Naive Bayes on DonorsChoose data set

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, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

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.

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 (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **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.

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]:

```
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline


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
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import cross_validation
```

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]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	pr
101880	5749	p096076	6eaa448903897a152320bd23a30147b2	Mrs.	CA	2016-01-05 00:00:00	Grades PreK-2	Ma
31477	47750	p185738	3afe10b996b7646d8641985a4b4b570d	Mrs.	UT	2016-01-05 01:05:00	Grades PreK-2	Ma

In [6]:

```
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[6]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

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/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
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", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "S
cience"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        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]))
```

1.3 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/47301924/4084039

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

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", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "S
cience"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        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/22898595/4084039
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]))
```

Preprocessing of `teacher_prefix`

In [9]:
```
#“Teacher prefix” data having the dots(.) and its has been observed the some rows are  empty in  this feature .
#the dot(.) and  empty row available in the data consider as float datatype and it does not
# accepted by the .Split() - Pandas function , so removing the same.
# cleaning has been done for the same following references are used
# 1.    Removing (.) from dataframe column - used ".str.replce" funtion (padas documentation)
# 2.    for  empty cell in datafram column - added the "Mrs." (in train data.cvs) which has me mostly occured in data
 set.

project_data["teacher_prefix_clean"] = project_data["teacher_prefix"].str.replace(".","")
project_data.head(2)
print(project_data.teacher_prefix_clean.shape)
```

```
(109248,)
```

1.4 Text preprocessing

In [10]:
```
# 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 [11]:
```
project_data.head(2)
```

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	pr
101880	5749	p096076	6eaa448903897a152320bd23a30147b2	Mrs.	CA	2016-01-05 00:00:00	Grades PreK-2	Ma Ma
31477	47750	p185738	3afe10b996b7646d8641985a4b4b570d	Mrs.	UT	2016-01-05 01:05:00	Grades PreK-2	Ma

1.4.1 Train , Cross Validation and Test Data Split

In [12]:

```
#As recommended in the Lecture video, splitting the Data in Train, Test and Cross validation data set
#before applying Vectorization to avoid the data leakage issues.
# As suggested to use stratify sampling, Referred following site for code
# https://stackoverflow.com/questions/29438265/stratified-train-test-split-in-scikit-learn


# split the data set into train and test
X_train, X_test, y_train, y_test = cross_validation.train_test_split(project_data, project_data['project_is_approved'
], test_size=0.3,stratify = project_data['project_is_approved'
])

# split the train data set into cross validation train and cross validation test
X_train, X_cv, y_train, y_cv = cross_validation.train_test_split(X_train, y_train, test_size=0.3,stratify=y_train)
```

In [13]:

```
#Removing the class label from the data set, in our case the class label is “project is approved”
#From all Train, Test and Cross validation data set

#Train Data

X_train.drop(['project_is_approved'] , axis = 1 , inplace =True)

#Test Data

X_test.drop(['project_is_approved'] , axis = 1 , inplace =True)

#Cross Validation data

X_cv.drop(['project_is_approved'] , axis = 1 , inplace =True)
```

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)
```

```
A typical day in our classroom is full of encouragement and exploration. With common core and being more open to allo
wing students to make more mistakes has helped me improve and see students thinking in a different way. My students a
re math problem solvers who are enjoying math.I have 28 first graders who want to be heard and understood. Who want t
o enjoy math. By creating and finding different math games that continues to help them build fluency and number sens
e, my students are enjoying and doing math at their own pace. They are enjoying what they are learning and want to pr
actice it in numerous ways.These materials will be used in math centers. Students will be able to explore and play ga
mes while practicing the skills they need. By playing these games they will be more engaged and will learn as they ga
in the skills they need to learn. They will practice learning their doubles, practice adding and subtracting and will
be able to have fun. I want to create an environment in which my students are loving what they are learning.I will be
able to use these donations for countless years. I will be able to use the dice in numerous games. I will be able to
provide some families with these games to try and continue to practice at home. I want to help my students in every w
ay possible.
==================================================
My students have learned what amazing adventures they can have when reading picture books; now I want them to realize
the endless possibilities for adventure that chapter books give!My students are all bright and inquisitive learners w
ho love to read! My students live in the heart of the city, most in extreme poverty. All students at my school receiv
e free breakfast and lunch; many families also utilize the food pantry that our runs out of its basement on the weeke
nd.My classroom library is filled with picture books. Having these chapter books would enable my students to build th
eir reading stamina within a book. They would be able to apply the many comprehension strategies and skills we've lea
rned to deeper text.Having these chapter books in our classroom library will help my students become VORACIOUS reader
s! This love of reading will continue in their lives as they continue to second grade and beyond.
==================================================
\"What are we working on today Mrs. Mistry?\" is typically the question I get asked as excited students walk into my
classroom ready to learn. The students at this school are so excited to walk into a room where they know they will ha
ve the chance to express themselves through art. \r\nThese K-5 suburban students are motivated to learn about anythin
g that is handed to them.   I enjoy supporting student learning with creative expression, and engagement in the art c
lassroom!Architecture is such an important part of my students lives right now. As our city is growing, students are
surrounded by tall buildings and construction. The materials for this  project will allow my students to immerse them
selves in something that connects art and what they are currently experiencing in their surroundings. \r\nThese growi
ng minds will definitely enjoy seeing their drawing on paper come to life ! First the students will learn beginning s
teps in constructing a building. Students will learn the process by creating blueprints of buildings, and use the mat
erials requested to create models of buildings that are imagined.nannan
==================================================
```

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[2000])
print(sent)
print("="*50)
```

```
My students attend a Title I school in downtown Oakland. Coming from diverse cultural/ethnic backgrounds and socioeco
nomically disadvantaged neighborhoods, these students know the meaning of perseverance & consistently give their best
in all that they do. They are inquisitive, enthusiastic, curious, eager to explore new things and ask compelling ques
tions. \r\nUnsatisfied by the cursory \"textbook\" explanation and uninterested in just memorizing the the correct fo
rmula to get a good grade, these students are always asking the how is and why is, thinking critically and analyzing
the information that is presented to them. As a result of their hard work, they have consistently exceeded district n
orms on standardized testing. Between Math and ELA, we had a total of 14 perfect scores in our class, last year, on t
he SBAC.\r\nChallenges we face at our school include having limited or outdated technological equipment and software,
no science lab, and little funding for resources beyond the basic educational supplies. These students will be contri
buting members of society one day. Sowing into them is sowing into tomorrow is leaders.During the time for the “curio
sity project,” students will be able to explore and research within a current unit/topic of study. Students will be l
ed by curiosity, ask inquiring questions, do online research, read e-books (reference materials), and make presentati
ons using the Amazon Kindle Fire. \r\nThis type of “inquiry-based” learning is aimed at sparking interest within stud
ents, encouraging them to initiate learning, giving them opportunity to pursue topics that fascinate them, teaching t
hem vital research online research and organizational skills, and challenging them to present information they have l
earned to the entire class in a compelling and insightful way. It allows students to learn material beyond what is pr
esented in class or in the textbook and is aligned with NGSS Science and Engineering Practices. Currently, our schoo
l’s technological equipment is outdated and extremely limited. With this project funded, my class will have close to
a 2:1 ratio of students to devices.nannan
==================================================
```

In [17]:

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

```
My students attend a Title I school in downtown Oakland. Coming from diverse cultural/ethnic backgrounds and socioeco
nomically disadvantaged neighborhoods, these students know the meaning of perseverance & consistently give their best
in all that they do. They are inquisitive, enthusiastic, curious, eager to explore new things and ask compelling ques
tions.   Unsatisfied by the cursory  textbook  explanation and uninterested in just memorizing the the correct formul
a to get a good grade, these students are always asking the how is and why is, thinking critically and analyzing the
information that is presented to them. As a result of their hard work, they have consistently exceeded district norms
on standardized testing. Between Math and ELA, we had a total of 14 perfect scores in our class, last year, on the SB
AC.  Challenges we face at our school include having limited or outdated technological equipment and software, no sci
ence lab, and little funding for resources beyond the basic educational supplies. These students will be contributing
members of society one day. Sowing into them is sowing into tomorrow is leaders.During the time for the “curiosity pr
oject,” students will be able to explore and research within a current unit/topic of study. Students will be led by c
uriosity, ask inquiring questions, do online research, read e-books (reference materials), and make presentations usi
ng the Amazon Kindle Fire.   This type of “inquiry-based” learning is aimed at sparking interest within students, enc
ouraging them to initiate learning, giving them opportunity to pursue topics that fascinate them, teaching them vital
research online research and organizational skills, and challenging them to present information they have learned to
the entire class in a compelling and insightful way. It allows students to learn material beyond what is presented in
class or in the textbook and is aligned with NGSS Science and Engineering Practices. Currently, our school’s technolo
gical equipment is outdated and extremely limited. With this project funded, my class will have close to a 2:1 ratio
of students to devices.nannan
```

In [18]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My students attend a Title I school in downtown Oakland Coming from diverse cultural ethnic backgrounds and socioecon
omically disadvantaged neighborhoods these students know the meaning of perseverance consistently give their best in
all that they do They are inquisitive enthusiastic curious eager to explore new things and ask compelling questions U
nsatisfied by the cursory textbook explanation and uninterested in just memorizing the the correct formula to get a g
ood grade these students are always asking the how is and why is thinking critically and analyzing the information th
at is presented to them As a result of their hard work they have consistently exceeded district norms on standardized
testing Between Math and ELA we had a total of 14 perfect scores in our class last year on the SBAC Challenges we fac
e at our school include having limited or outdated technological equipment and software no science lab and little fun
ding for resources beyond the basic educational supplies These students will be contributing members of society one d
ay Sowing into them is sowing into tomorrow is leaders During the time for the curiosity project students will be abl
e to explore and research within a current unit topic of study Students will be led by curiosity ask inquiring questi
ons do online research read e books reference materials and make presentations using the Amazon Kindle Fire This type
of inquiry based learning is aimed at sparking interest within students encouraging them to initiate learning giving
them opportunity to pursue topics that fascinate them teaching them vital research online research and organizational
skills and challenging them to present information they have learned to the entire class in a compelling and insightf
ul way It allows students to learn material beyond what is presented in class or in the textbook and is aligned with
NGSS Science and Engineering Practices Currently our school s technological equipment is outdated and extremely limit
ed With this project funded my class will have close to a 2 1 ratio of students to devices nannan
```

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're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'furthe
r',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'mor
e',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',
 \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "were
n't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

1.4.2 Train - Data Pracessing (Essay)

In [20]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['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_train.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████| 53531/53531 [00:26<00:00, 2011.17it/
s]
```

In [21]:
```
# after preprocesing
preprocessed_essays_train[1000]
```

Out[21]:
```
'students attend title 1 school county large population live poverty loving community parents teachers strive provide
children often things need education not affordable children love learn need every resource available prepare future
deserve best want experience educational opportunities variety avenues technology key success students kindergarten l
earning read looking pictures reading sight words retelling read love read love without help even kindle tablets smal
l group reading centers phenomenal students learning experience children learn use technology quickly today world wan
t students opportunity grow using technology kindles used read students play learning games well help students use ap
ps record progress work nannan'
```

1.4.3 Cross Validation - Data Pracessing (Essay)

```
In [22]: preprocessed_essays_cv = []
         # tqdm is for printing the status bar
         for sentence in tqdm(X_cv['essay'].values):
             sent = decontracted(sentence)
             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_cv.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 22942/22942 [00:11<00:00, 2007.17it/
s]
```

```
In [23]: # after preprocesing
         preprocessed_essays_cv[1000]
```

```
Out[23]: 'physics one fields science requires hands experimentation understand material undertaking renovation curriculum invo
         lve experiments projects end course students extensive experience experimentation teach public school low economicall
         y depressed area city student body 34 african american 35 hispanic 21 white 9 school small caring strive educate stud
         ents best ability guide life much students enrolled physics come walks life many looking go college many students fir
         st family go college preparing college life one main objectives students spend three weeks performing basic current v
         oltage measurements using circuits boards learn relationship described ohm law circuits parallel series also investig
         ate capacitors work donators enable improve classroom provide experimental experiences students goals foster apprecia
         te sciences general physics particular change class curriculum believe students ready go college appreciation science
         enter scientific field excel endeavors achieve dreams'
```

1.4.4 Test - Data Pracessing (Essay)

```
In [24]: preprocessed_essays_test = []
         # tqdm is for printing the status bar
         for sentence in tqdm(X_test['essay'].values):
             sent = decontracted(sentence)
             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_test.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 32775/32775 [00:16<00:00, 2012.75it/
s]
```

```
In [25]: # after preprocesing
         preprocessed_essays_test[5]
```

```
Out[25]: 'students work often considered risk generally speaking behaviors characteristics associated risk student based resea
         rch observable patterns student demographics school performance frequently sent alternative school due behavior probl
         ems disciplinary action numerous academic studies demonstrated correlations certain risk factors student likelihood s
         ucceeding academically graduating high school pursuing postsecondary education goal classroom develop strategies aime
         d identifying student risk factors intervening assistance support intended help risk students succeed academically co
         mplete school students work often considered risk generally speaking behaviors characteristics associated risk studen
         t based research observable patterns student demographics school performance frequently sent alternative school due b
         ehavior problems disciplinary action numerous academic studies demonstrated correlations certain risk factors student
         likelihood succeeding academically graduating high school pursuing postsecondary education goal classroom develop str
         ategies aimed identifying student risk factors intervening assistance support intended help risk students succeed aca
         demically complete school nannan'
```

1.5 Preprocessing of `project_title`

```
In [26]: # Data processing for project titles
         Title_clean = project_data.project_title
         Title_clean.head(2)
```

```
Out[26]: 101880    Math Madness
         31477     Math is Fun!
         Name: project_title, dtype: object
```

```
In [27]: P = decontracted(project_data['project_title'].values[1])
         print(P)
```

```
         Math is Fun!
```

In [28]:

```
# \r \n \t  and -- remove from string python: http://texthandler.com/info/remove-line-breaks-python/
P = P.replace('\\r', ' ')
P = P.replace('\\"', ' ')
P = P.replace('\\n', ' ')
P = P.replace('--', ' ')
print(P)
```

```
Math is Fun!
```

1.5.1 Train - Data Pracessing (Project Title)

In [29]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles_train = []
# tqdm is for printing the status bar
for Pance in tqdm(X_train['project_title'].values):
    P = decontracted(Pance)
    P = P.replace('\\r', ' ')
    P = P.replace('\\"', ' ')
    P = P.replace('\\n', ' ')
    P = re.sub('[^A-Za-z0-9]+', ' ', P)
    # https://gist.github.com/sebleier/554280
    P = ' '.join(e for e in P.split() if e not in stopwords)
    preprocessed_titles_train.append(P.lower().strip())
```

```
100%|█████████████████████████████████████████████████████████████████████████| 53531/53531 [00:01<00:00, 44652.16it/
s]
```

1.5.2 Cross Validation - Data Pracessing (Project Title)

In [30]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles_cv = []
# tqdm is for printing the status bar
for Pance in tqdm(X_cv['project_title'].values):
    P = decontracted(Pance)
    P = P.replace('\\r', ' ')
    P = P.replace('\\"', ' ')
    P = P.replace('\\n', ' ')
    P = re.sub('[^A-Za-z0-9]+', ' ', P)
    # https://gist.github.com/sebleier/554280
    P = ' '.join(e for e in P.split() if e not in stopwords)
    preprocessed_titles_cv.append(P.lower().strip())
```

```
100%|█████████████████████████████████████████████████████████████████████████| 22942/22942 [00:00<00:00, 44739.77it/
s]
```

1.5.3 Test - Data Pracessing (Project Title)

In [31]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles_test = []
# tqdm is for printing the status bar
for Pance in tqdm(X_test['project_title'].values):
    P = decontracted(Pance)
    P = P.replace('\\r', ' ')
    P = P.replace('\\"', ' ')
    P = P.replace('\\n', ' ')
    P = re.sub('[^A-Za-z0-9]+', ' ', P)
    # https://gist.github.com/sebleier/554280
    P = ' '.join(e for e in P.split() if e not in stopwords)
    preprocessed_titles_test.append(P.lower().strip())
```

```
100%|█████████████████████████████████████████████████████████████████████████| 32775/32775 [00:00<00:00, 44684.57it/
s]
```

1.6 Preparing data for models

In [32]:

```
project_data.columns
```

Out[32]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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',
       'clean_categories', 'clean_subcategories', 'teacher_prefix_clean',
       'essay'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

1.6.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/ (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)

Project_categories - Vectorization

In [33]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_Cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)

vectorizer_Cat.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer_Cat.transform(X_train['clean_categories'].values)
categories_one_hot_cv = vectorizer_Cat.transform(X_cv['clean_categories'].values)
categories_one_hot_test = vectorizer_Cat.transform(X_test['clean_categories'].values)

print(vectorizer_Cat.get_feature_names())

print("Shape of matrix after one hot encodig ",categories_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",categories_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",categories_one_hot_test.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_S
cience', 'Literacy_Language']
Shape of matrix after one hot encodig  (53531, 9)
Shape of matrix after one hot encodig  (22942, 9)
Shape of matrix after one hot encodig  (32775, 9)
```

Project_sub_categories - Vectorization

In [34]:

```
# we use count vectorizer to convert the values into one
vectorizer_sub_cat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)

vectorizer_sub_cat.fit(X_train['clean_subcategories'].values)

sub_categories_one_hot_train = vectorizer_sub_cat.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer_sub_cat.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer_sub_cat.transform(X_test['clean_subcategories'].values)

print(vectorizer_sub_cat.get_feature_names())

print("Shape of matrix after one hot encodig ",sub_categories_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_test.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government',
'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEduc
ation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelo
pment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNee
ds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (53531, 30)
Shape of matrix after one hot encodig  (22942, 30)
Shape of matrix after one hot encodig  (32775, 30)
```

School_State - Vectorization

In [35]:

```
# we use count vectorizer to convert the values into one hot encoded features
from collections import Counter
my_counter_state = Counter()
for word in project_data['school_state'].values:
    my_counter_state.update(word.split())

state_dict = dict(my_counter_state)
sorted_state_dict = dict(sorted(state_dict.items(), key=lambda kv: kv[1]))

vectorizer_state = CountVectorizer(vocabulary=list(sorted_state_dict.keys()), lowercase=False, binary=True)

vectorizer_state.fit(X_train['school_state'].values)

school_state_one_hot_train = vectorizer_state.transform(X_train['school_state'].values)
school_state_one_hot_cv = vectorizer_state.transform(X_cv['school_state'].values)
school_state_one_hot_test = vectorizer_state.transform(X_test['school_state'].values)

print(vectorizer_state.get_feature_names())


print("Shape of matrix after one hot encodig ",school_state_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",school_state_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",school_state_one_hot_test.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'C
O', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'MA', 'LA', 'OH',
'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX', 'CA']
Shape of matrix after one hot encodig  (53531, 51)
Shape of matrix after one hot encodig  (22942, 51)
Shape of matrix after one hot encodig  (32775, 51)
```

teacher_prefix - Vectorization

In [36]:

```
#“Teacher prefix” data having the dots(.) and its has been observed the some rows are  empty in  this feature .
#the dot(.) and  empty row available in the data consider as float datatype and it does not
# accepted by the .Split() – Pandas function , so removing the same.
# cleaning has been done for the same following references are used
# 1.    Removing (.) from dataframe column - used ".str.replce" funtion (padas documentation)
# 2.    for  empty cell in datafram column - added the "Mrs." (in train data.cvs) which has me mostly occured in data
 set.

project_data["teacher_prefix_clean"] = project_data["teacher_prefix"].str.replace(".","")
project_data.head(2)
print(project_data.teacher_prefix_clean.shape)
```

```
(109248,)
```

In [37]:

```
from collections import Counter
my_counter_T = Counter()
for word in project_data["teacher_prefix_clean"].values:

        my_counter_T.update(word.split())


Teacher_dict = dict(my_counter_T)
sorted_Teacher_dict = dict(sorted(Teacher_dict.items(), key=lambda kv: kv[1]))

vectorizer_teacher = CountVectorizer(vocabulary=list(Teacher_dict.keys()), lowercase=False, binary=True)
#vectorizer.fit(project_data.teacher_prefix_clean.values)


vectorizer_teacher.fit(X_train["teacher_prefix_clean"].values)
print(vectorizer_teacher.get_feature_names())

Teacher_Prefix_one_hot_train = vectorizer_teacher.transform(X_train["teacher_prefix_clean"].values)
Teacher_Prefix_one_hot_cv = vectorizer_teacher.transform(X_cv["teacher_prefix_clean"].values)
Teacher_Prefix_one_hot_test = vectorizer_teacher.transform(X_test["teacher_prefix_clean"].values)

print("Shape of matrix after one hot encodig ",Teacher_Prefix_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",Teacher_Prefix_one_hot_cv.shape)
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  (53531, 5)
Shape of matrix after one hot encodig  (22942, 5)
Shape of matrix after one hot encodig  (32775, 5)
```

project_grade_category - Vectorization

In [38]:
```
# Used this as reference to avoide the space between grades and category ,
# it has split the string with comma , now getting four project grade category as required.
# https://stackoverflow.com/questions/4071396/split-by-comma-and-strip-whitespace-in-python
from collections import Counter
my_counter_project_grade_category= Counter()
for word in project_data['project_grade_category'].values:
    my_counter_project_grade_category.update(word.split(','))

project_grade_category_dict = dict(my_counter_project_grade_category)
sorted_project_grade_category_prefix_dict = dict(sorted(project_grade_category_dict.items(), key=lambda kv: kv[1]))

vectorizer_grade = CountVectorizer(vocabulary=list(project_grade_category_dict.keys()), lowercase=False, binary=True)

vectorizer_grade.fit(X_train["project_grade_category"].values)
print(vectorizer_grade.get_feature_names())

project_grade_category_one_hot_train = vectorizer_grade.transform(X_train["project_grade_category"].values)
project_grade_category_one_hot_cv = vectorizer_grade.transform(X_cv["project_grade_category"].values)
project_grade_category_one_hot_test = vectorizer_grade.transform(X_test["project_grade_category"].values)


print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_train.shape)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_cv.shape)
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_test.shape)
```

```
['Grades PreK-2', 'Grades 9-12', 'Grades 6-8', 'Grades 3-5']
Shape of matrix after one hot encodig  (53531, 4)
Shape of matrix after one hot encodig  (22942, 4)
Shape of matrix after one hot encodig  (32775, 4)
```

1.6.2 Vectorizing Text data

1.6.2.1 Bag of words

Train Data Vectorization - BOW (essays)

In [39]:
```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(preprocessed_essays_train)
bow_essays_train = vectorizer_bow_essay.fit_transform(preprocessed_essays_train)
print("Shape of matrix after one hot encodig ",bow_essays_train.shape)
```

```
Shape of matrix after one hot encodig  (53531, 12433)
```

CV Data Vectorization - BOW (essays)

In [40]:
```
bow_essays_cv = vectorizer_bow_essay.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encodig ",bow_essays_cv.shape)
```

```
Shape of matrix after one hot encodig  (22942, 12433)
```

Test Data Vectorization - BOW (essays)

In [41]:
```
bow_essays_test = vectorizer_bow_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",bow_essays_test.shape)
```

```
Shape of matrix after one hot encoding  (32775, 12433)
```

Train Data Vectorization - BOW (Project Titles)

In [42]:
```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_title = CountVectorizer(min_df=10)
bow_title_train = vectorizer_bow_title.fit_transform(preprocessed_titles_train)
print("Shape of matrix after one hot encodig ",bow_title_train.shape)
```

```
Shape of matrix after one hot encodig  (53531, 2187)
```

CV Data Vectorization - BOW (Project Titles)

In [43]:
```
bow_title_cv = vectorizer_bow_title.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encodig ",bow_title_cv.shape)
```

```
Shape of matrix after one hot encodig  (22942, 2187)
```

Test Data Vectorization - BOW (Project Titles)

In [44]:
```
bow_title_test = vectorizer_bow_title.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encodig ",bow_title_test.shape)
```

```
Shape of matrix after one hot encodig  (32775, 2187)
```

1.6.2.2 TFIDF vectorizer

Train Data Vectorization - TFIDF (essays)

In [45]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
tfidf_essays_train = vectorizer_tfidf_essay.fit_transform(preprocessed_essays_train)
print("Shape of matrix after one hot encodig ",tfidf_essays_train.shape)
```

```
Shape of matrix after one hot encodig  (53531, 12433)
```

CV Data Vectorization - TFIDF (essays)

In [46]:
```
tfidf_essays_cv = vectorizer_tfidf_essay.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encodig ",tfidf_essays_cv.shape)
```

```
Shape of matrix after one hot encodig  (22942, 12433)
```

Test Data Vectorization - TFIDF (essays)

In [47]:
```
tfidf_essays_test = vectorizer_tfidf_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encodig ",tfidf_essays_test.shape)
```

```
Shape of matrix after one hot encodig  (32775, 12433)
```

Train Data Vectorization - TFIDF (Project Titles)

In [48]:
```
vectorizer_tfidf_title = CountVectorizer(min_df=10)
tfidf_title_train = vectorizer_tfidf_title.fit_transform(preprocessed_titles_train)
print("Shape of matrix after one hot encodig ",bow_title_train.shape)
```

```
Shape of matrix after one hot encodig  (53531, 2187)
```

CV Data Vectorization - TFIDF (Project Titles)

In [49]:
```
tfidf_title_cv = vectorizer_tfidf_title.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encodig ",bow_title_cv.shape)
```

```
Shape of matrix after one hot encodig  (22942, 2187)
```

Test Data Vectorization - TFIDF (Project Titles)

In [50]:
```
tfidf_title_test = vectorizer_tfidf_title.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encodig ",bow_title_test.shape)
```

```
Shape of matrix after one hot encodig  (32775, 2187)
```

1.6.3 Vectorizing Numerical features

1.6.3.1 Vectorizing Numerical features - Price

In [51]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()

# Merging the project data train , Cv , test with price from resource data
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
```

In [52]:
```
X_train.head(2)
```

Out[52]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_
0	22883	p170463	37847ef88d0a5db6ea0e535cdcf640b0	Mrs.	OH	2016-11-28 17:05:00	Grades PreK-2	Eager Learners Need Wor Work Manipulat
1	164154	p187013	39d9155e3783ead4da6759c1fb982282	Ms.	IA	2017-04-24 22:23:00	Grades 3-5	Wanted: Class Rot

In [53]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(X_train['price'].values.reshape(-1,1))

price_train = normalizer.transform(X_train['price'].values.reshape(-1,1))
price_cv = normalizer.transform(X_cv['price'].values.reshape(-1,1))
price_test = normalizer.transform(X_test['price'].values.reshape(-1,1))

print(price_train.shape)
print(price_cv.shape)
print(price_test.shape)
```

```
(53531, 1)
(22942, 1)
(32775, 1)
```

1.6.3.2 Vectorizing Numerical features - teacher_number_of_previously_posted_projects

In [54]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

prev_post_train = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_post_cv = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_post_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print(prev_post_train.shape)
print(prev_post_cv.shape)
print(prev_post_test.shape)
```

```
(53531, 1)
(22942, 1)
(32775, 1)
```

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

2. **The hyper paramter tuning(find best Alpha)**
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Feature importance**
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of \`feature_log_prob_\` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names

4. **Representation of results**
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
    - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
    - Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

      (https://seaborn.pydata.org/generated/seaborn.heatmap.html)
      (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

   (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

   (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

5. **Conclusion** (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

   (https://seaborn.pydata.org/generated/seaborn.heatmap.html)
    - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

2. Naive Bayes

2.1 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

In [55]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code

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

2.1.1 Applying Naive Bayes on BOW, SET 1

In [56]:

```
from scipy.sparse import hstack
X_train_bow = hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one_hot_train ,Teacher_Prefix
_one_hot_train,project_grade_category_one_hot_train,bow_essays_train,bow_title_train,price_train,prev_post_train)).toc
sr()
X_train_bow.shape
```

Out[56]:

```
(53531, 14721)
```

In [57]:

```
X_cv_bow = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_one_hot_cv ,Teacher_Prefix_one_hot_cv,
project_grade_category_one_hot_cv,bow_essays_cv,bow_title_cv,price_cv,prev_post_cv)).tocsr()
X_cv_bow.shape
```

Out[57]:

```
(22942, 14721)
```

In [58]:

```
X_test_bow = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hot_test ,Teacher_Prefix_one
_hot_test,project_grade_category_one_hot_test,bow_essays_test,bow_title_test,price_test,prev_post_test)).tocsr()

X_test_bow.shape
```

Out[58]:

```
(32775, 14721)
```

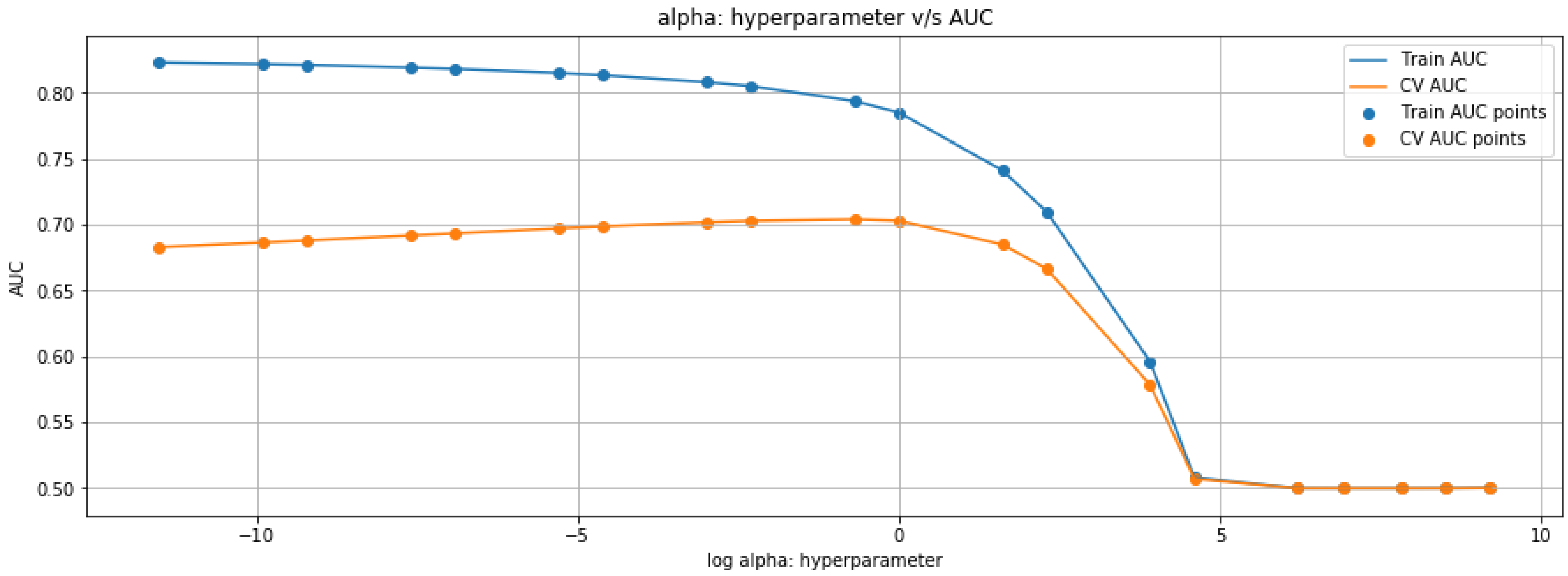
Finding the best hyper parameter That maximum AUC value

In [59]:

```
from sklearn.metrics import roc_auc_score
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [61]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_alphas = []
# considering the hyperparameter (alpha) values from 10-4  to 10 4  as suggested in NB lecture video
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 50
00, 10000]
for i in alphas:
        nb = MultinomialNB(alpha = i)
        nb.fit(X_train_bow, y_train)
        y_train_pred = batch_predict(nb, X_train_bow)
        y_cv_pred = batch_predict(nb, X_cv_bow)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
# Plotting alpha again AUC making graph not intuitive to taken log of alpha and plotting against AUC.
#https://stackoverflow.com/questions/11656767/how-to-take-the-log-of-all-elements-of-a-list

for P in alphas:
    T = math.log(P)
    log_alphas.append(T)


plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')
plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.rcParams["figure.figsize"] = [16,9]
plt.show()
```



In [62]:
```
#http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
x = PrettyTable()
column_names = ['alphas', 'log_alphas', 'train_auc', 'cv_auc']
x.add_column(column_names[0],alphas)
x.add_column(column_names[1],log_alphas)
x.add_column(column_names[2],train_auc)
x.add_column(column_names[3],cv_auc)
print(x)
```

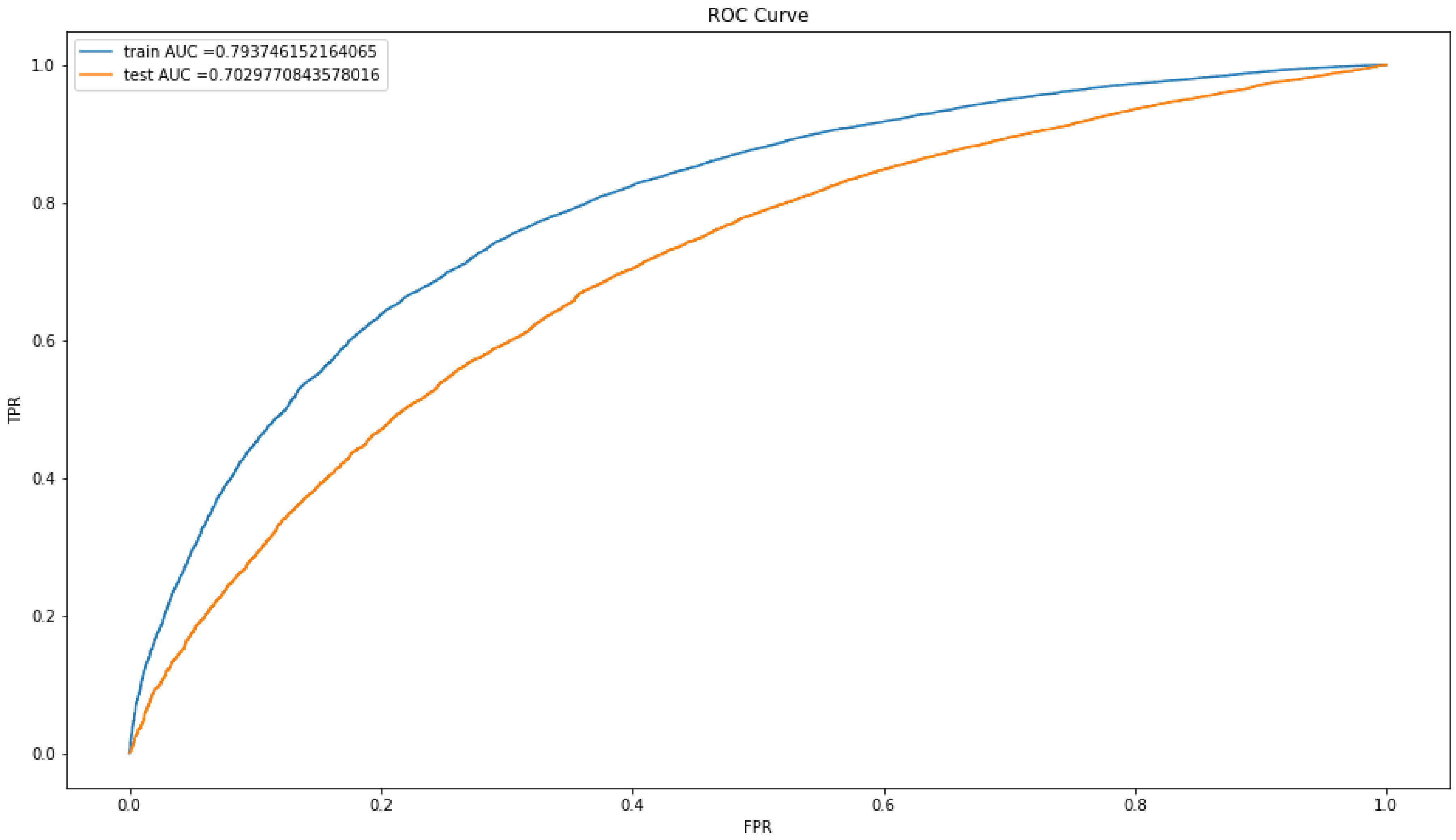
```
+--------+---------------------+--------------------+---------------------+
| alphas |      log_alphas     |      train_auc     |        cv_auc       |
+--------+---------------------+--------------------+---------------------+
| 1e-05  | -11.512925464970229 | 0.8229745264603592 |  0.6830547559320883 |
| 5e-05  |  -9.903487552536127 | 0.8218146790138556 |  0.6865350401864021 |
| 0.0001 |  -9.210340371976182 | 0.8211715426134003 |  0.6881151984763624 |
| 0.0005 |  -7.600902459542082 | 0.8192509946215378 |  0.6918594412761139 |
| 0.001  |  -6.907755278982137 | 0.8182016074138976 |  0.6935006433065425 |
| 0.005  |  -5.298317366548036 | 0.8150906357100958 |  0.6972177391261558 |
|  0.01  |  -4.605170185988091 | 0.8133795612787336 |  0.6987418735012235 |
|  0.05  |  -2.995732273553991 | 0.8081538867111816 |  0.7018453011889432 |
|  0.1   | -2.3025850929940455 | 0.8050868910512322 |  0.7029465503758644 |
|  0.5   | -0.6931471805599453 | 0.793746152164065  |  0.7042067188125261 |
|   1    |         0.0         | 0.7849773789957366 |  0.7030619250414509 |
|   5    |  1.6094379124341003 | 0.7414839661812245 |  0.6850531566260101 |
|   10   |  2.302585092994046  | 0.7099051781323114 |  0.6665517134002816 |
|   50   |  3.912023005428146  | 0.5961697045065707 |  0.5784338061402802 |
|  100   |  4.605170185988092  | 0.5082150378840132 |  0.5072621202986192 |
|  500   |  6.214608098422191  | 0.5000506847603194 | 0.49997431682761456 |
|  1000  |  6.907755278982137  |        0.5         |         0.5         |
|  2500  |  7.824046010856292  |        0.5         |         0.5         |
|  5000  |  8.517193191416238  | 0.5000616903146207 | 0.49997431682761456 |
| 10000  |  9.210340371976184  | 0.5002514016803786 |  0.5002263948727577 |
+--------+---------------------+--------------------+---------------------+
```

Using Best K Value – Training the Model

In [63]:
```
#By observing the plot and the tables for alpha (hyperparameter) ,
#Train and Test AUC. IT has been noted the for alpha value 0.5, there is optimal values of Train and Test AUC can be o
btained,
#so taking Optimal alpha as 0.5

optimal_alpha_bow = 0.5
```

In [64]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


nb_bow = MultinomialNB(alpha = optimal_alpha_bow)
nb_bow.fit(X_train_bow, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(nb_bow, X_train_bow)
y_test_pred = batch_predict(nb_bow, X_test_bow)

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

In [65]:
```
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.show()
```



Confusion Matrix

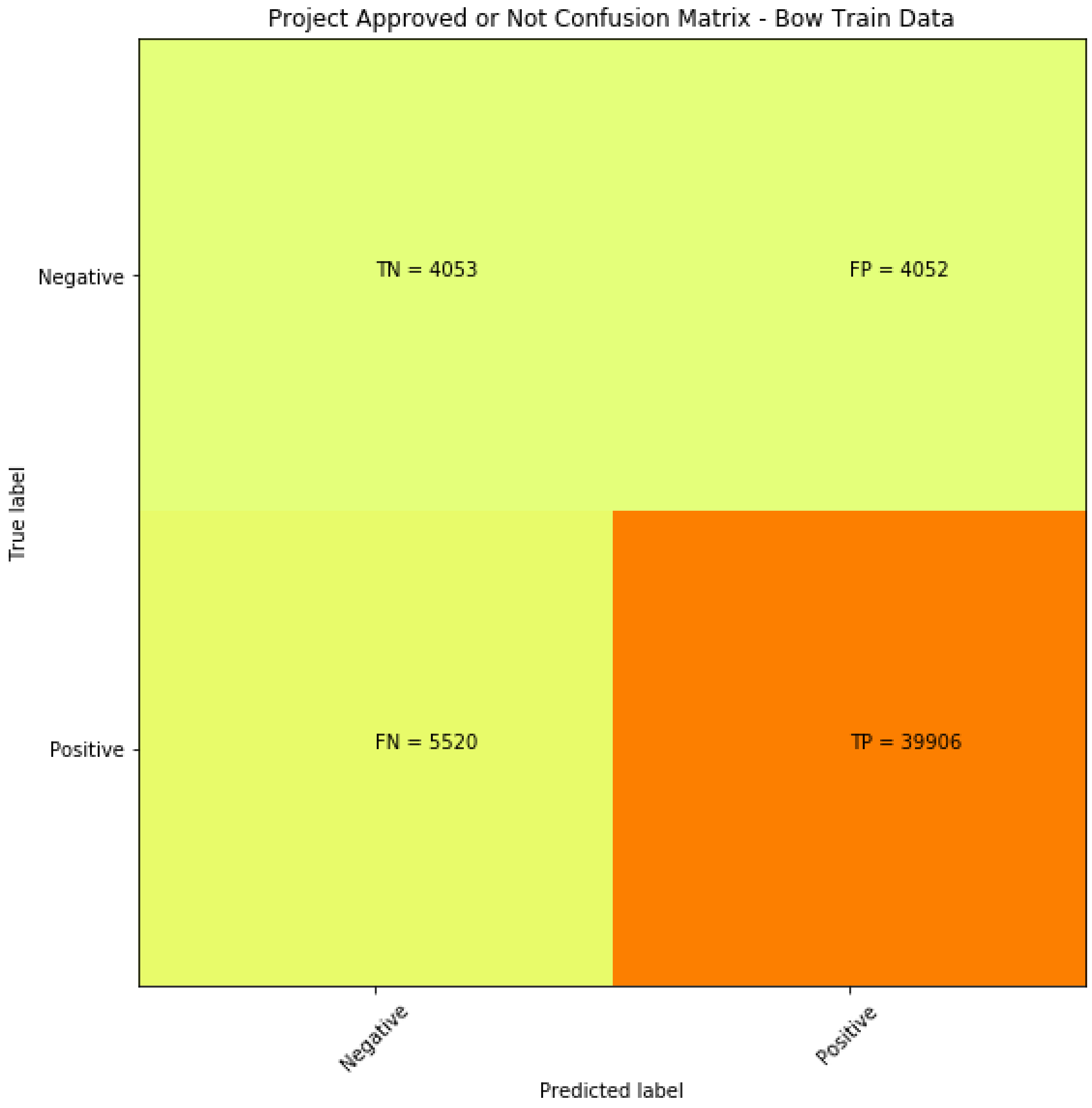
Train confusion matrix

In [66]:
```
#https://stackoverflow.com/questions/28719067/roc-curve-and-cut-off-point-python

def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(fpr*(1-tpr))]
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [67]:
```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
bow_train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
print(bow_train_confusion_matrix)
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999961943051 for threshold 0.12
[[ 4053  4052]
 [ 5520 39906]]
```

In [68]:
```
#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(bow_train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative','Positive']
plt.title('Project Approved or Not Confusion Matrix - Bow Train Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN','FP'], ['FN', 'TP']]
for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(bow_train_confusion_matrix[i][j]))
plt.show()
```

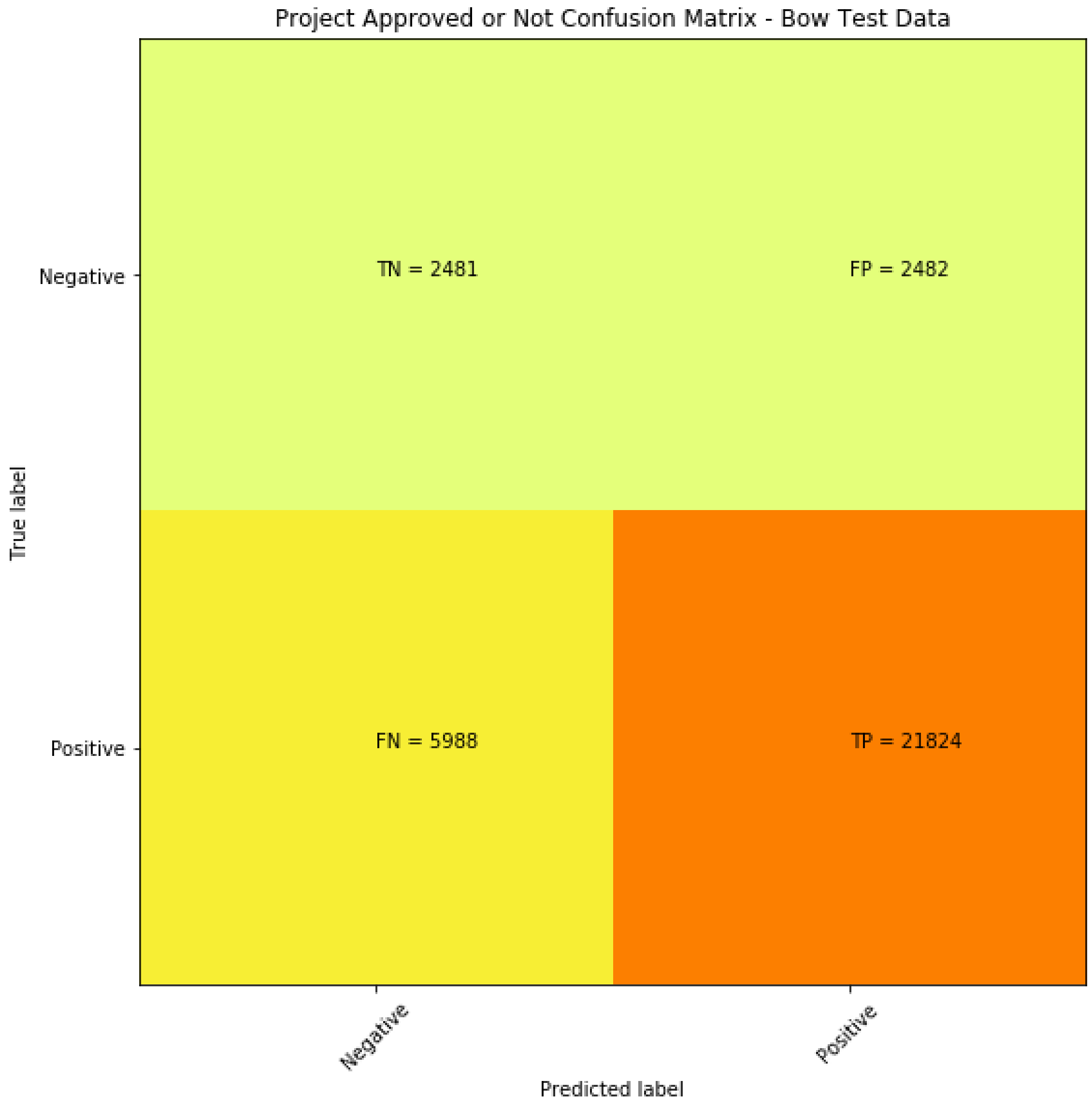


Test confusion matrix

In [69]:
```
print("Train confusion matrix")
bow_test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_fpr))
print(bow_test_confusion_matrix)
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999998985034083 for threshold 0.526
[[ 2481  2482]
 [ 5988 21824]]
```

```
In [70]: ##http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

         plt.clf()
         plt.imshow(bow_test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
         classNames = ['Negative','Positive']
         plt.title('Project Approved or Not Confusion Matrix - Bow Test Data')
         plt.ylabel('True label')
         plt.xlabel('Predicted label')
         tick_marks = np.arange(len(classNames))
         plt.xticks(tick_marks, classNames, rotation=45)
         plt.yticks(tick_marks, classNames)
         s = [['TN','FP'], ['FN', 'TP']]
         for i in range(2):
             for j in range(2):
                 plt.text(j,i, str(s[i][j])+" = "+str(bow_test_confusion_matrix[i][j]))
         plt.show()
```



2.1.2 Applying Naive Bayes on TFIDF, SET 2

```
In [71]: X_train_tfidf = hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one_hot_train ,Teacher_Pref
         ix_one_hot_train,project_grade_category_one_hot_train,tfidf_essays_train,tfidf_title_train,price_train,prev_post_train
         )).tocsr()
         X_train_tfidf.shape

Out[71]: (53531, 14721)
```

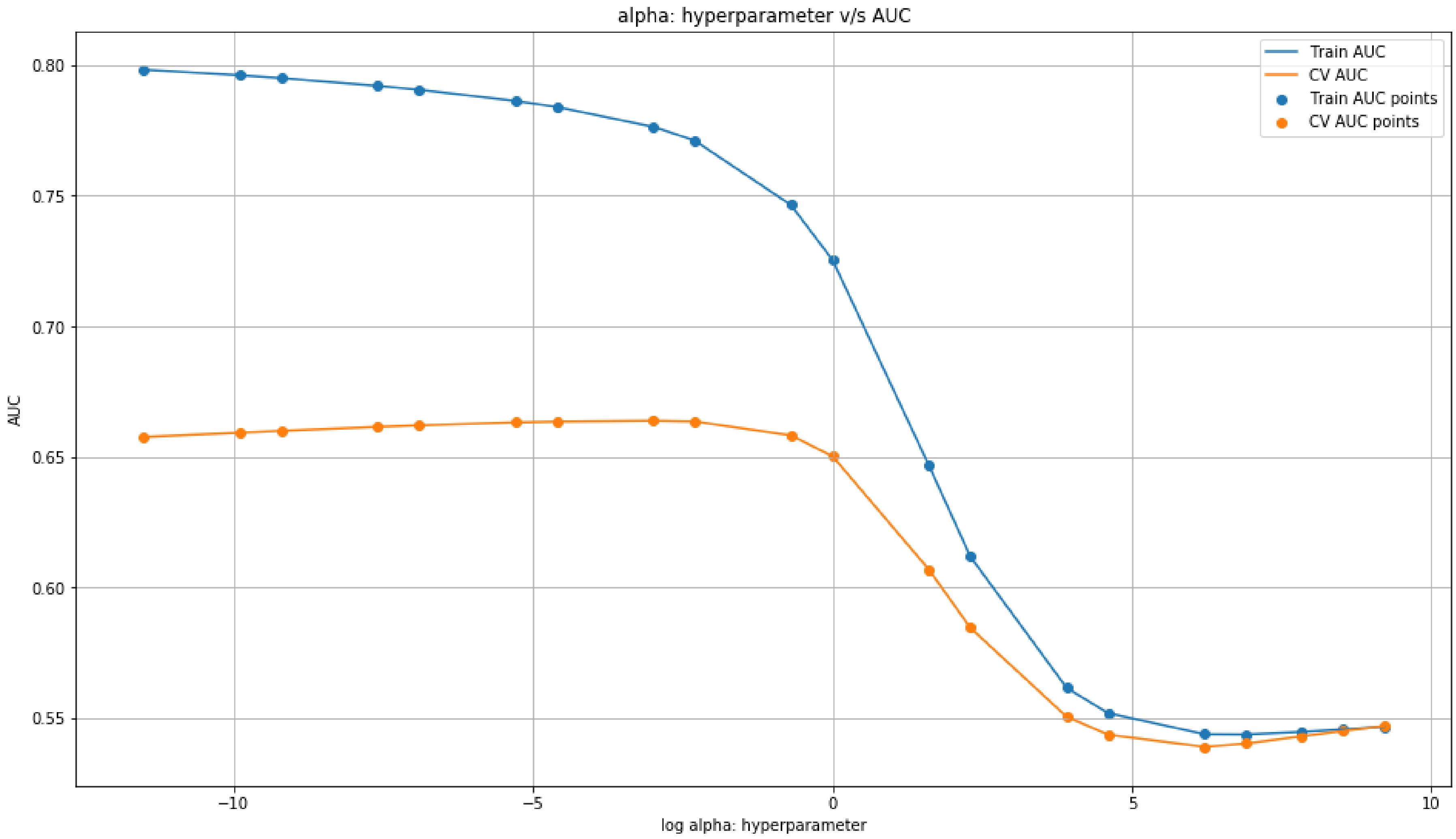
```
In [72]: X_cv_tfidf = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_one_hot_cv ,Teacher_Prefix_one_hot_c
         v,project_grade_category_one_hot_cv,tfidf_essays_cv,tfidf_title_cv,price_cv,prev_post_cv)).tocsr()
         X_cv_tfidf.shape

Out[72]: (22942, 14721)
```

```
In [73]: X_test_tfidf = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hot_test ,Teacher_Prefix_o
         ne_hot_test,project_grade_category_one_hot_test,tfidf_essays_test,tfidf_title_test,price_test,prev_post_test)).tocsr()
         X_test_tfidf.shape

Out[73]: (32775, 14721)
```

Finding the best hyper parameter That maximum AUC value

In [74]:

```
train_auc = []
cv_auc = []
log_alphas = []
# considering the hyperparameter (alpha) values from 10-4  to 10 4  as suggested in NB lecture video
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 50
00, 10000]
for i in alphas:
        nb = MultinomialNB(alpha = i)
        nb.fit(X_train_tfidf, y_train)
        y_train_pred = batch_predict(nb, X_train_tfidf)
        y_cv_pred = batch_predict(nb, X_cv_tfidf)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
# Plotting alpha again AUC making graph not intuitive to taken log of alpha and plotting against AUC.
#https://stackoverflow.com/questions/11656767/how-to-take-the-log-of-all-elements-of-a-list

for P in alphas:
    T = math.log(P)
    log_alphas.append(T)


plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')
plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.rcParams["figure.figsize"] = [20,10]
plt.show()
```



In [75]:

```
#http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
x = PrettyTable()
column_names = ['alphas', 'log_alphas', 'train_auc', 'cv_auc']
x.add_column(column_names[0],alphas)
x.add_column(column_names[1],log_alphas)
x.add_column(column_names[2],train_auc)
x.add_column(column_names[3],cv_auc)
print(x)
```

```
+--------+---------------------+--------------------+--------------------+
| alphas |      log_alphas     |     train_auc      |       cv_auc       |
+--------+---------------------+--------------------+--------------------+
| 1e-05  | -11.512925464970229 | 0.7981452666895414 | 0.6575126339916387 |
| 5e-05  |  -9.903487552536127 | 0.7960083856782972 | 0.6591541982775213 |
| 0.0001 |  -9.210340371976182 | 0.7949231163438375 | 0.6598618901821262 |
| 0.0005 |  -7.600902459542082 | 0.7919504813612708 | 0.6613925007975534 |
| 0.001  |  -6.907755278982137 | 0.790438719636845  | 0.661979428562574  |
| 0.005  |  -5.298317366548036 | 0.7861768350301904 | 0.6630955893668532 |
|  0.01  |  -4.605170185988091 | 0.7838762939301083 | 0.6634339803186169 |
|  0.05  |  -2.995732273553991 | 0.7762692015619739 | 0.6637563285288501 |
|  0.1   | -2.3025850929940455 | 0.770997978340515  | 0.6634595156316334 |
|  0.5   | -0.6931471805599453 | 0.7463198588898899 | 0.6581360608418829 |
|   1    |         0.0         | 0.7252839749432971 | 0.6501670470792511 |
|   5    |  1.6094379124341003 | 0.6466557184216437 | 0.6067857292406333 |
|   10   |  2.302585092994046  | 0.6117354857394552 | 0.5844673185845388 |
|   50   |  3.912023005428146  | 0.5614389075080668 | 0.5503904655429118 |
|  100   |  4.605170185988092  | 0.5518028602653398 | 0.5435587047235391 |
|  500   |  6.214608098422191  | 0.5437410622310046 | 0.5389221380251832 |
|  1000  |  6.907755278982137  | 0.5436588397130918 | 0.5401701302428124 |
|  2500  |  7.824046010856292  | 0.5446188516073474 | 0.5429071106635113 |
|  5000  |  8.517193191416238  | 0.5455803166041574 | 0.5449490485486184 |
| 10000  |  9.210340371976184  | 0.5465719735411483 | 0.546837404907204  |
+--------+---------------------+--------------------+--------------------+
```

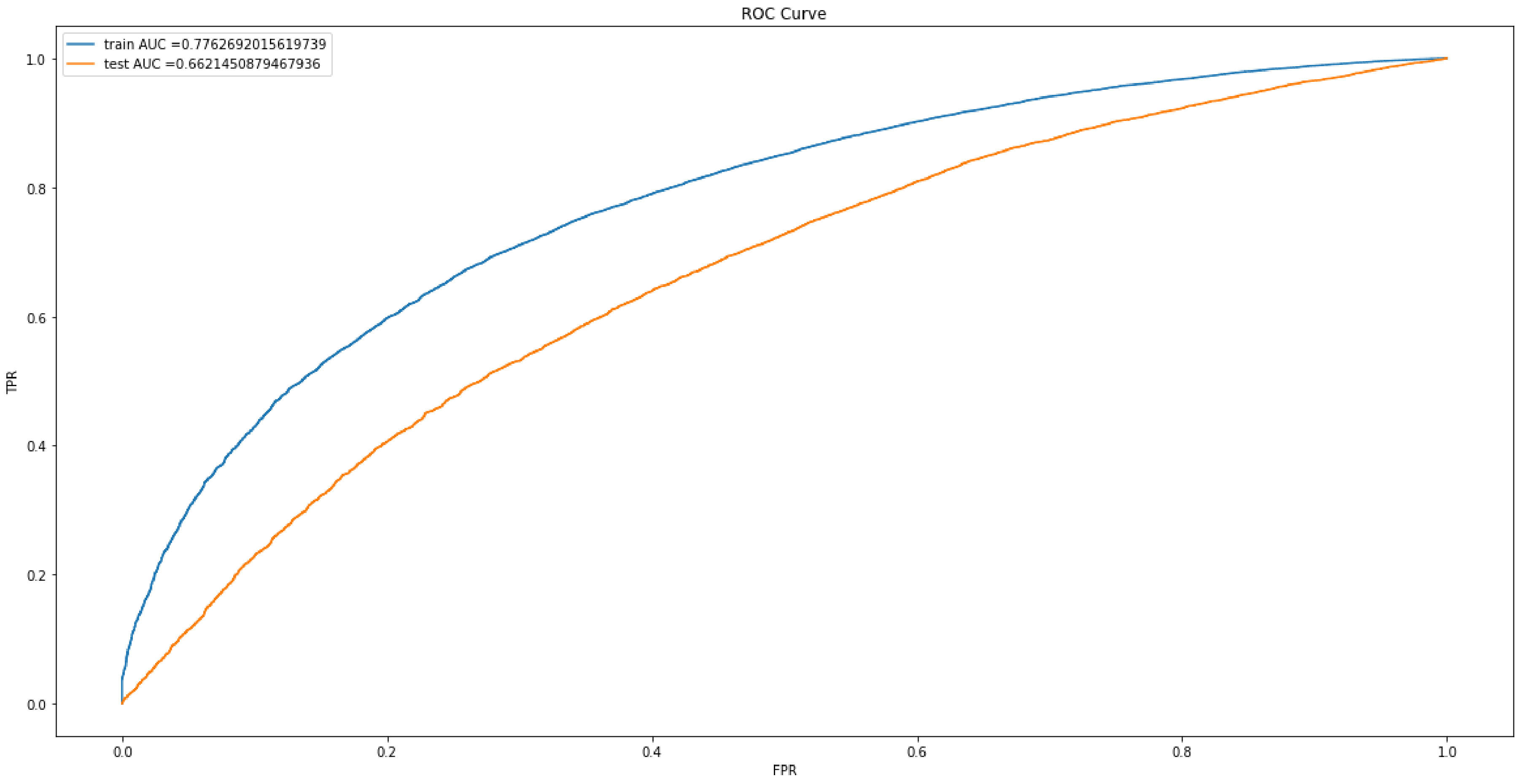
Using Best K Value – Training the Model

In [76]:

```
#By observing the plot and the tables for alpha (hyperparameter) ,
#Train and Test AUC. IT has been noted the for alpha value 0.05, there is optimal values of Train and Test AUC can be
 obtained,
#so taking Optimal alpha as 0.05

optimal_alpha_tfidf  = 0.05
```

In [77]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


nb_tfidf = MultinomialNB(alpha = optimal_alpha_tfidf)
nb_tfidf.fit(X_train_tfidf, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(nb_tfidf, X_train_tfidf)
y_test_pred = batch_predict(nb_tfidf, X_test_tfidf)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.show()
```



Confusion Matrix

Train confusion matrix

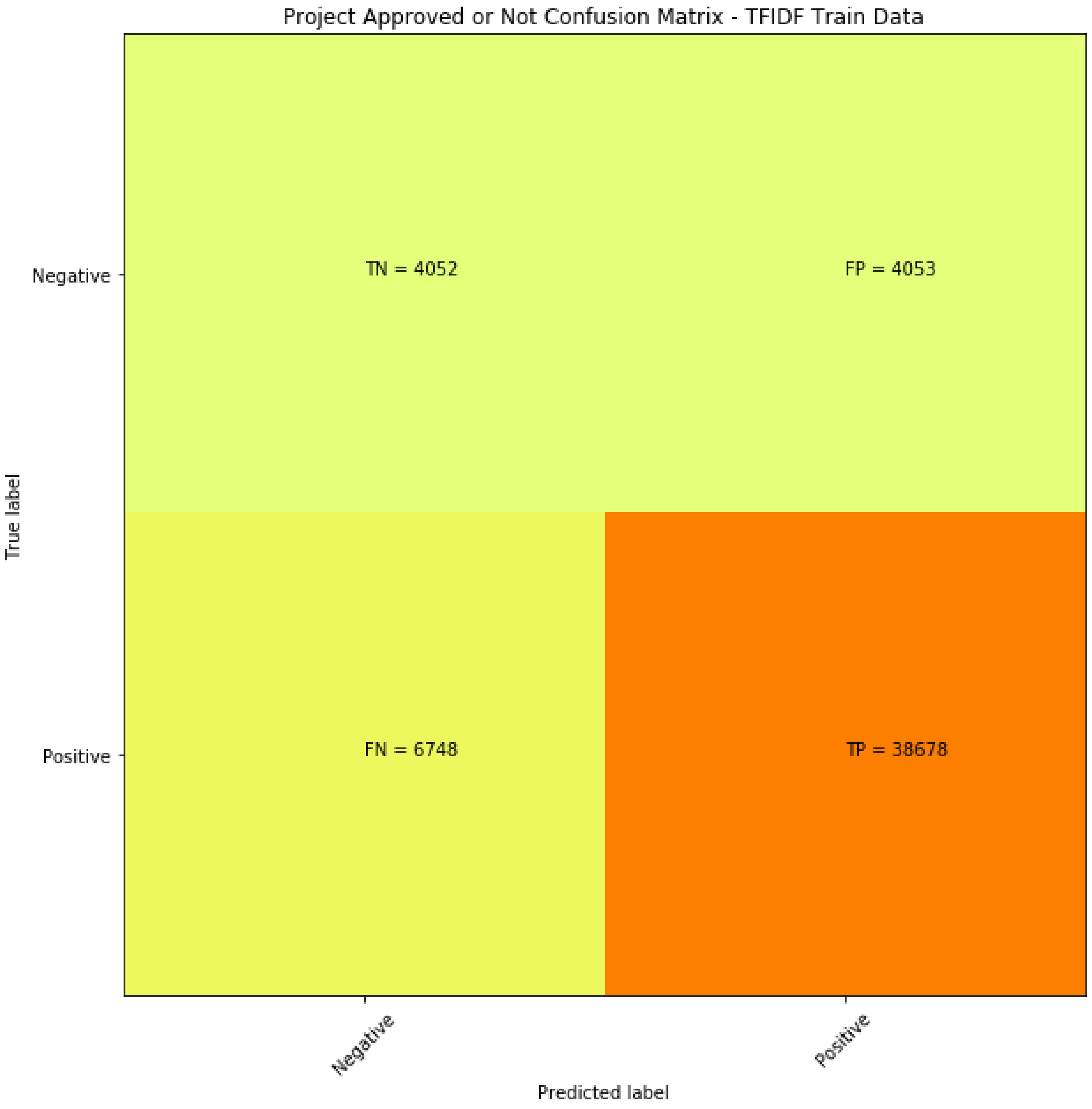
In [78]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
tfidf_train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
print(tfidf_train_confusion_matrix)
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999619430507 for threshold 0.729
[[ 4052  4053]
 [ 6748 38678]]
```

In [79]:

```
#http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(tfidf_train_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative','Positive']
plt.title('Project Approved or Not Confusion Matrix - TFIDF Train Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN','FP'], ['FN', 'TP']]
for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(tfidf_train_confusion_matrix[i][j]))
plt.show()
```



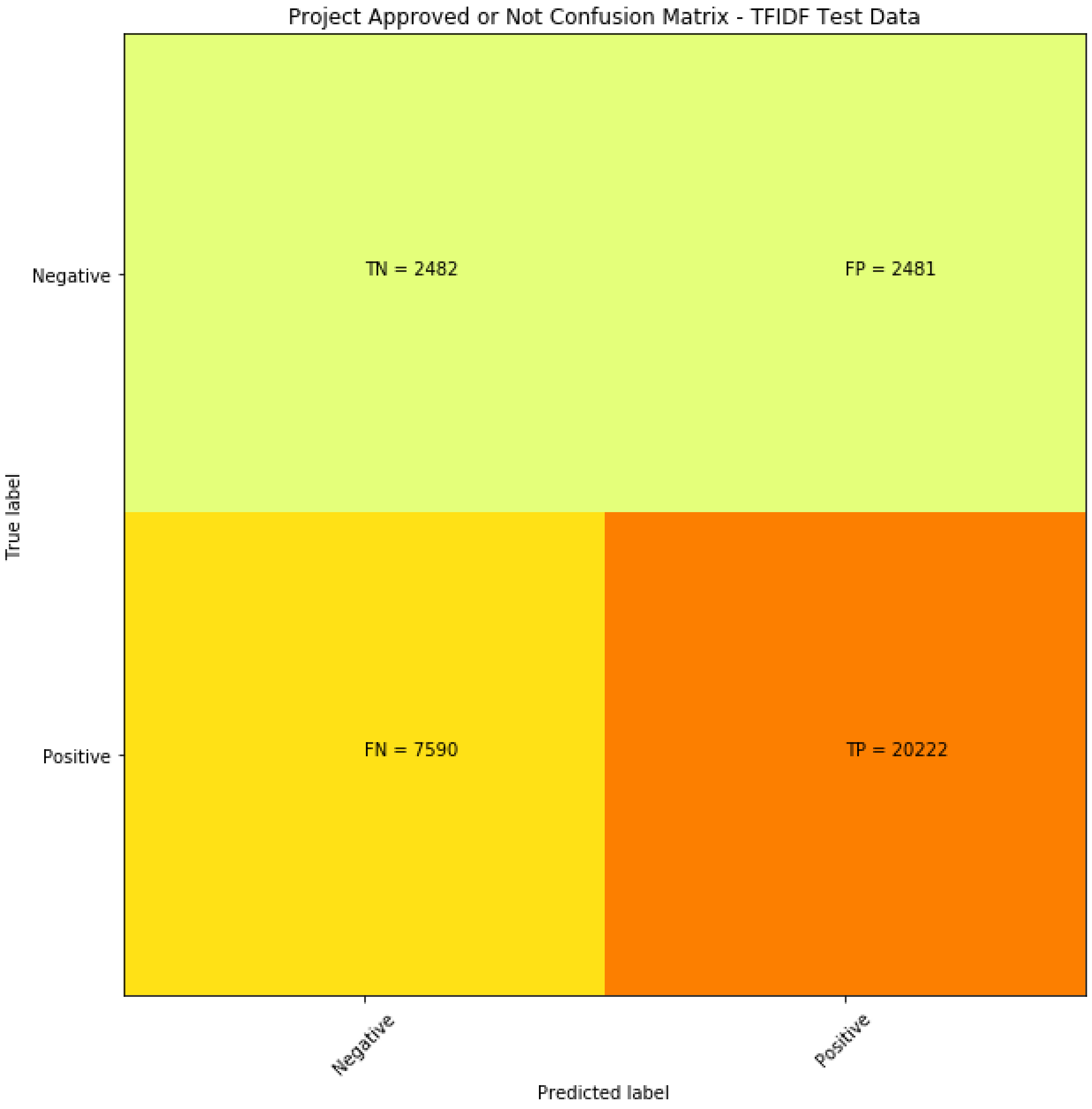
Test confusion matrix

In [80]:

```
print("Test confusion matrix")
tfidf_test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_fpr))
print(tfidf_test_confusion_matrix)
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999998985034083 for threshold 0.821
[[ 2482  2481]
 [ 7590 20222]]
```

In [81]:
```
##http://www.tarekatwan.com/index.php/2017/12/how-to-plot-a-confusion-matrix-in-python/

plt.clf()
plt.imshow(tfidf_test_confusion_matrix, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative','Positive']
plt.title('Project Approved or Not Confusion Matrix - TFIDF Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN','FP'], ['FN', 'TP']]
for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(tfidf_test_confusion_matrix[i][j]))
plt.show()
```



Data Processing for important features from SET 1 (BOW)

In [82]:
```
# Taken the help of Applied AI team to get the steps for Feature Selection for NB
# Usedthis reference as well - https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bay
es

print("Train Data matrix")
print(X_train_bow.shape, y_train.shape)
```

```
Train Data matrix
(53531, 14721) (53531,)
```

In [83]:
```
feature_names_bow =[]
feature_names_bow.extend(vectorizer_Cat.get_feature_names())
feature_names_bow.extend(vectorizer_sub_cat.get_feature_names())
feature_names_bow.extend(vectorizer_state.get_feature_names())
feature_names_bow.extend(vectorizer_teacher.get_feature_names())
feature_names_bow.extend(vectorizer_grade.get_feature_names())
feature_names_bow.extend(vectorizer_bow_essay.get_feature_names())
feature_names_bow.extend(vectorizer_bow_title.get_feature_names())
feature_names_bow.extend('price')
feature_names_bow.extend('teacher_number_of_previously_posted_projects')
len(feature_names_bow)
```

Out[83]:
```
14768
```

2.1.2.1 Top 10 important features of positive class from SET 1 (BOW)

```
In [84]: max_ind_pov=np.argsort((nb_bow.feature_log_prob_)[0])[::-1][0:10]
         top_pov_bow =np.take(feature_names_bow,max_ind_pov)
         print(top_pov_bow)
```

```
['students' 'school' 'learning' 'classroom' 'not' 'learn' 'help' 'p'
 'nannan' 'many']
```

2.1.2.2 Top 10 important features of negative class from SET 1 (BOW)

```
In [85]: max_ind_neg=np.argsort(-1*(nb_bow.feature_log_prob_)[0])[::-1][0:10]
         top_neg_bow=np.take(feature_names_bow,max_ind_neg)
         print(top_neg_bow)
```

```
['moons' 'scratches' 'fidelity' 'rainy' 'reef' 'dell' 'universally' 'ffa'
 'fever' 'scratched']
```

Data Processing for important features from SET 2 (TFIDF)

```
In [86]: feature_names_tfidf =[]
         feature_names_tfidf.extend(vectorizer_Cat.get_feature_names())
         feature_names_tfidf.extend(vectorizer_sub_cat.get_feature_names())
         feature_names_tfidf.extend(vectorizer_state.get_feature_names())
         feature_names_tfidf.extend(vectorizer_teacher.get_feature_names())
         feature_names_tfidf.extend(vectorizer_grade.get_feature_names())
         feature_names_tfidf.extend(vectorizer_tfidf_essay.get_feature_names())
         feature_names_tfidf.extend(vectorizer_tfidf_title.get_feature_names())
         feature_names_tfidf.extend('price')
         feature_names_tfidf.extend('teacher_number_of_previously_posted_projects')
         len(feature_names_tfidf)
```

```
Out[86]: 14768
```

2.1.2.2 Top 10 important features of positive class from SET 2 (TFIDF)

```
In [87]: max_ind_pov=np.argsort((nb_tfidf.feature_log_prob_)[0])[::-1][0:10]
         top_pov_tfidf =np.take(feature_names_tfidf,max_ind_pov)
         print(top_pov_tfidf)
```

```
['p' 'r' 'Mrs' 'Literacy_Language' 'Math_Science' 'Ms' 'Mathematics'
 'Literacy' 'Literature_Writing' 'SpecialNeeds']
```

2.1.2.2 Top 10 important features of Negative class from SET 2 (TFIDF)

```
In [88]: max_ind_neg=np.argsort(-1*(nb_tfidf.feature_log_prob_)[0])[::-1][0:10]
         top_neg_tfidf=np.take(feature_names_tfidf,max_ind_neg)
         print(top_neg_tfidf)
```

```
['sons' 'inconsistent' 'canvas' 'cdc' 'card' 'cards' 'carry' 'revolution'
 'cause' 'centerpiece']
```

3. Conclusions

```
In [1]: from prettytable import PrettyTable
        #If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
        x = PrettyTable()
        x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]
        x.add_row(["BOW", "Naive Bayes", 0.5, 0.70])
        x.add_row(["TFIDF", "Naive Bayes", 0.05, 0.66])
        print(x)
```

```
+------------+-------------+-----------------+------+
| Vectorizer |    Model    | Hyper Parameter | AUC  |
+------------+-------------+-----------------+------+
|    BOW     | Naive Bayes |       0.5       | 0.7  |
|   TFIDF    | Naive Bayes |       0.05      | 0.66 |
+------------+-------------+-----------------+------+
```