prabhudayala@gmail.com_4

April 17, 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 [2]: %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 [3]: project_data = pd.read_csv('train_data.csv')
        resource_data = pd.read_csv('resources.csv')
```

```
In [4]: 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 [5]: 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[5]:         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 [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'],
              dtype='object')
```

```
In [7]: # 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 [8]: project_data.columns
```

```
Out[8]: 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 [9]: 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 [10]: sub_catogories = list(project_data['project_subject_subcategories'].values)
         # remove special characters from list of strings python: https://stackoverflow.com/a/

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

         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", "Warm
                 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 replac
                 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/22898595/40840
    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 [11]: # 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 [12]: project_data.head(2)

Out[12]:    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 [13]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [14]:
```
# 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 [15]:
```
# 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 [16]:
```
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 [17]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-
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 [18]:
```
#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 [19]:
```
# 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'
            "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', "
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', '
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'a
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'throug
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', '
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'a
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'to
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", '
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mi
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
            'won', "won't", 'wouldn', "wouldn't"]
```

```
In [20]: # 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.lower() not in stopwords)
             preprocessed_essays.append(sent.lower().strip())
```

```
100%|| 109248/109248 [00:43<00:00, 2484.86it/s]
```

```
In [21]: # after preprocesing
         print(project_data.essay.values[20000])
         print(preprocessed_essays[20000])
         project_data['essay']=preprocessed_essays
         print(project_data.essay.values[20000])
```

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

1.4 Preprocessing of project_title

```
In [22]: 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, 55727.11it/s]
```

```
In [23]: print(project_data['project_title'].values[20001])
         project_data['project_title']=preprocessed_project_title
         print(project_data['project_title'].values[20001])
```

```
The Beautiful Life of a Butterfly
beautiful life butterfly
```

2 Assignment 4: Naive Bayes

```
<li><strong>Apply Multinomial NaiveBayes on these feature sets</strong>
    <ul>
        <li><font color='red'>Set 1</font>: categorical, numerical features + project_title(BOW
        <li><font color='red'>Set 2</font>: categorical, numerical features + project_title(TFI
    </ul>
</li>
<br>
<li><strong>The hyper paramter tuning(find best Alpha)</strong>
    <ul>
<li>Find the best hyper parameter which will give the maximum <a href='https://www.appliedaicou
<li>Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001</l
<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>Feature importance</strong>
    <ul>
<li>Find the top 10 features of positive class and top 10 features of negative class for both f
    </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>
    </ul>
</li>
<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. Naive Bayes

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling
2.1.1 Under Sampling OR over sampling

```
In [24]: sampling=True
         undersampling=True
         if (not sampling):
             print("Total data ",project_data.shape)
             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)
         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_X=pd.concat([project_data_positive,project_data_negative])
                 project_data_Y=project_data_X.project_is_approved
                 project_data_X=project_data_X.drop(columns=['project_is_approved'])
                 print("After sampling: ",project_data_X.shape)
             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_X=pd.concat([project_data_positive,project_data_negative])
                 project_data_Y=project_data_X.project_is_approved
                 project_data_X=project_data_X.drop(columns=['project_is_approved'])
                 print("After sampling: ",project_data_X.shape)
```

```
Total data  (109248, 20)
Positive points:  16542
Negaitive points:  16542
After sampling:  (33084, 19)
```

```
In [25]: from sklearn.model_selection import train_test_split
         project_data_X_train,project_data_X_test,project_data_Y_train,project_data_Y_test=trai

In [26]: print(project_data_X_train.shape)
         print(project_data_X_test.shape)
         print(project_data_Y_train.shape)
         print(project_data_Y_test.shape)
```

```
(23158, 19)
(9926, 19)
(23158,)
(9926,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 Categorical features

```
In [27]: from sklearn.feature_extraction.text import CountVectorizer
         clean_categories_vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys())
         clean_categories_vectorizer.fit(project_data_X_train['clean_categories'].values)
         print(clean_categories_vectorizer.get_feature_names())

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


         #for test
         categories_one_hot_test = clean_categories_vectorizer.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  (23158, 9)
Shape of matrix after one hot encodig  (9926, 9)
```

```
In [28]: clean_subcategories_vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.
         clean_subcategories_vectorizer.fit(project_data_X_train['clean_subcategories'].values
         print(clean_subcategories_vectorizer.get_feature_names())

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

         #for test
         sub_categories_one_hot_test = clean_subcategories_vectorizer.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  (23158, 30)
Shape of matrix after one hot encodig  (9926, 30)
```

```
In [29]: project_data_X_train.teacher_prefix = project_data_X_train.teacher_prefix.replace(np.
         print(project_data_X_train.teacher_prefix.value_counts())
         project_data_X_test.teacher_prefix = project_data_X_test.teacher_prefix.replace(np.na
         print(project_data_X_test.teacher_prefix.value_counts())
```

```
Mrs.       11863
Ms.         8406
Mr.         2284
Teacher      599
Dr.            5
               1
Name: teacher_prefix, dtype: int64
Mrs.       5119
Ms.        3579
Mr.         996
Teacher     230
Dr.           2
Name: teacher_prefix, dtype: int64
```

In [30]:
```
# we use count vectorizer to convert the values into one hot encoded features
teacher_prefix_vectorizer = CountVectorizer(vocabulary=list(project_data_X_train['tea
teacher_prefix_vectorizer.fit(project_data_X_train['teacher_prefix'].values)
print(teacher_prefix_vectorizer.get_feature_names())

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

teacher_prefix_one_hot_test = teacher_prefix_vectorizer.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  (23158, 6)
Shape of matrix after one hot encodig  (9926, 6)
```

In [31]:
```
# we use count vectorizer to convert the values into one hot encoded features
project_grade_category_vectorizer = CountVectorizer(vocabulary=list(project_data_X_tra
project_grade_category_vectorizer.fit(project_data_X_train['project_grade_category'].
print(project_grade_category_vectorizer.get_feature_names())

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

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

```
['Grades 3-5', 'Grades 6-8', 'Grades 9-12', 'Grades PreK-2']
Shape of matrix after one hot encodig  (23158, 4)
Shape of matrix after one hot encodig  (9926, 4)
```

In [32]:
```
# we use count vectorizer to convert the values into one hot encoded features
school_state_vectorizer = CountVectorizer(vocabulary=list(project_data_X_train['schoo
```

```
        school_state_vectorizer.fit(project_data_X_train['school_state'].values)
        print(school_state_vectorizer.get_feature_names())


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

```
['NY', 'CO', 'SC', 'CA', 'MI', 'PA', 'IL', 'DC', 'GA', 'AZ', 'IN', 'LA', 'TX', 'AR', 'MN', 'UT
Shape of matrix after one hot encodig  (23158, 51)
Shape of matrix after one hot encodig  (9926, 51)
```

2.2.2 Numerical features

```
In [33]: # 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.var

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

```
Mean : 323.8183128940323, Standard deviation : 371.64607018076947
```

```
In [34]: # 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['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 : 9.234130753951119, Standard deviation : 23.32717552963384
```

2.3 Make Data Model Ready: encoding eassay, and project_title

2.3.1 Bag of words

In [35]:
```
bow_vectorizer_essay = CountVectorizer(min_df=10)
bow_vectorizer_essay.fit(project_data_X_train.essay.values)

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

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

```
(23158, 8831)
(9926, 8831)
```

In [36]:
```
# Similarly you can vectorize for title also
bow_vectorizer_title = CountVectorizer(min_df=10)
bow_vectorizer_title.fit(project_data_X_train.project_title.values)

title_text_bow_train=bow_vectorizer_title.fit_transform(project_data_X_train.project_
print(title_text_bow_train.shape)

title_text_bow_test=bow_vectorizer_title.transform(project_data_X_test.project_title.
print(title_text_bow_test.shape)
print(len(bow_vectorizer_title.get_feature_names()))
```

```
(23158, 1159)
(9926, 1159)
1159
```

2.3.2 TFIDF

In [37]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(min_df=10)
tfidf_vectorizer.fit(project_data_X_train.essay.values)
```

```
text_tfidf_train=tfidf_vectorizer.fit_transform(project_data_X_train.essay.values)
print(text_tfidf_train.shape)

text_tfidf_test=tfidf_vectorizer.transform(project_data_X_test.essay.values)
print(text_tfidf_test.shape)
print(len(tfidf_vectorizer.vocabulary_))
```

```
(23158, 8831)
(9926, 8831)
8831
```

```
In [38]: # Similarly you can vectorize for title also
         from sklearn.feature_extraction.text import TfidfVectorizer
         tfidf_vectorizer_title = TfidfVectorizer(min_df=10)
         tfidf_vectorizer_title.fit(project_data_X_train.project_title.values)

         title_text_tfidf_train=tfidf_vectorizer_title.fit_transform(project_data_X_train.proje
         print(title_text_tfidf_train.shape)

         title_text_tfidf_test=tfidf_vectorizer_title.transform(project_data_X_test.project_tit
         print(title_text_tfidf_test.shape)
```

```
(23158, 1159)
(9926, 1159)
```

2.4 Appling NB() on different kind of featurization as mentioned in the instructions

Apply Naive Bayes 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 Naive Bayes on BOW, SET 1

```
In [39]: # 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)
```

```
(23158, 10088)
(9926, 10088)
```

```
In [40]: from sklearn.naive_bayes import MultinomialNB
         from sklearn.model_selection import GridSearchCV
         model=MultinomialNB()
         a=[0.00001,0.0001,0.001,0.01,1,10,100]
```

```
        print(a)
        parameters = {'alpha': a }
        clf = GridSearchCV(model, parameters, scoring='roc_auc',n_jobs=1,verbose=10)
        clf.fit(BOW,project_data_Y_train)
```

```
[1e-05, 0.0001, 0.001, 0.01, 1, 10, 100]
Fitting 3 folds for each of 7 candidates, totalling 21 fits


[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.


[CV] alpha=1e-05 ...
[CV] ... alpha=1e-05, score=0.6534990886638128, total=   0.0s


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s


[CV] alpha=1e-05 ...
[CV] ... alpha=1e-05, score=0.6458044463575994, total=   0.0s


[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s


[CV] alpha=1e-05 ...
[CV] ... alpha=1e-05, score=0.654453487174619, total=   0.0s


[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.1s remaining:    0.0s


[CV] alpha=0.0001 ...
[CV] ... alpha=0.0001, score=0.6537519150744878, total=   0.0s


[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.1s remaining:    0.0s


[CV] alpha=0.0001 ...
[CV] ... alpha=0.0001, score=0.6458526481941297, total=   0.0s


[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.2s remaining:    0.0s


[CV] alpha=0.0001 ...
[CV] ... alpha=0.0001, score=0.6544823210030171, total=   0.0s
```

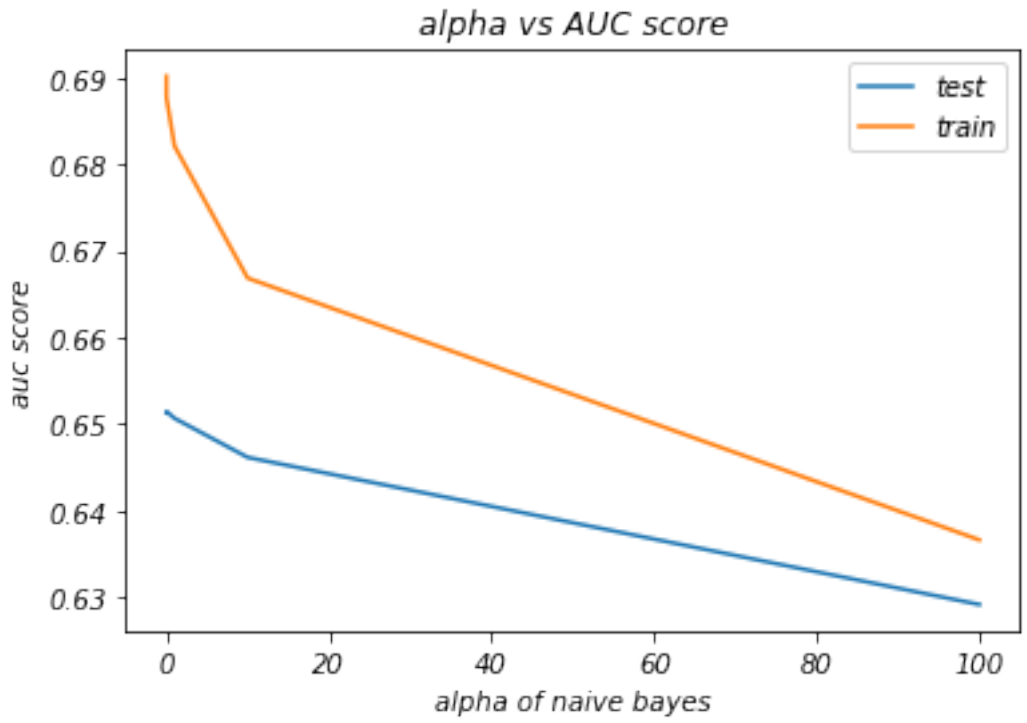
```
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.2s remaining:    0.0s


[CV] alpha=0.001 ...
[CV] ... alpha=0.001, score=0.6538586296996466, total=   0.0s

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.3s remaining:    0.0s


[CV] alpha=0.001 ...
[CV] ... alpha=0.001, score=0.6459314293294615, total=   0.0s

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.4s remaining:    0.0s


[CV] alpha=0.001 ...
[CV] ... alpha=0.001, score=0.6544878930816133, total=   0.0s

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.4s remaining:    0.0s


[CV] alpha=0.01 ...
[CV] ... alpha=0.01, score=0.6539160475372714, total=   0.0s
[CV] alpha=0.01 ...
[CV] ... alpha=0.01, score=0.6459890298527803, total=   0.0s
[CV] alpha=0.01 ...
[CV] ... alpha=0.01, score=0.6543712822319572, total=   0.0s
[CV] alpha=1 ...
[CV] ... alpha=1, score=0.6533792528378874, total=   0.0s
[CV] alpha=1 ...
[CV] ... alpha=1, score=0.6452048772739619, total=   0.0s
[CV] alpha=1 ...
[CV] ... alpha=1, score=0.6533255440597682, total=   0.0s
[CV] alpha=10 ...
[CV] ... alpha=10, score=0.648452124496906, total=   0.0s
[CV] alpha=10 ...
[CV] ... alpha=10, score=0.6411098694401565, total=   0.0s
[CV] alpha=10 ...
[CV] ... alpha=10, score=0.6488032316444654, total=   0.0s
[CV] alpha=100 ...
[CV] ... alpha=100, score=0.6300638187016508, total=   0.0s
[CV] alpha=100 ...
[CV] ... alpha=100, score=0.6253208476702468, total=   0.0s
[CV] alpha=100 ...
[CV] ... alpha=100, score=0.6319644772604815, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    1.1s finished
```

```
Out[40]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True),
                fit_params=None, iid='warn', n_jobs=1,
                param_grid={'alpha': [1e-05, 0.0001, 0.001, 0.01, 1, 10, 100]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='roc_auc', verbose=10)
```

```
In [41]: k=a
         auc_cv=clf.cv_results_['mean_test_score']
         auc_train=clf.cv_results_['mean_train_score']
         plt.plot(k,auc_cv)
         plt.plot(k,auc_train)
         plt.title('alpha vs AUC score')
         plt.xlabel('alpha of naive bayes')
         plt.ylabel('auc score')
         plt.legend({"test":"","train":""})
```

```
Out[41]: <matplotlib.legend.Legend at 0x187031c98d0>
```




```
In [42]: x=np.argsort(auc_cv)
         optimal_value=k[x[-1]]
         print("optimal value is: ",optimal_value)
```

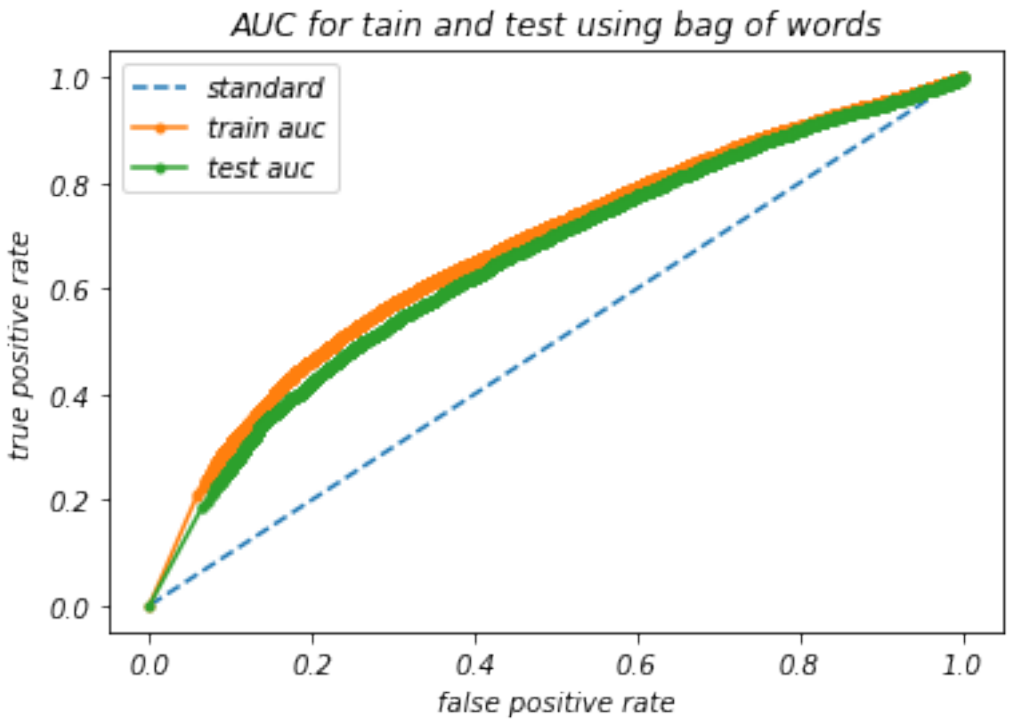
```
optimal value is:  0.001
```

```
In [43]: from sklearn.model_selection import GridSearchCV
         model=MultinomialNB(alpha=optimal_value)
         model.fit(BOW,project_data_Y_train)
```

```
Out[43]: MultinomialNB(alpha=0.001, class_prior=None, fit_prior=True)
```

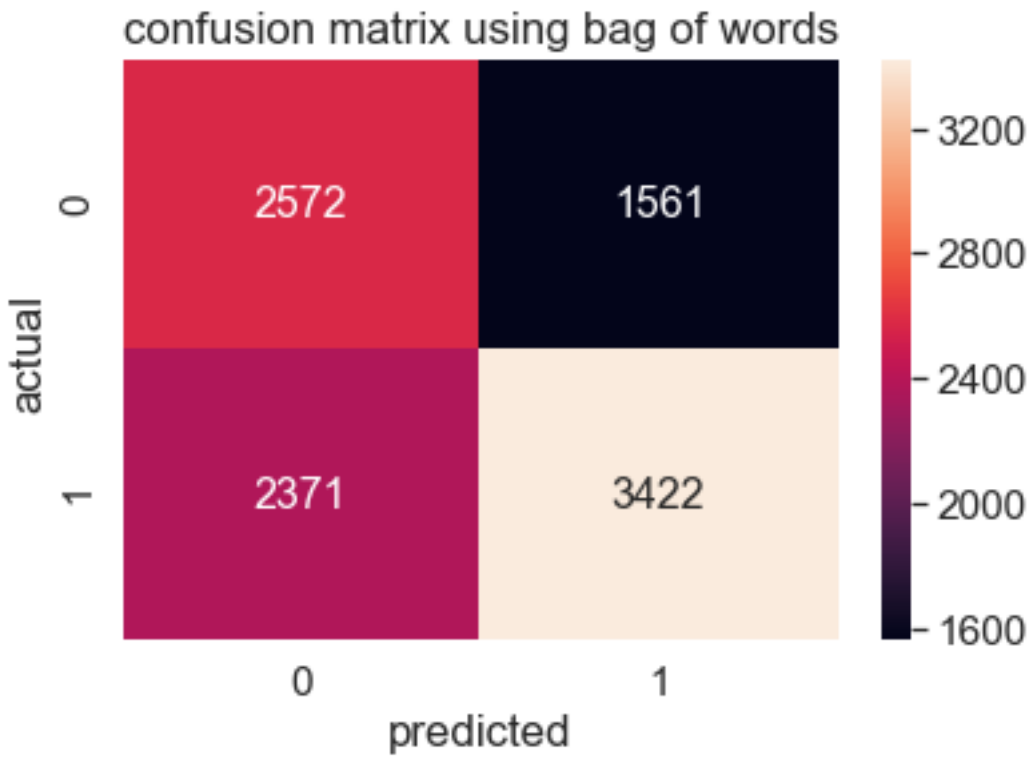
```
In [44]: #https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classi
         from sklearn.metrics import roc_curve
         from sklearn.metrics import roc_auc_score

         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: 0.653
AUC: 0.675
```



In [45]:
```
#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)

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("confusion matrix using bag of words")
plt.xlabel("predicted")
plt.ylabel("actual")
plt.show()
```

```
2572 2371 1561 3422
[[2572, 1561], [2371, 3422]]
```



In [46]:
```
features_list_bow=[]
features_list_bow=clean_categories_vectorizer.get_feature_names()+clean_subcategories_
features_list_bow.append("price")
features_list_bow.append("teacher_number_of_previously_posted_projects")
features_list_bow+=bow_vectorizer_essay.get_feature_names()
features_list_bow+=bow_vectorizer_title.get_feature_names()
print(len(features_list_bow))
```

```
10088
```

In [47]:
```
'''model.coef_ will not suit my purpose as it shows importance of words on whole mode
gives us probability for both positive and negative class.

I peronlayy believe that feature_log_prob_ is more useful as it tell me which word ha
which word have more weights for negative class independently.

Please validate my understanding.

'''
```

Out[47]:
```
'model.coef_ will not suit my purpose as it shows importance of words on whole model.
```

2.4.1.1 Top 10 important features of positive class from SET 1

```
In [48]: #https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-
         pos_class_prob_sorted = model.feature_log_prob_[1, :].argsort()[::-1]
         print(np.take(features_list_bow, pos_class_prob_sorted[:10]))
```

```
['price' 'teacher_number_of_previously_posted_projects' 'students'
 'school' 'learning' 'classroom' 'not' 'learn' 'help' 'many']
```

2.4.1.2 Top 10 important features of negative class from SET 1

```
In [49]: #https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-
         neg_class_prob_sorted = model.feature_log_prob_[0, :].argsort()[::-1]
         print(np.take(features_list_bow, neg_class_prob_sorted[:10]))
```

```
['price' 'students' 'teacher_number_of_previously_posted_projects'
 'school' 'learning' 'classroom' 'not' 'learn' 'help' 'nannan']
```

2.0.2 2.4.2 Applying Naive Bayes on TFIDF, SET 2

```
In [50]: # 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_o
         print(TFIDF.shape)
         TFIDF_test = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_stat
         print(TFIDF_test.shape)
```

```
(23158, 10088)
(9926, 10088)
```

```
In [51]: from sklearn.naive_bayes import MultinomialNB
         from sklearn.model_selection import GridSearchCV
         model=MultinomialNB()
         a=[0.00001,0.0001,0.001,0.01,1,10]
         print(a)
         parameters = {'alpha': a }
         clf = GridSearchCV(model, parameters, scoring='roc_auc',n_jobs=1,verbose=10)
         clf.fit(TFIDF,project_data_Y_train)
```

```
[1e-05, 0.0001, 0.001, 0.01, 1, 10]
Fitting 3 folds for each of 6 candidates, totalling 18 fits
```

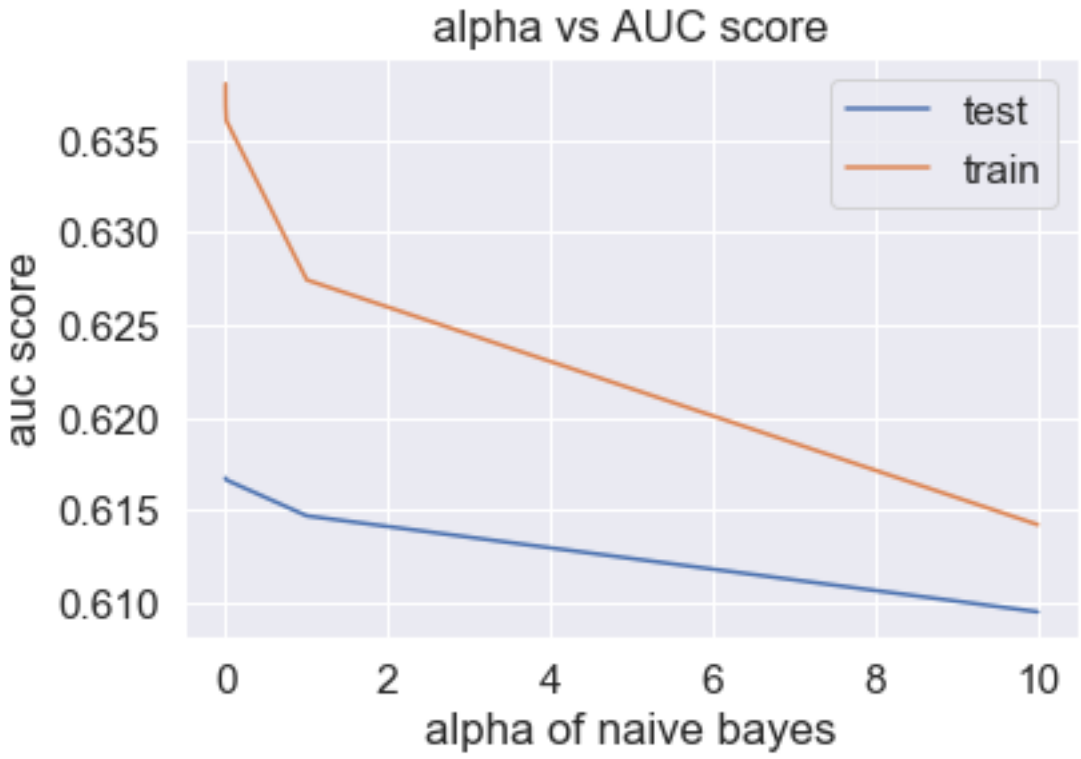
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
[CV] alpha=1e-05 ...
[CV] ... alpha=1e-05, score=0.6157867777357855, total=   0.0s

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s

[CV] alpha=1e-05 ...
[CV] ... alpha=1e-05, score=0.6146736460419646, total=   0.0s

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s

[CV] alpha=1e-05 ...
[CV] ... alpha=1e-05, score=0.6197611216339107, total=   0.0s

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.1s remaining:    0.0s

[CV] alpha=0.0001 ...
[CV] ... alpha=0.0001, score=0.6158468131019519, total=   0.0s

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.1s remaining:    0.0s

[CV] alpha=0.0001 ...
[CV] ... alpha=0.0001, score=0.6146956993891793, total=   0.0s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.2s remaining:    0.0s

[CV] alpha=0.0001 ...
[CV] ... alpha=0.0001, score=0.6196482702589701, total=   0.0s

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.2s remaining:    0.0s

[CV] alpha=0.001 ...
[CV] ... alpha=0.001, score=0.6159037275687033, total=   0.0s

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.3s remaining:    0.0s

[CV] alpha=0.001 ...
[CV] ... alpha=0.001, score=0.6146820041598587, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.4s remaining:    0.0s


[CV] alpha=0.001 ...
[CV] ... alpha=0.001, score=0.6195299139389104, total=   0.0s

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.4s remaining:    0.0s


[CV] alpha=0.01 ...
[CV] ... alpha=0.01, score=0.6159215804556795, total=   0.0s
[CV] alpha=0.01 ...
[CV] ... alpha=0.01, score=0.6146391058680164, total=   0.0s
[CV] alpha=0.01 ...
[CV] ... alpha=0.01, score=0.6193272715383998, total=   0.0s
[CV] alpha=1 ...
[CV] ... alpha=1, score=0.6141528358807591, total=   0.0s
[CV] alpha=1 ...
[CV] ... alpha=1, score=0.6126220805496996, total=   0.0s
[CV] alpha=1 ...
[CV] ... alpha=1, score=0.6172920866145379, total=   0.0s
[CV] alpha=10 ...
[CV] ... alpha=10, score=0.6086391126819862, total=   0.0s
[CV] alpha=10 ...
[CV] ... alpha=10, score=0.607237102954155, total=   0.0s
[CV] alpha=10 ...
[CV] ... alpha=10, score=0.6126323519716901, total=   0.0s

[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    0.9s finished


Out[51]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True),
                fit_params=None, iid='warn', n_jobs=1,
                param_grid={'alpha': [1e-05, 0.0001, 0.001, 0.01, 1, 10]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='roc_auc', verbose=10)
```

```
In [52]: k=a
         auc_cv=clf.cv_results_['mean_test_score']
         auc_train=clf.cv_results_['mean_train_score']
         plt.plot(k,auc_cv)
         plt.plot(k,auc_train)
         plt.title('alpha vs AUC score')
         plt.xlabel('alpha of naive bayes')
         plt.ylabel('auc score')
         plt.legend({"test":"","train":""})
```

Out[52]: <matplotlib.legend.Legend at 0x1870891bcf8>



In [53]:
```
x=np.argsort(auc_cv)
optimal_value=k[x[-1]]
print("optimal value is: ",optimal_value)
```

```
optimal value is:  1e-05
```

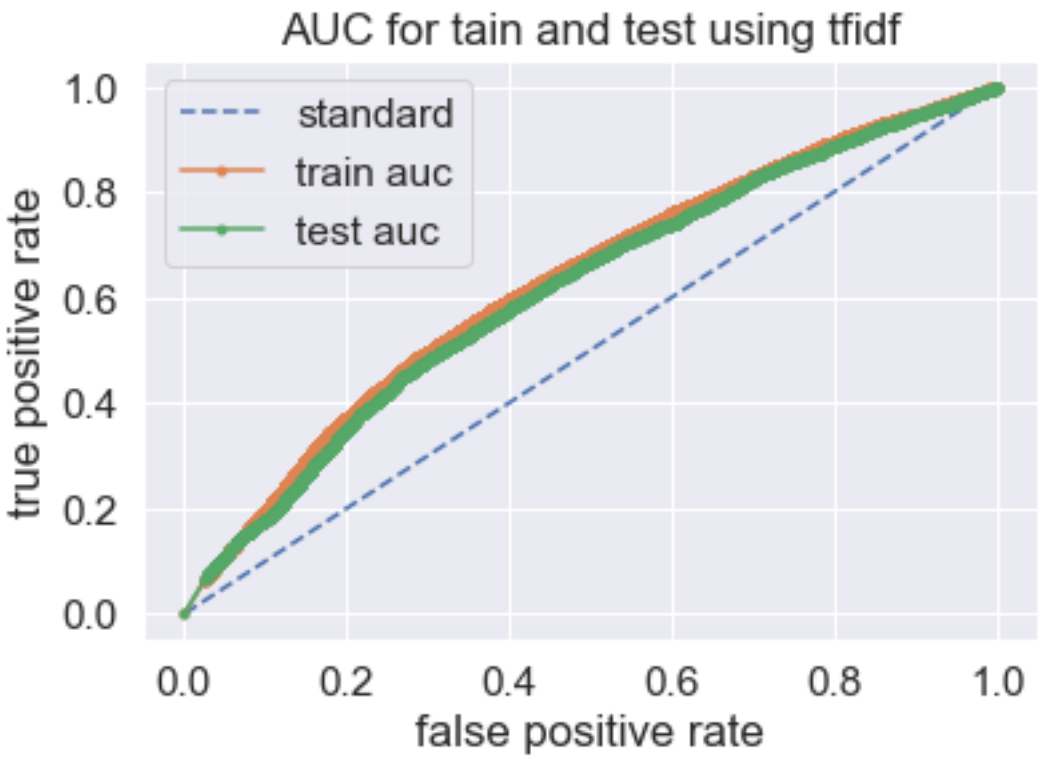
In [54]:
```
from sklearn.model_selection import GridSearchCV
model=MultinomialNB(alpha=optimal_value)
model.fit(TFIDF,project_data_Y_train)
```

Out[54]: MultinomialNB(alpha=1e-05, class_prior=None, fit_prior=True)

In [55]:
```
#https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classi
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

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)
```

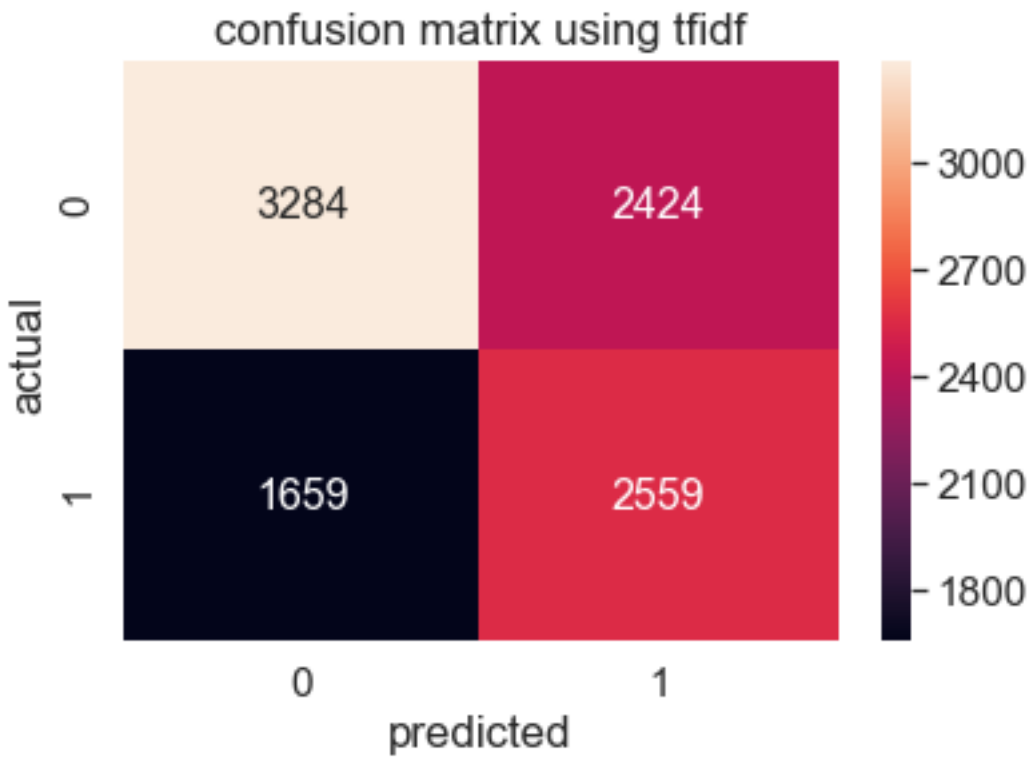
```
print('AUC: %.3f' % auc_test)
fpr, tpr, thresholds = roc_curve(project_data_Y_test, probs_test)

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 tfidf")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.show()
```

```
AUC: 0.617
AUC: 0.629
```



In [56]:
```
#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)

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("confusion matrix using tfidf")
plt.xlabel("predicted")
plt.ylabel("actual")
plt.show()
```

```
3284 1659 2424 2559
[[3284, 2424], [1659, 2559]]
```



```
In [57]: features_list_tfidf=[]
         features_list_tfidf=clean_categories_vectorizer.get_feature_names()+clean_subcategorie
         features_list_tfidf.append("price")
         features_list_tfidf.append("teacher_number_of_previously_posted_projects")
         features_list_tfidf+=tfidf_vectorizer.get_feature_names()
         features_list_tfidf+=tfidf_vectorizer_title.get_feature_names()
         print(len(features_list_tfidf))
```

```
10088
```

2.4.2.1 Top 10 important features of positive class from SET 2

```
In [58]: pos_class_prob_sorted = model.feature_log_prob_[1, :].argsort()[::-1]
         print(np.take(features_list_bow, pos_class_prob_sorted[:10]))
```

```
['price' 'teacher_number_of_previously_posted_projects'
 'Literacy_Language' 'Math_Science' 'Literacy' 'Mathematics'
 'Literature_Writing' 'CA' 'students' 'Health_Sports']
```

2.4.2.2 Top 10 important features of negative class from SET 2

```
In [59]: #https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-
         neg_class_prob_sorted = model.feature_log_prob_[0, :].argsort()[::-1]
         print(np.take(features_list_bow, neg_class_prob_sorted[:10]))
```

```
['price' 'teacher_number_of_previously_posted_projects'
 'Literacy_Language' 'Math_Science' 'Mathematics' 'Literacy'
 'Literature_Writing' 'SpecialNeeds' 'SpecialNeeds' 'students']
```

3. Conclusions

```
In [60]: from prettytable import PrettyTable
         x = PrettyTable()
         x.field_names = ["Vectorizer", "Model","Over Sampling","Under Sampling","alpha", "AUC"
         x.add_row(["BAG of words", "Naive bayes", True, False, 0.00001, 0.673])
         x.add_row(["TFIDF", "Naive bayes" ,True, False , 0.00001, 0.637])
         x.add_row(["BAG of words", "Naive bayes", False, True,0.01, 0.652])
         x.add_row(["TFIDF", "Naive bayes" , False,True, 0.00001, 0.618])
         x.add_row(["BAG of words", "Naive bayes", False, False,10, 0.663])
         x.add_row(["TFIDF", "Naive bayes" , False,False, 0.001, 0.625])
         x.border=True
         print(x)
```

```
+-------------+-------------+---------------+----------------+-------+-------+
|  Vectorizer |    Model    | Over Sampling | Under Sampling | alpha |  AUC  |
+-------------+-------------+---------------+----------------+-------+-------+
```

BAG of words	Naive bayes	True	False	1e-05	0.673
TFIDF	Naive bayes	True	False	1e-05	0.637
BAG of words	Naive bayes	False	True	0.01	0.652
TFIDF	Naive bayes	False	True	1e-05	0.618
BAG of words	Naive bayes	False	False	10	0.663
TFIDF	Naive bayes	False	False	0.001	0.625

Lesson learnt: Imbalanced data really affects naive bayes.