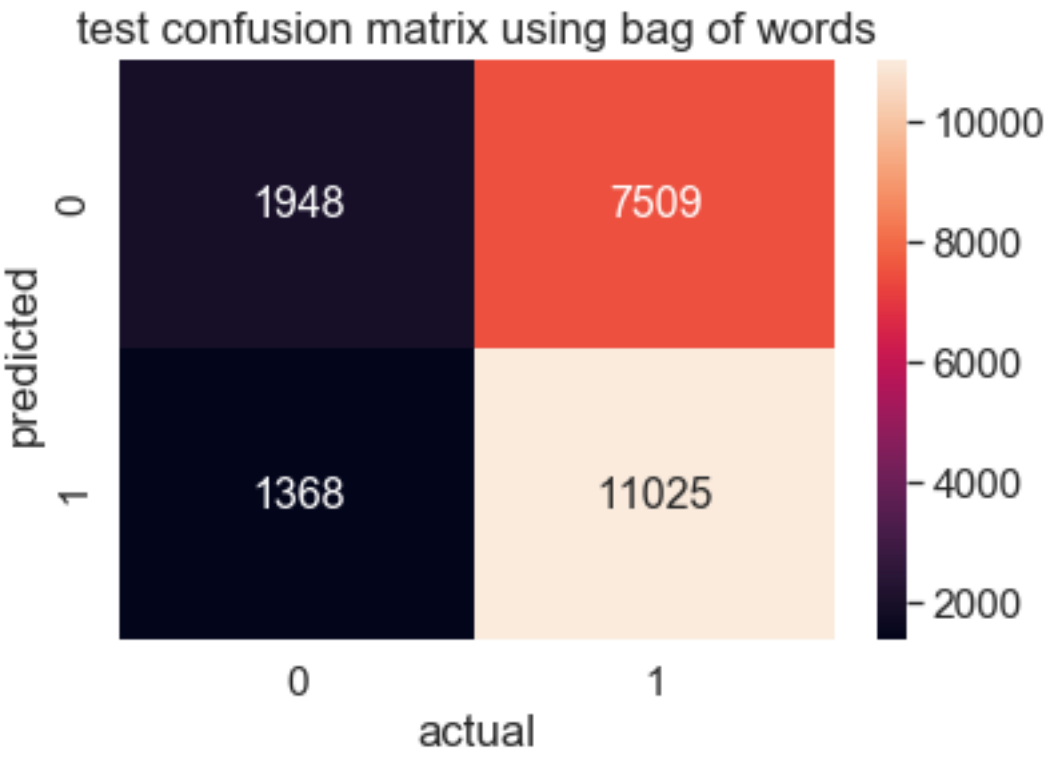
```
AUC: 0.624
AUC: 0.650
```



In [89]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#compute confudion matrix values and plot
from sklearn.metrics import confusion_matrix
predicted_bow_test=model.predict(AVG_W2V_test)
tn, fp, fn, tp = confusion_matrix(project_data_Y_test,predicted_bow_test).ravel()
print(tn, fp, fn, tp)
print("true positive rate",(tp/(tp+fn)))
print("true negaitive rate",(tn/(tn+fp)))
matrix=[[tn,fn],[fp,tp]]
print(matrix)
df_cm = pd.DataFrame(matrix, range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')# font size
plt.title("test confusion matrix using bag of words")
plt.xlabel("actual")
plt.ylabel("predicted")
plt.show()
```

```
1948 1368 7509 11025
true positive rate 0.5948527031401748
true negaitive rate 0.5874547647768396
```

```
[[1948, 7509], [1368, 11025]]
```

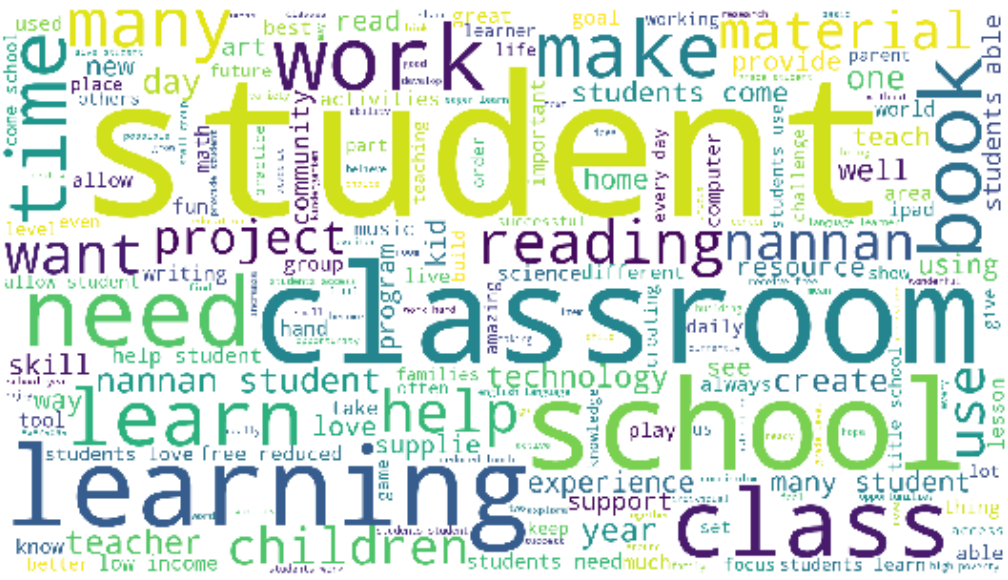


In [90]:
```
#collect ppredicted value and actual value from model and available labels respective
predicted_bow_test=model.predict(AVG_W2V_test)

#create mask as your wish
mask1 = predicted_bow_test > 0.5
mask2 = (project_data_Y_test.values == 0)

#combine the mask and select the index of mask
special_mask=[]
count=0
for i in range(len(mask1)):
    #print(i)
    if(mask1[i] and mask2[i]):
        #print(mask1[i] and mask2[i])
        special_mask.append(i)
        count+=1

#get the copy of data frame
data_fp_whole_project_data = project_data_X_test
print("Whole test data shape",data_fp_whole_project_data.shape)
```

```
        #reset the index
        a = np.arange(0,data_fp_whole_project_data.shape[0])
        data_fp_whole_project_data.index=a

        #apply same mask
        data_fp_whole_project_data=data_fp_whole_project_data.iloc[special_mask,]
        print("False positive from test data shape",data_fp_whole_project_data.shape)
```

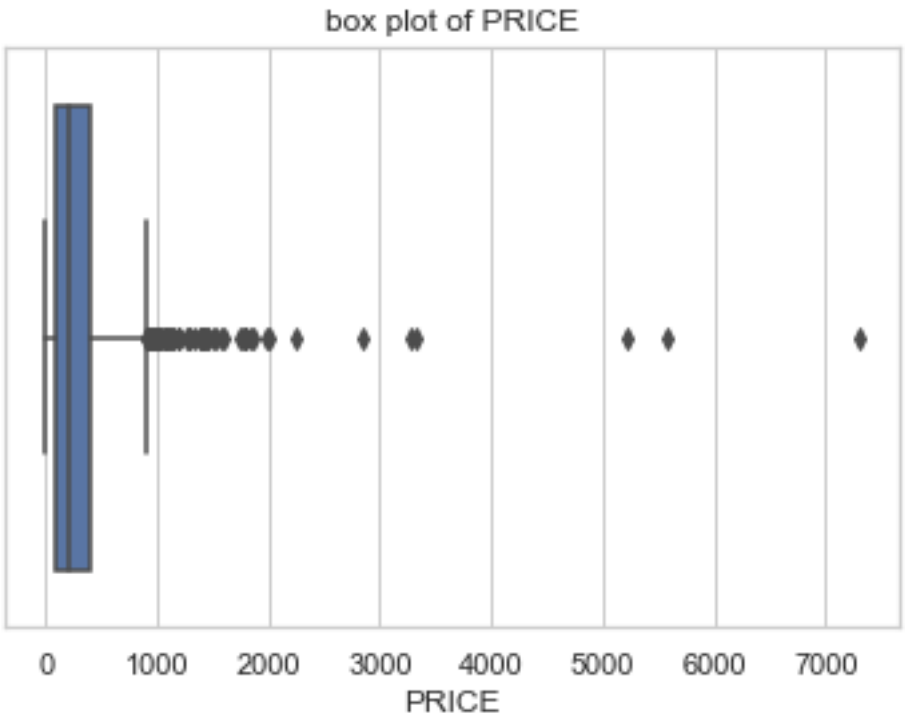
```
Whole test data shape (21850, 23)
False positive from test data shape (1368, 23)
```

```
In [91]: from wordcloud import WordCloud, STOPWORDS
         import matplotlib.pyplot as plt
         import pandas as pd

         comment_words = ' '
         stopwords = set(STOPWORDS)

         # iterate through the csv file
         for val in data_fp_whole_project_data.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()

             for words in tokens:
                 comment_words = comment_words + words + ' '


         wordcloud = WordCloud(width = 1920, height = 1080,
                         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()
```

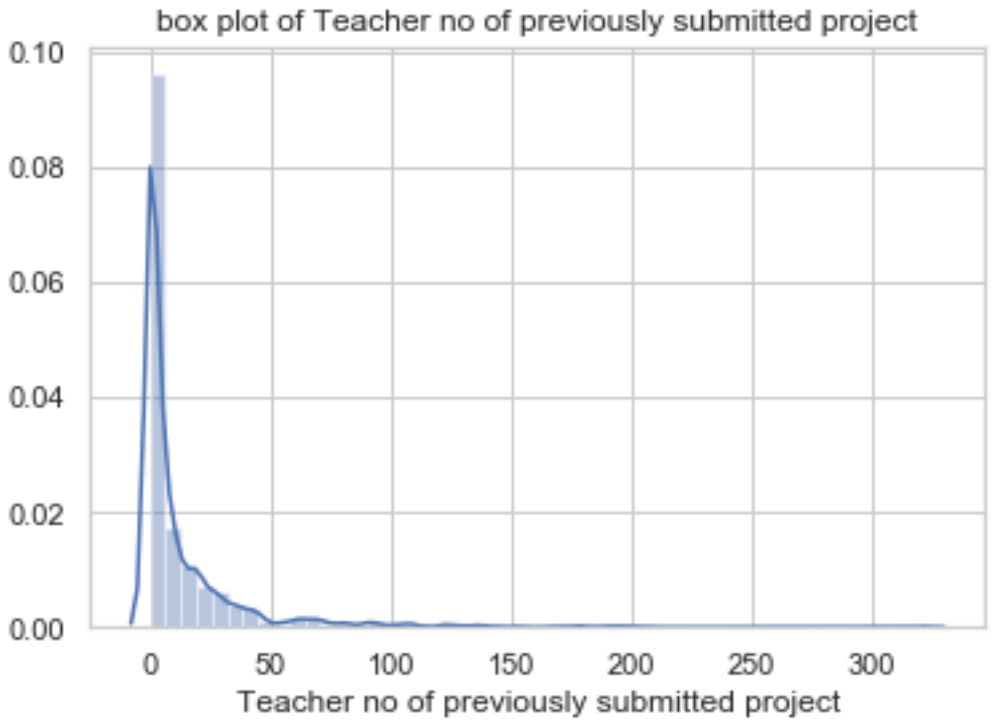


Observation: As Student, School, Classroom, Learning words are present, these points are being falsely predicted as positive.

```
In [92]: import seaborn as sns
         sns.set(style="whitegrid")
         tips = sns.load_dataset("tips")
         ax = sns.boxplot(x=data_fp_whole_project_data.price)
         plt.xlabel("PRICE")
         plt.title("box plot of PRICE")
         plt.show()
```




```
In [93]: import seaborn as sns
         sns.set(style="whitegrid")
         tips = sns.load_dataset("tips")
         ax = sns.distplot(data_fp_whole_project_data.teacher_number_of_previously_posted_proje
         plt.xlabel("Teacher no of previously submitted project")
         plt.title("box plot of Teacher no of previously submitted project")
         plt.show()
```



2.0.4 2.4.4 Applying Decision Trees on TFIDF W2V, SET 4

In [94]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense mati
TFIDF_W2V = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_sta
print(TFIDF_W2V.shape)
TFIDF_W2V_test = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_
print(TFIDF_W2V_test.shape)
```

```
(87398, 697)
(21850, 697)
```

In [95]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
model=DecisionTreeClassifier(class_weight='balanced')
depth=[1,5,10,50,100,1000]
split=[5,10,100,500]
parameters = {'min_samples_split': split, 'max_depth':depth}
clf = GridSearchCV(model, parameters,scoring='roc_auc',n_jobs=4,verbose=10)
clf.fit(TFIDF_W2V,project_data_Y_train)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   5 tasks      | elapsed:   15.6s
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:   23.4s
[Parallel(n_jobs=4)]: Done  17 tasks      | elapsed:  1.2min
[Parallel(n_jobs=4)]: Done  24 tasks      | elapsed:  1.6min
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:  4.8min
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed: 11.5min
[Parallel(n_jobs=4)]: Done  53 tasks      | elapsed: 20.3min
[Parallel(n_jobs=4)]: Done  64 tasks      | elapsed: 27.2min
[Parallel(n_jobs=4)]: Done  72 out of  72 | elapsed: 32.3min finished
```

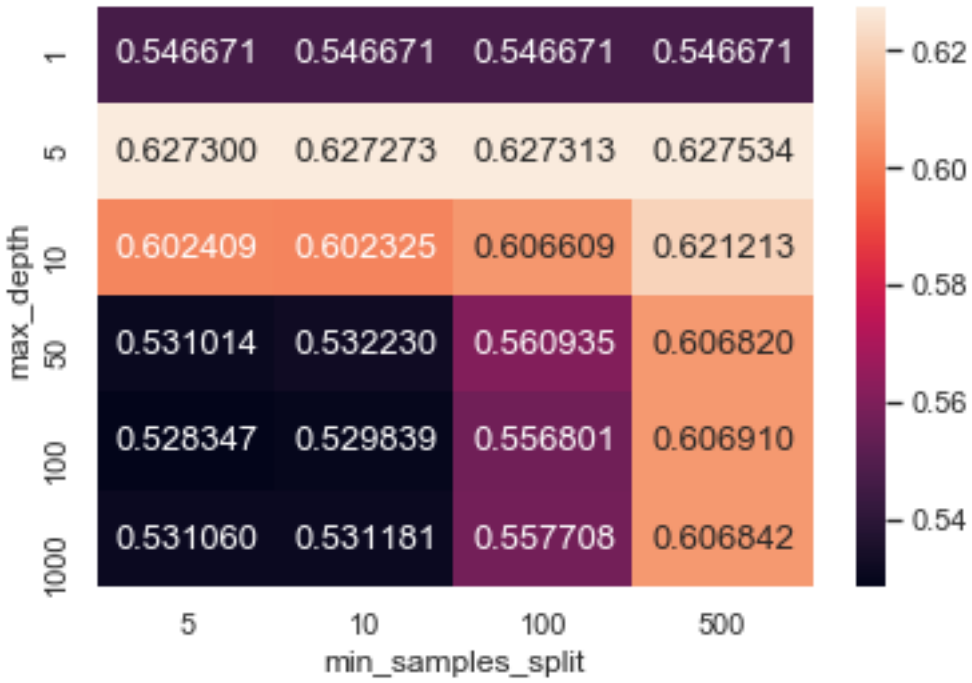
Out[95]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight='balanced', criterion='gini',
            max_depth=None, max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=4,
       param_grid={'min_samples_split': [5, 10, 100, 500], 'max_depth': [1, 5, 10, 50
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [96]:
```
clf.best_params_
```

Out[96]:
```
{'max_depth': 5, 'min_samples_split': 500}
```

In [97]:
```
#https://stackoverflow.com/questions/30522724/take-multiple-lists-into-dataframe
#https://seaborn.pydata.org/generated/seaborn.heatmap.html
max_depth_all=[]
min_samples_split_all=[]
for i in range (0,len(clf.cv_results_['params'])):
    max_depth_all.append(clf.cv_results_['params'][i]['max_depth'])
    min_samples_split_all.append(clf.cv_results_['params'][i]['min_samples_split'])
#print(max_depth_all)
#print(min_samples_split_all)
score_all=clf.cv_results_['mean_test_score']
#print(score_all)
data=pd.DataFrame(
    {'max_depth': max_depth_all,
     'min_samples_split': min_samples_split_all,
     'auc': score_all
    })
data=data.pivot('max_depth','min_samples_split','auc')
sns.heatmap(data, annot=True,annot_kws={"size": 13}, fmt="f")
```

Out[97]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27c825ea208>
```



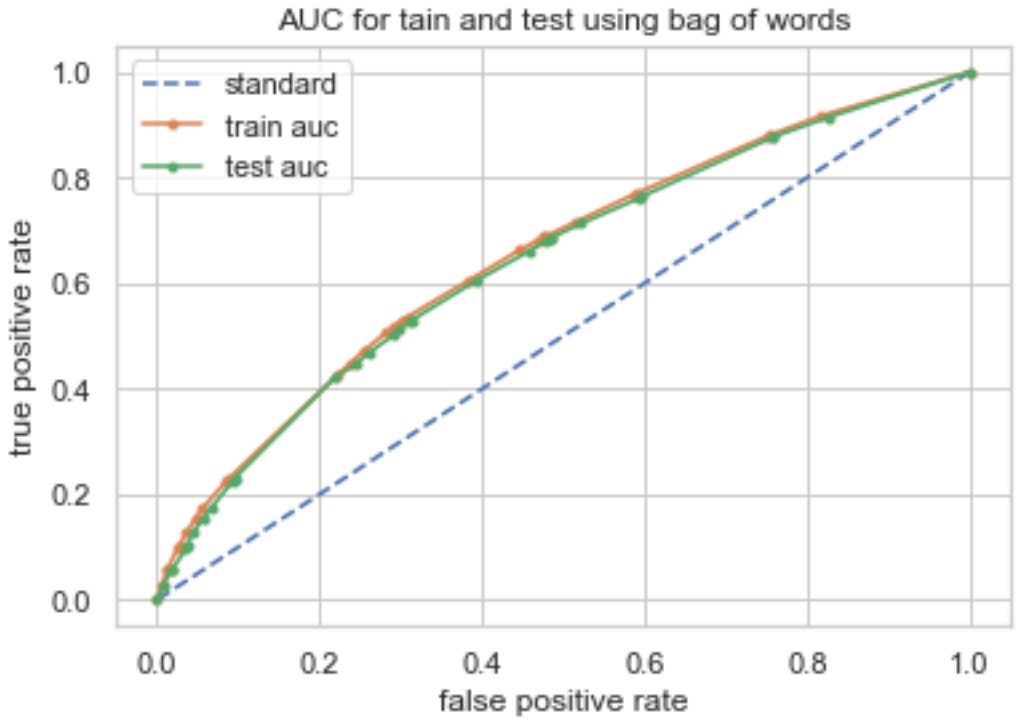

```
In [98]: model=DecisionTreeClassifier(class_weight='balanced',max_depth=5,min_samples_split=500
         model.fit(TFIDF_W2V,project_data_Y_train)
```

```
Out[98]: DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=5,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=500,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')
```

```
In [99]: #https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classi
         from sklearn.metrics import roc_curve
         from sklearn.metrics import roc_auc_score
         from tqdm import tqdm

         probs_test = model.predict_proba(TFIDF_W2V_test)
         # keep probabilities for the positive outcome only
         probs_test = probs_test[:, 1]
         auc_test = roc_auc_score(project_data_Y_test, probs_test)
         print('AUC: %.3f' % auc_test)
         fpr, tpr, thresholds = roc_curve(project_data_Y_test, probs_test)

         probs_train = model.predict_proba(TFIDF_W2V)
         # keep probabilities for the positive outcome only
```

```
probs_train = probs_train[:, 1]
auc_train = roc_auc_score(project_data_Y_train, probs_train)
print('AUC: %.3f' % auc_train)
fpr1, tpr1, thresholds1 = roc_curve(project_data_Y_train, probs_train)


plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr1, tpr1, marker='.')
plt.plot(fpr, tpr, marker='.')

plt.legend({"standard":"","train auc":"","test auc":""})
plt.title("AUC for tain and test using bag of words")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.show()
```

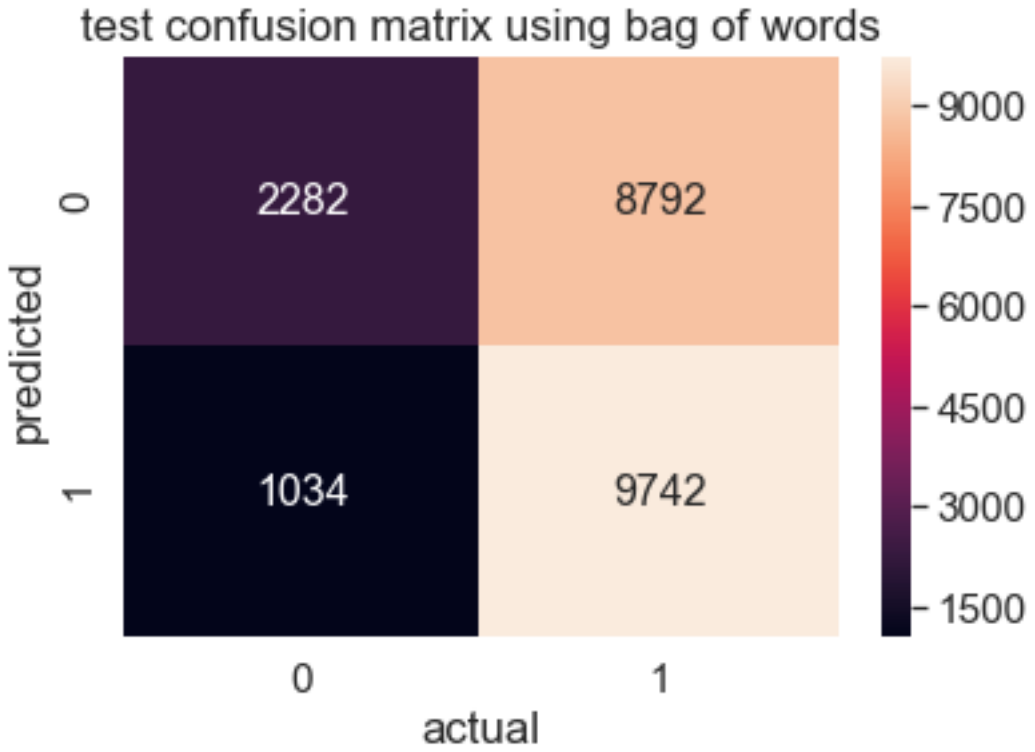
```
AUC: 0.642
AUC: 0.651
```




```
In [100]: #https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
          #compute confudion matrix values and plot
          from sklearn.metrics import confusion_matrix
          predicted_bow_test=model.predict(TFIDF_W2V_test)
```

```
tn, fp, fn, tp = confusion_matrix(project_data_Y_test,predicted_bow_test).ravel()
print(tn, fp, fn, tp)
print("true positive rate",(tp/(tp+fn)))
print("true negaitive rate",(tn/(tn+fp)))
matrix=[[tn,fn],[fp,tp]]
print(matrix)
df_cm = pd.DataFrame(matrix, range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')# font size
plt.title("test confusion matrix using bag of words")
plt.xlabel("actual")
plt.ylabel("predicted")
plt.show()
```

```
2282 1034 8792 9742
true positive rate 0.5256285745117082
true negaitive rate 0.6881785283474066
[[2282, 8792], [1034, 9742]]
```



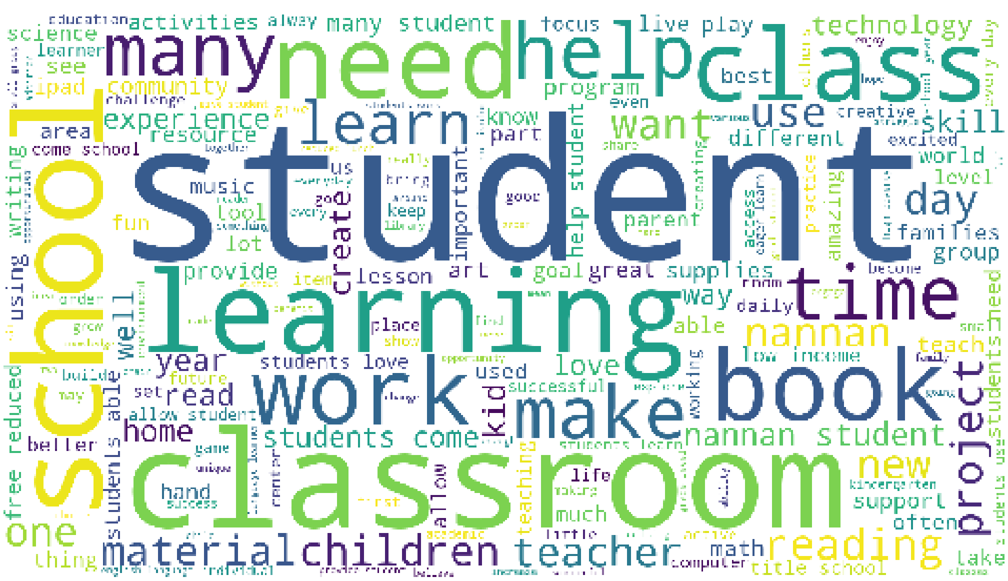
In [101]:
```
#collect ppredicted value and actual value from model and available labels respectiv
predicted_bow_test=model.predict(TFIDF_W2V_test)
```

```
        #create mask as your wish
        mask1 = predicted_bow_test > 0.5
        mask2 = (project_data_Y_test.values == 0)

        #combine the mask and select the index of mask
        special_mask=[]
        count=0
        for i in range(len(mask1)):
            #print(i)
            if(mask1[i] and mask2[i]):
                #print(mask1[i] and mask2[i])
                special_mask.append(i)
                count+=1

        #get the copy of data frame
        data_fp_whole_project_data = project_data_X_test
        print("Whole test data shape",data_fp_whole_project_data.shape)

        #reset the index
        a = np.arange(0,data_fp_whole_project_data.shape[0])
        data_fp_whole_project_data.index=a

        #apply same mask
        data_fp_whole_project_data=data_fp_whole_project_data.iloc[special_mask,]
        print("False positive from test data shape",data_fp_whole_project_data.shape)
```

```
Whole test data shape (21850, 23)
False positive from test data shape (1034, 23)
```

```
In [102]: from wordcloud import WordCloud, STOPWORDS
          import matplotlib.pyplot as plt
          import pandas as pd

          comment_words = ' '
          stopwords = set(STOPWORDS)

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

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

              # split the value
              tokens = val.split()

              # Converts each token into lowercase
```

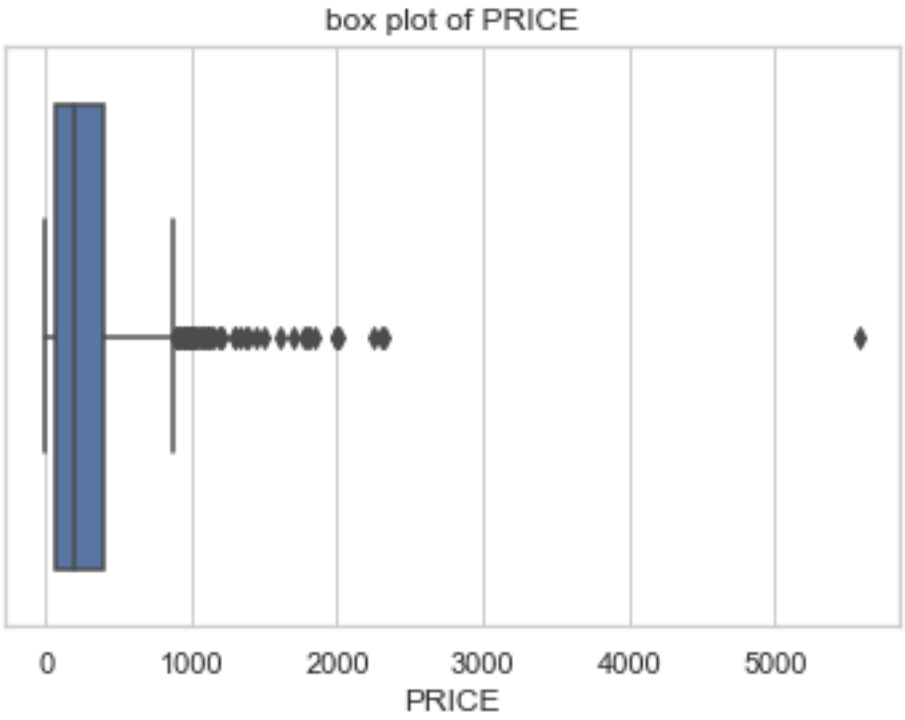
```
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 1920, height = 1080,
                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()
```



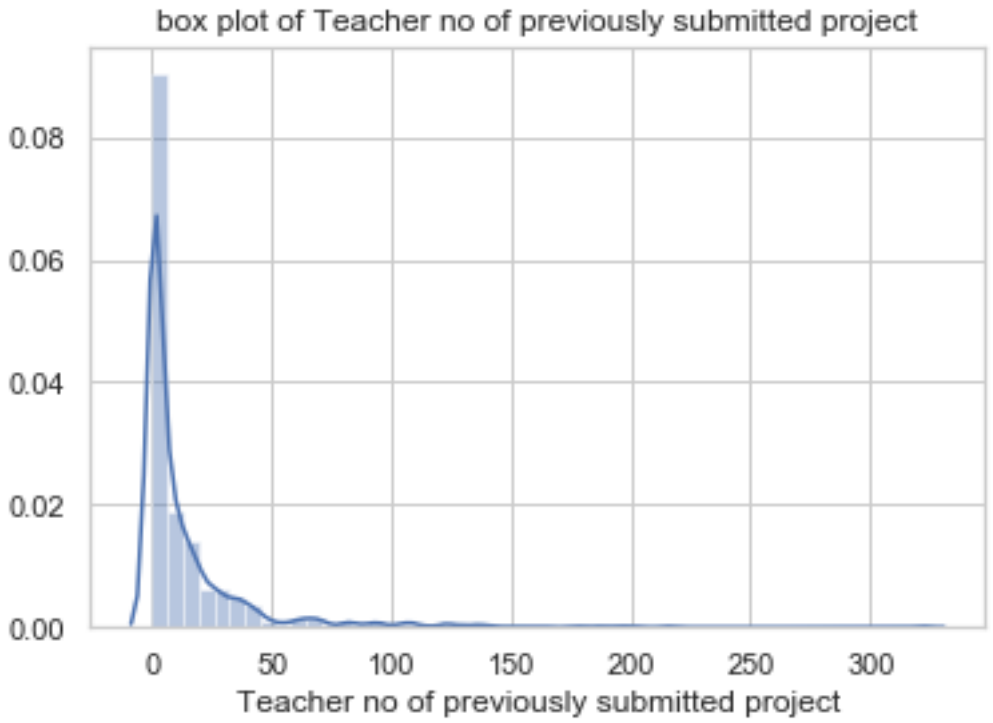
Observation: As Student, School, Classroom, Learning words are present, these points are being falsely predicted as positive.

```
In [103]: import seaborn as sns
          sns.set(style="whitegrid")
          tips = sns.load_dataset("tips")
```

```
ax = sns.boxplot(x=data_fp_whole_project_data.price)
plt.xlabel("PRICE")
plt.title("box plot of PRICE")
plt.show()
```



In [104]:
```
import seaborn as sns
sns.set(style="whitegrid")
tips = sns.load_dataset("tips")
ax = sns.distplot(data_fp_whole_project_data.teacher_number_of_previously_posted_proj
plt.xlabel("Teacher no of previously submitted project")
plt.title("box plot of Teacher no of previously submitted project")
plt.show()
```



2.5 [Task-2]Getting top 5k features using `feature_importances_`

```
In [105]: model=DecisionTreeClassifier(class_weight='balanced',max_depth=10,min_samples_split=5
          model.fit(BOW,project_data_Y_train)
          feature_sorted_accending=np.argsort(model.feature_importances_)
          feature_sorted_desccending=feature_sorted_accending[::-1]
```

```
In [106]: #create dataframe from sparse matrix for ease of selecting data
          feature_sorted_desccending=feature_sorted_desccending[:5000]
          bow_dataframe = pd.DataFrame(BOW.todense())
          print(bow_dataframe.shape)
          bow_dataframe_test = pd.DataFrame(BOW_test.todense())
          print(bow_dataframe.shape)
```

```
(87398, 18154)
(87398, 18154)
```

```
In [107]: #select top 5000 feature columns for train and test
          bow_dataframe_5000=bow_dataframe[feature_sorted_desccending]
          print(bow_dataframe_5000.shape)
          bow_dataframe_5000_test=bow_dataframe_test[feature_sorted_desccending]
          print(bow_dataframe_5000_test.shape)
```

```
(87398, 5000)
(21850, 5000)
```

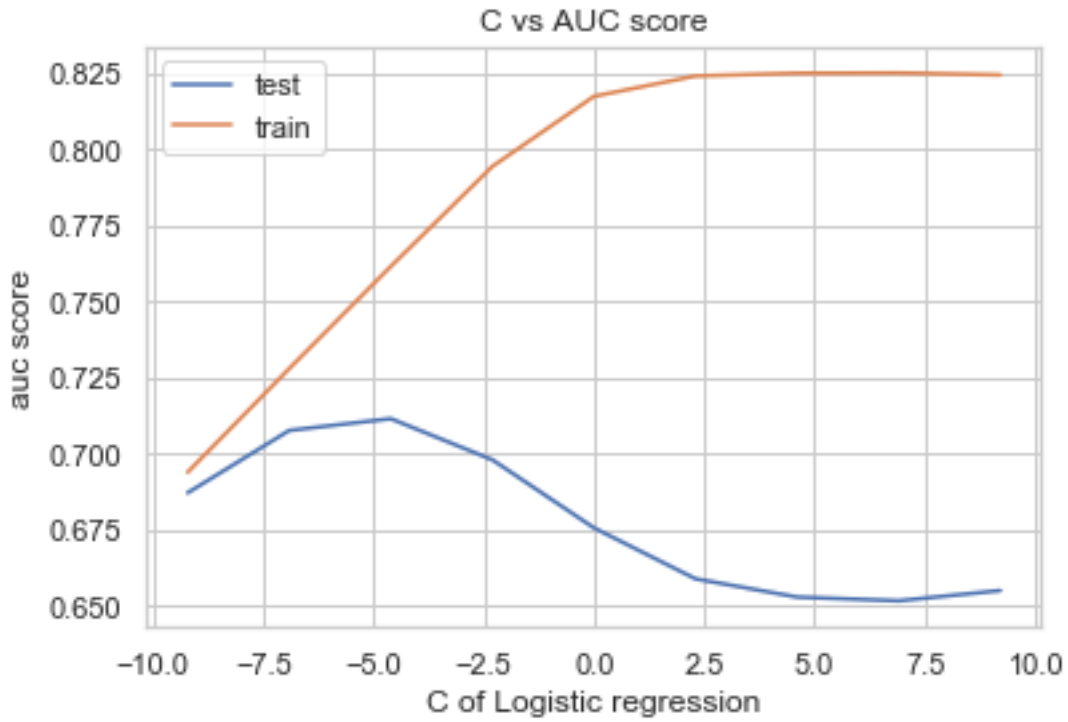
2.5.1 Using Logistic Regression

```
In [108]: from sklearn.linear_model import LogisticRegression
          from sklearn.model_selection import GridSearchCV
          model=LogisticRegression(class_weight='balanced')
          a=[10**-4,10**-3,10**-2,10**-1,10**0,10**1,10**2,10**3,10**4]
          print(a)
          parameters = {'C': a }
          clf = GridSearchCV(model, parameters,scoring='roc_auc',n_jobs=3,verbose=10)
          clf.fit(bow_dataframe_5000,project_data_Y_train)
```

```
[0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]
Fitting 3 folds for each of 9 candidates, totalling 27 fits


[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Done   2 tasks      | elapsed:  1.3min
[Parallel(n_jobs=3)]: Done   7 tasks      | elapsed:  1.7min
[Parallel(n_jobs=3)]: Done  12 tasks      | elapsed:  2.1min
[Parallel(n_jobs=3)]: Done  19 tasks      | elapsed:  4.8min
[Parallel(n_jobs=3)]: Done  25 out of  27 | elapsed:  7.4min remaining:   35.5s
[Parallel(n_jobs=3)]: Done  27 out of  27 | elapsed:  8.3min finished


Out[108]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                estimator=LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2', random_state=None,
                   solver='warn', tol=0.0001, verbose=0, warm_start=False),
                fit_params=None, iid='warn', n_jobs=3,
                param_grid={'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='roc_auc', verbose=10)
```

```
In [109]: k=a
          auc_cv=clf.cv_results_['mean_test_score']
          auc_train=clf.cv_results_['mean_train_score']
          plt.plot(np.log(k),auc_cv)
          plt.plot(np.log(k),auc_train)
          plt.title('C vs AUC score')
          plt.xlabel('C of Logistic regression')
          plt.ylabel('auc score')
          plt.legend({"test":"","train":""})
```

```
Out[109]: <matplotlib.legend.Legend at 0x27c824986a0>
```



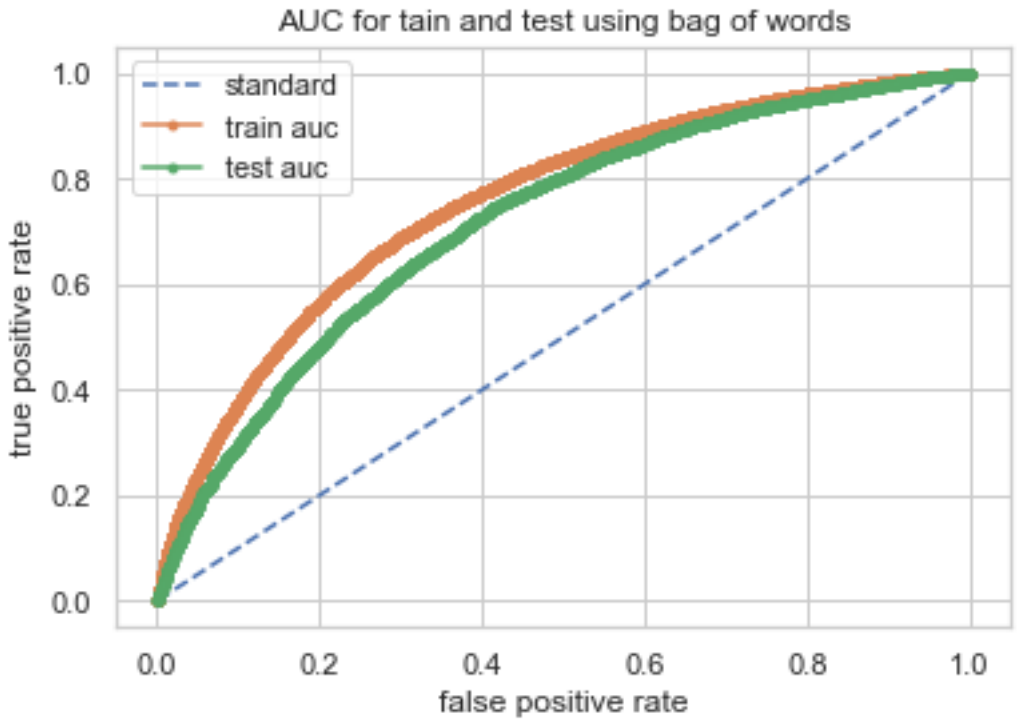

```
In [110]: print(clf.cv_results_['mean_test_score'])
          print(np.log(k))

[0.68718101 0.70768095 0.71154465 0.6980331  0.67564934 0.65890979
 0.6529187  0.65169472 0.65506402]
[-9.21034037 -6.90775528 -4.60517019 -2.30258509  0.          2.30258509
  4.60517019  6.90775528  9.21034037]


In [111]: print(np.exp(-4.60517019))

0.009999999959880915
```

10**-3 is the optimal value

```
In [112]: from sklearn.linear_model import LogisticRegression
          model=LogisticRegression(class_weight='balanced',C=10**-2,n_jobs=4)
          model.fit(bow_dataframe_5000,project_data_Y_train)

Out[112]: LogisticRegression(C=0.01, class_weight='balanced', dual=False,
                    fit_intercept=True, intercept_scaling=1, max_iter=100,
                    multi_class='warn', n_jobs=4, penalty='l2', random_state=None,
                    solver='warn', tol=0.0001, verbose=0, warm_start=False)
```

In [113]:
```
#https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-class
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from tqdm import tqdm

probs_test = model.predict_proba(bow_dataframe_5000_test)
# keep probabilities for the positive outcome only
probs_test = probs_test[:, 1]
auc_test = roc_auc_score(project_data_Y_test, probs_test)
print('AUC: %.3f' % auc_test)
fpr, tpr, thresholds = roc_curve(project_data_Y_test, probs_test)

probs_train = model.predict_proba(bow_dataframe_5000)
# keep probabilities for the positive outcome only
probs_train = probs_train[:, 1]
auc_train = roc_auc_score(project_data_Y_train, probs_train)
print('AUC: %.3f' % auc_train)
fpr1, tpr1, thresholds1 = roc_curve(project_data_Y_train, probs_train)


plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr1, tpr1, marker='.')
plt.plot(fpr, tpr, marker='.')

plt.legend({"standard":"","train auc":"","test auc":""})
plt.title("AUC for tain and test using bag of words")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.show()
```

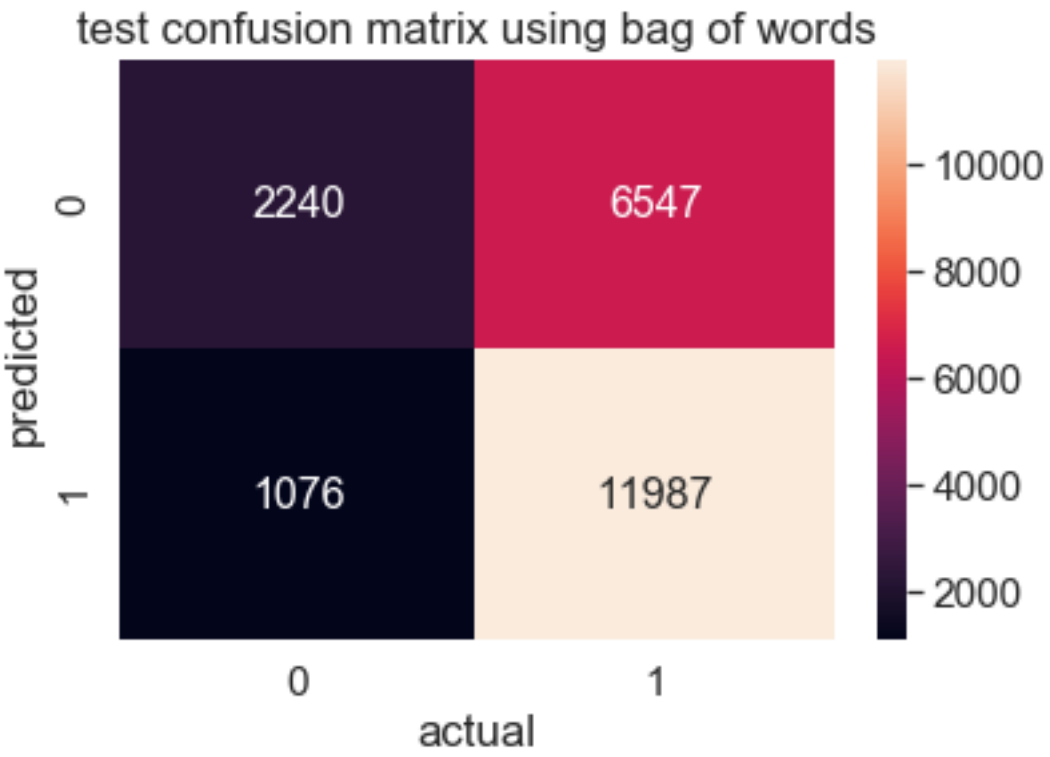
```
AUC: 0.718
AUC: 0.756
```



In [114]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#compute confudion matrix values and plot
from sklearn.metrics import confusion_matrix
predicted_bow_test=model.predict(bow_dataframe_5000_test)
tn, fp, fn, tp = confusion_matrix(project_data_Y_test,predicted_bow_test).ravel()
print(tn, fp, fn, tp)
print("true positive rate",(tp/(tp+fn)))
print("true negaitive rate",(tn/(tn+fp)))
matrix=[[tn,fn],[fp,tp]]
print(matrix)
df_cm = pd.DataFrame(matrix, range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')# font size
plt.title("test confusion matrix using bag of words")
plt.xlabel("actual")
plt.ylabel("predicted")
plt.show()
```

```
2240 1076 6547 11987
true positive rate 0.6467573108880975
true negaitive rate 0.6755126658624849
```

```
[[2240, 6547], [1076, 11987]]
```

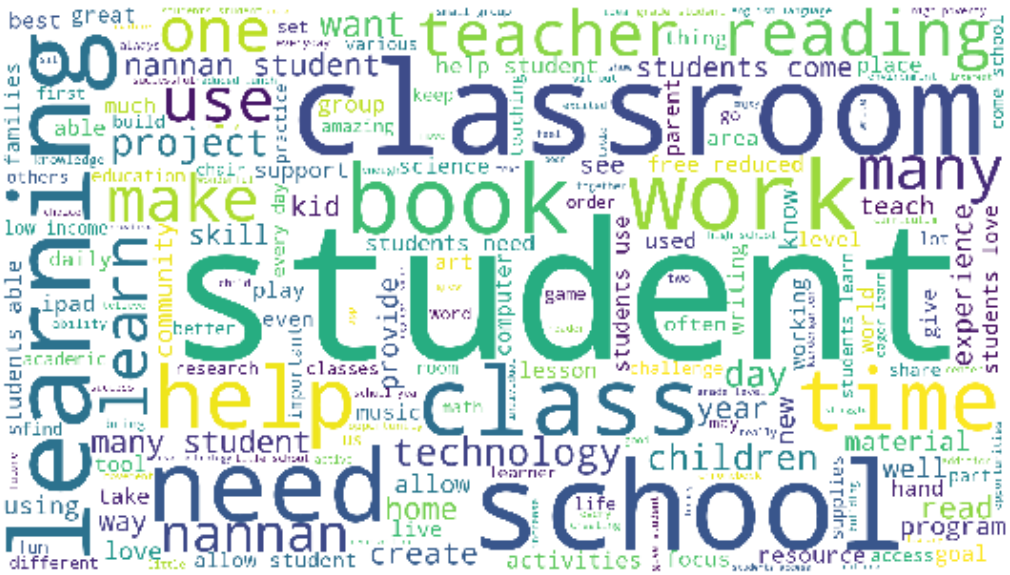


In [115]:
```
#collect ppredicted value and actual value from model and available labels respectiv
predicted_bow_test=model.predict(bow_dataframe_5000_test)

#create mask as your wish
mask1 = predicted_bow_test > 0.5
mask2 = (project_data_Y_test.values == 0)

#combine the mask and select the index of mask
special_mask=[]
count=0
for i in range(len(mask1)):
    #print(i)
    if(mask1[i] and mask2[i]):
        #print(mask1[i] and mask2[i])
        special_mask.append(i)
        count+=1

#get the copy of data frame
data_fp_whole_project_data = project_data_X_test
print("Whole test data shape",data_fp_whole_project_data.shape)
```

```
        #reset the index
        a = np.arange(0,data_fp_whole_project_data.shape[0])
        data_fp_whole_project_data.index=a

        #apply same mask
        data_fp_whole_project_data=data_fp_whole_project_data.iloc[special_mask,]
        print("False positive from test data shape",data_fp_whole_project_data.shape)
```

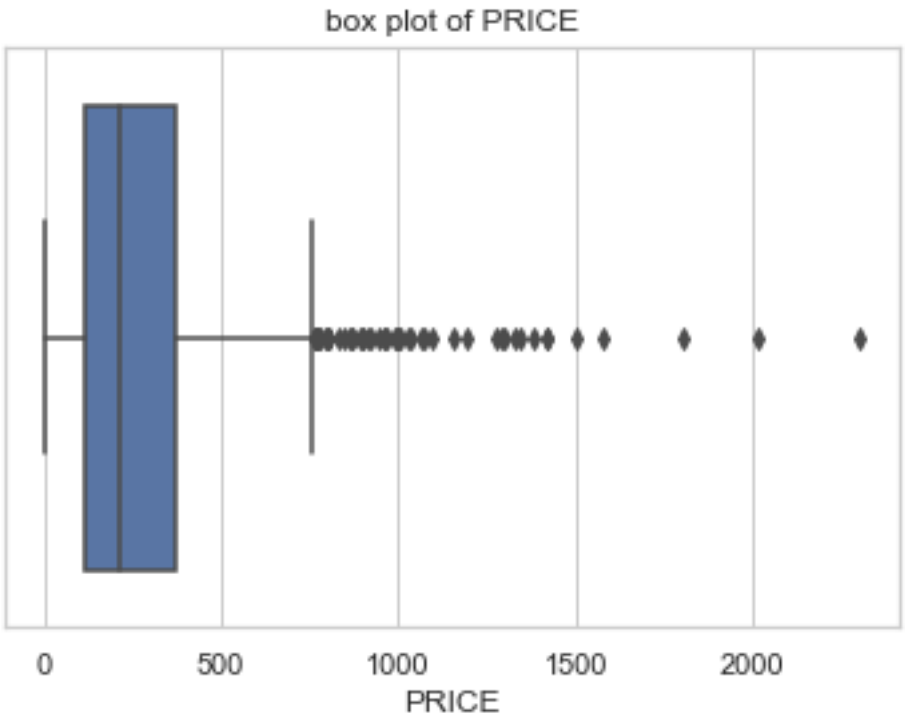
```
Whole test data shape (21850, 23)
False positive from test data shape (1076, 23)
```

```
In [116]: from wordcloud import WordCloud, STOPWORDS
          import matplotlib.pyplot as plt
          import pandas as pd

          comment_words = ' '
          stopwords = set(STOPWORDS)

          # iterate through the csv file
          for val in data_fp_whole_project_data.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()

              for words in tokens:
                  comment_words = comment_words + words + ' '


          wordcloud = WordCloud(width = 1920, height = 1080,
                          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()
```

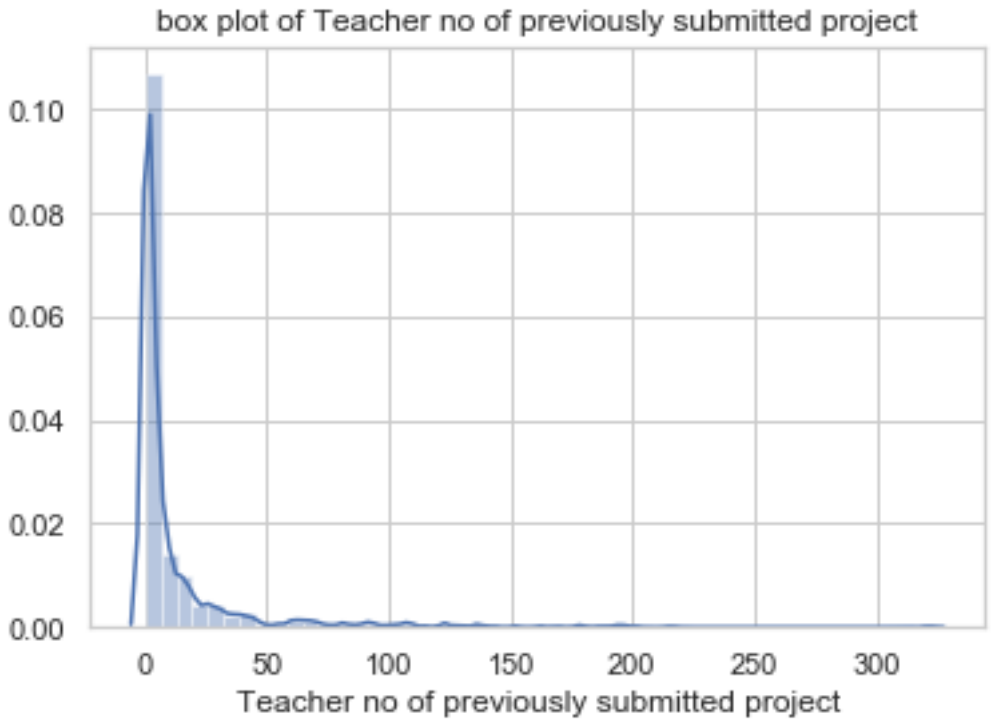


Observation: As Student, School, Classroom, Learning words are present, these points are being falsely predicted as positive.

```
In [117]: import seaborn as sns
          sns.set(style="whitegrid")
          tips = sns.load_dataset("tips")
          ax = sns.boxplot(x=data_fp_whole_project_data.price)
          plt.xlabel("PRICE")
          plt.title("box plot of PRICE")
          plt.show()
```




```
In [118]: import seaborn as sns
          sns.set(style="whitegrid")
          tips = sns.load_dataset("tips")
          ax = sns.distplot(data_fp_whole_project_data.teacher_number_of_previously_posted_proj
          plt.xlabel("Teacher no of previously submitted project")
          plt.title("box plot of Teacher no of previously submitted project")
          plt.show()
```



2.0.5 The results of logistic regression seems promising even though we had only 5000 best features from decision tree.

learning pointWe can pick decision tree for feature selection extensively

3. Conclusion

```
In [14]: from prettytable import PrettyTable
         x = PrettyTable()
         x.field_names = ["Vectorizer", "Model","penalty", "alpha", "max_depth", "min_sample_s
         x.add_row(["BAG of words", "Decision tree", "N/A", "N/A",10,500,0.658])
         x.add_row(["TFIDF", "Decision tree" , "N/A", "N/A",10,500,0.655])
         x.add_row(["Average W2V", "Decision tree","N/A", "N/A",5,500,0.624])
         x.add_row(["TFIDF W2V", "Decision tree" ,"N/A",  "N/A",5,500,0.642])
         x.add_row(["LR on best 5000 from decison tree", "Logistic regression" ,"l2", 0.01,"N/
         x.border=True
         print(x)
```

```
+-----------------------------------+---------------------+---------+-------+-----------+-----
|            Vectorizer             |        Model        | penalty | alpha | max_depth | min_
+-----------------------------------+---------------------+---------+-------+-----------+-----
|           BAG of words            |    Decision tree    |   N/A   |  N/A  |     10    |
|              TFIDF                |    Decision tree    |   N/A   |  N/A  |     10    |
|           Average W2V             |    Decision tree    |   N/A   |  N/A  |     5     |
```

TFIDF W2V	Decision tree	N/A	N/A	5
LR on best 5000 from decison tree	Logistic regression	l2	0.01	N/A

Let me know if my base of understanding on crazy cheating idea is correct or not, though I will never breach information. :)

One point I came to know by analysing the false positive that, somone can cheat a machine learning model if given resourses. For example if I am a developer at DonorsChoose.org and I want my uncle's project to be accepted. In that case if I breach the information and ask my uncle to use words such as: Student, School, Classroom, Learning in his project then his project is most likely to be accepted.