Table of Contents

Author : SAURABH SINGHAI

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, 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`
	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`

Feature	Description
project_subject_categories	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
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.

Import Libraries

In [1]:

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

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

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

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

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

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

Reading Data

In [2]:

```
# ******PLEASE NOTE--Considering 20K points as system becomes Unresponsive with 50K, 30K & 25K poi
nts

project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
project_data= project_data.sample(n=20000, random_state=0)
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna(' ')
```

Approved And Non Approved Projects

In [3]:

```
y_value_counts = project_data['project_is_approved'].value_counts()

print("Project Not Approved & Approved Count=\n",y_value_counts)

print("Number of projects approved for funding ", y_value_counts[1], "=", (y_value_counts[1]/(y_val
ue_counts[1]+y_value_counts[0]))*100,"%")

print("Number of projects not approved for funding ", y_value_counts[0], "=", (y_value_counts[0]/(y
_value_counts[1]+y_value_counts[0]))*100,"%")
```

```
Project Not Approved & Approved Count=
 1    16915
0     3085
Name: project_is_approved, dtype: int64
Number of projects approved for funding  16915 = 84.575 %
Number of projects not approved for funding  3085 = 15.425 %
```

Project Dataframe shape and Column Values

In [4]:

```
print("Number of data points in project data", project_data.shape)
print('*'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in project data (20000, 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']
```

Resource data shape and Column Values

In [5]:

```
print("Number of data points in resource data", resource_data.shape)
print('*'*50)
print(resource_data.columns.values)
resource_data.head(5)
```

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

Out[5]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00

	id	description	quantity	price
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

Preprocessing of `project_subject_categories`

In [6]:

```
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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            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]))
```

Preprocessing of `project_subject_subcategories`

In [7]:

```
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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            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 `project_grade_category`

In [8]:

```
print(project_data["project_grade_category"].values[0:10])
```

```
['Grades 3-5' 'Grades 6-8' 'Grades 3-5' 'Grades 3-5' 'Grades PreK-2'
 'Grades PreK-2' 'Grades 3-5' 'Grades 3-5' 'Grades PreK-2' 'Grades 9-12']
```

In [9]:

```
project_data["project_grade_category"] =
project_data["project_grade_category"].str.replace("Grades ", "")
project_data["project_grade_category"] = project_data["project_grade_category"].str.replace("-", "_
")

print(project_data["project_grade_category"].values[0:10])
```

```
['3_5' '6_8' '3_5' '3_5' 'PreK_2' 'PreK_2' '3_5' '3_5' 'PreK_2' '9_12']
```

Create new column 'Essay' by merging all project Essays

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['essay'].head(5)
```

Out[11]:

```
75155    Starting the new year off right sets the tone ...
77488    Have you ever worked so hard on a project only...
7803     My students come to class every day ready to l...
56268    \"We love science in your class!\" CJ exclaime...
46902    My students are caring, outgoing, and creative...
Name: essay, dtype: object
```

In [12]:

```
# printing some random reviews
print(project_data['essay'].values[0])
```

```
Starting the new year off right sets the tone for months to come. My class will be thrilled to rec
eive basic supplies to help them be successful.\r\n\r\nMy students are curious, inquisitive, and e
nthusiastic learners who enjoy school.\r\n\r\nOur school is a public community school in New York
City that receives Title I funding, which means that many students are eligible for free or reduce
d price lunch. Most of my students are English language learners. Our self-contained class is comp
```

rised of students with disabilities in second and third grade.We need printer ink so we can showca
se our wonderful work, and other supplies such as pocket charts for subject-specific word walls.\r
\n\r\nThe poetry book will align with our specialized phonics and reading program, and the Recipro
cal Teaching Strategies book will help us get where we need to be.\r\n\r\nChart paper is a staple
for any literacy or math lesson, and folders will help keep us organized. Ziplock pouches will att
ach to students' homework folders, making it simple and easy to transport school books home and ba
ck. \r\n\r\nPlease help us meet our needs with your support and generous donations. Thank
you!nannan

Use Decontraction function to decontract project essay

In [13]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [14]:

```
sent = decontracted(project_data['essay'].values[1])
print(sent)
print("="*50)
```

Have you ever worked so hard on a project only to get it back from a teacher with a dismal grade?
Or have a loved one that tries so very hard in school but just does not seem to grasp the
concepts? That is how my students with special needs feel everyday! I create a classroom where eve
ryone succeeds.\r\nMy students all have mild to moderate disabilities. The disabilities range fro
m various levels of autism, moderate learning disabilities, challenges with attention, to being
classified as intellectually impaired. \r\nThese students are just wonderful people but face dail
y challenges that you and I could never fathom. Most of my students come from low socioeconomic ho
mes. Suffering from disabilities makes it difficult for them to read, comprehend, write, and solve
math equations using typical learning styles. Technology is a way to bridge that learning gap thes
e students struggle with each and every day.\r\nThese Chromebooks will be used in my classroom to
help students complete their Common Core assignments in all subject areas. Students will be able t
o use this technology to help with the 21st century skills needed to be successful with the new Co
mmon Core State Standards and daily life.\r\nThis technology will make a huge impact on their
lives. We currently have a teacher computer, document camera, projector, printer, and one
chromebook per two students. The school itself has a few computer labs that it shares with the ent
ire student body. These Chromebooks will be a huge benefit to my students' lives.nannan
==

Remove line breaks

In [15]:

```
# \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)
```

Have you ever worked so hard on a project only to get it back from a teacher with a dismal grade?
Or have a loved one that tries so very hard in school but just does not seem to grasp the
concepts? That is how my students with special needs feel everyday! I create a classroom where eve
ryone succeeds. My students all have mild to moderate disabilities. The disabilities range from

fyone succeeds. My students all have mild to moderate disabilities. The disabilities range from various levels of autism, moderate learning disabilities, challenges with attention, to being classified as intellectually impaired. These students are just wonderful people but face daily challenges that you and I could never fathom. Most of my students come from low socioeconomic home s. Suffering from disabilities makes it difficult for them to read, comprehend, write, and solve m ath equations using typical learning styles. Technology is a way to bridge that learning gap these students struggle with each and every day. These Chromebooks will be used in my classroom to help students complete their Common Core assignments in all subject areas. Students will be able to use this technology to help with the 21st century skills needed to be successful with the new Common C ore State Standards and daily life. This technology will make a huge impact on their lives. We currently have a teacher computer, document camera, projector, printer, and one chromebook per two students. The school itself has a few computer labs that it shares with the entire student body. T hese Chromebooks will be a huge benefit to my students' lives.nannan

Remove Special Chars

In [16]:

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

Have you ever worked so hard on a project only to get it back from a teacher with a dismal grade O r have a loved one that tries so very hard in school but just does not seem to grasp the concepts That is how my students with special needs feel everyday I create a classroom where everyone succe eds My students all have mild to moderate disabilities The disabilities range from various levels of autism moderate learning disabilities challenges with attention to being classified as intellectually impaired These students are just wonderful people but face daily challenges that yo u and I could never fathom Most of my students come from low socioeconomic homes Suffering from di sabilities makes it difficult for them to read comprehend write and solve math equations using typical learning styles Technology is a way to bridge that learning gap these students struggle wi th each and every day These Chromebooks will be used in my classroom to help students complete the ir Common Core assignments in all subject areas Students will be able to use this technology to he lp with the 21st century skills needed to be successful with the new Common Core State Standards a nd daily life This technology will make a huge impact on their lives We currently have a teacher c omputer document camera projector printer and one chromebook per two students The school itself ha s a few computer labs that it shares with the entire student body These Chromebooks will be a huge benefit to my students lives nannan

Remove Stopwards and Join the essays

In [17]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'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', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            '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', "d
esn'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', "weren't", \
```

```
            'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:

```
# Combining all the above stundents
def Text_cleaner(data):
    from tqdm import tqdm
    preprocessed_essays = []
    # tqdm is for printing the status bar
    for sentance in tqdm(data.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())
    return preprocessed_essays
```

In [19]:

```
# after preprocesing
preprocessed_essays=Text_cleaner(project_data['essay'])
```

```
100%|████████████████████████████████████████████████████████████████████████████| 20000/20000
[00:14<00:00, 1394.06it/s]
```

In [20]:

```
preprocessed_essays[1]
```

Out[20]:

```
'ever worked hard project get back teacher dismal grade loved one tries hard school not seem grasp
concepts students special needs feel everyday create classroom everyone succeeds students mild mod
erate disabilities disabilities range various levels autism moderate learning disabilities
challenges attention classified intellectually impaired students wonderful people face daily chall
enges could never fathom students come low socioeconomic homes suffering disabilities makes
difficult read comprehend write solve math equations using typical learning styles technology way
bridge learning gap students struggle every day chromebooks used classroom help students complete
common core assignments subject areas students able use technology help 21st century skills needed
successful new common core state standards daily life technology make huge impact lives currently
teacher computer document camera projector printer one chromebook per two students school computer
labs shares entire student body chromebooks huge benefit students lives nannan'
```

Drop essay columns 1, 2, 3, 4

In [21]:

```
project_data['essay'] = preprocessed_essays
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
project_data.head()
```

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
75155	144107	p064182	7414165942b20a8d7fe5bcdc96244624	Ms.	NY	2017-02-05 17:49:01

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
77488	89277	p187708	5b42a9aa00917ac1716d8063aebc6318	Mrs.	CA	2016-05-27 14:44:25
7803	123550	p142214	bec515840d4fb7d2ba1071211ba32231	Mrs.	CA	2016-12-09 19:23:16
56268	104617	p098697	8131749e34b7ef3fa0890b5d840deb2a	Ms.	NC	2016-06-20 13:27:03
46902	154452	p252651	d240517694ebcbe54a5ffa806a5ada2e	Ms.	MO	2016-11-09 15:54:10

Preprocessing of `project_title`

In [22]:

```
# similarly you can preprocess the titles also
preprocessed_project_title=Text_cleaner(project_data['project_title'])
```

```
100%|████████████████████████████████████████████████████████████| 20000/20000
[00:00<00:00, 28269.59it/s]
```

In [23]:

```
preprocessed_project_title[1]
```

Out[23]:

```
'keep spirit alive'
```

Drop column project_title and use Cleaned_Title

In [24]:

```
project_data['Cleaned_title']= preprocessed_project_title
project_data.drop(['project_title'], axis=1, inplace=True)
```

Add up the price based on project id

In [25]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [26]:

```
project_data.columns
```

Out[26]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'Cleaned_title',
       'price', 'quantity'],
      dtype='object')
```

In [27]:

```
project_data.drop(['project_resource_summary'], axis=1, inplace=True)
project_data.drop(['Unnamed: 0'], axis=1, inplace=True)
project_data.drop(['id'], axis=1, inplace=True)
project_data.drop(['teacher_id'], axis=1, inplace=True)
```

Adding the word count of essay and title as new columns

In [28]:

```
project_data['essay_count']=project_data['essay'].str.len()
project_data['title_count']=project_data['Cleaned_title'].str.len()
```

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

-Length of words in essay
-Length of words in title

Separate out the Dependant and independant variables

In [29]:

```
#https://stackoverflow.com/questions/29763620/how-to-select-all-columns-except-one-column-in-pandas
X=project_data.loc[:, project_data.columns != 'project_is_approved']
y=project_data['project_is_approved']
X.shape
```

Out[29]:

```
(20000, 13)
```

Splitting data into Test,Train,CV

In [30]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

from sklearn.model_selection import train_test_split

X_train,X_test,y_train, y_test=train_test_split(X, y, test_size=0.3, random_state=0,stratify=y)
X_train,X_cv,y_train,y_cv=train_test_split(X_train, y_train, test_size=0.3,
random_state=0,stratify=y_train)

print(X_train.shape)
print(X_test.shape)
print(X_cv.shape)
print(y_train.shape)
print(y_test.shape)
print(y_cv.shape)
```

```
(9800, 13)
(6000, 13)
(4200, 13)
(9800,)
(6000,)
(4200,)
```

In [31]:

```
X.head(2)
```

Out[31]:

	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	teacher_number_of_previously_poste
0	Ms.	NY	2017-02-05 17:49:01	3_5	7
1	Mrs.	CA	2016-05-27 14:44:25	6_8	3

Vectorize the features

Vectorize the Categorical Features - categories

In [32]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['clean_categories'].values)

# we use the fitted CountVectorizer to transform the text to vector
X_train_clean_categories=vectorizer.transform(X_train['clean_categories'].values)
X_test_clean_categories=vectorizer.transform(X_test['clean_categories'].values)
X_cv_clean_categories=vectorizer.transform(X_cv['clean_categories'].values)
```

```
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_clean_categories.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix after one hot encodig  (9800, 9)
```

Vectorize the Categorical Features - subcategories

In [33]:

```
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_sub_categories=vectorizer.transform(X_train['clean_subcategories'].values)
X_test_clean_sub_categories=vectorizer.transform(X_test['clean_subcategories'].values)
X_cv_clean_sub_categories=vectorizer.transform(X_cv['clean_subcategories'].values)


print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_clean_sub_categories.shape)
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
Shape of matrix after one hot encodig  (9800, 30)
```

Vectorize the Categorical Features - school state

In [34]:

```
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['school_state'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_skl_state=vectorizer.transform(X_train['school_state'].values)
X_test_skl_state=vectorizer.transform(X_test['school_state'].values)
X_cv_skl_state=vectorizer.transform(X_cv['school_state'].values)


print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_skl_state.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encodig  (9800, 51)
```

Vectorize the Categorical Features - teacher prefix

In [35]:

```
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['teacher_prefix'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_prefix=vectorizer.transform(X_train['teacher_prefix'].values)
X_test_teacher_prefix=vectorizer.transform(X_test['teacher_prefix'].values)
X_cv_teacher_prefix=vectorizer.transform(X_cv['teacher_prefix'].values)


print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_teacher_prefix.shape)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (9800, 5)
```

Vectorize the Categorical Features - project_grade_category

In [36]:

```
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_project_grade_category=vectorizer.transform(X_train['project_grade_category'].values)
X_test_project_grade_category=vectorizer.transform(X_test['project_grade_category'].values)
X_cv_project_grade_category=vectorizer.transform(X_cv['project_grade_category'].values)


print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_project_grade_category.shape)
```

```
['3_5', '6_8', '9_12', 'PreK_2']
Shape of matrix after one hot encodig  (9800, 4)
```

Vectorize the Numerical Features - price

In [37]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
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.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standard deviation
of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
X_train_price_standardized = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
X_test_price_standardized = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
X_cv_price_standardized = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
```

```
Mean : 297.2610204081633, Standard deviation : 364.035416010753
```

Vectorize the Numerical Features - quantity

In [38]:

```
quantity_scalar = StandardScaler()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation :
{np.sqrt(quantity_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
X_train_quantity_standardized = quantity_scalar.transform(X_train['quantity'].values.reshape(-1, 1)
)
X_test_quantity_standardized = quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
X_cv_quantity_standardized = quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
```

```
Mean : 16.866632653061224, Standard deviation : 25.64636006491646
```

Vectorize the Numerical Features - essay count

In [39]:

```
count_scalar = StandardScaler()
count_scalar.fit(X_train['essay_count'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
print(f"Mean : {count_scalar.mean_[0]}, Standard deviation : {np.sqrt(count_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
X_train_essay_count_standardized = count_scalar.transform(X_train['essay_count'].values.reshape(-1
, 1))
X_test_essay_count_standardized = count_scalar.transform(X_test['essay_count'].values.reshape(-1, 1
))
X_cv_essay_count_standardized = count_scalar.transform(X_cv['essay_count'].values.reshape(-1, 1))
```

```
Mean : 1012.2668367346939, Standard deviation : 275.8740431351019
```

Vectorize the Numerical Features - title count

In [40]:

```
count_scalar = StandardScaler()
count_scalar.fit(X_train['title_count'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
print(f"Mean : {count_scalar.mean_[0]}, Standard deviation : {np.sqrt(count_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
X_train_title_count_standardized = count_scalar.transform(X_train['title_count'].values.reshape(-1
, 1))
X_test_title_count_standardized = count_scalar.transform(X_test['title_count'].values.reshape(-1, 1
))
X_cv_title_count_standardized = count_scalar.transform(X_cv['title_count'].values.reshape(-1, 1))
```

```
Mean : 25.561122448979592, Standard deviation : 11.730413060793852
```

Vectorizing Text data

Bag of words - essay

In [41]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow=vectorizer.transform(X_train['essay'].values)
X_test_essay_bow=vectorizer.transform(X_test['essay'].values)
X_cv_essay_bow=vectorizer.transform(X_cv['essay'].values)


print("Shape of matrix after one hot encodig ",X_train_essay_bow.shape)
```

```
Shape of matrix after one hot encodig  (9800, 5000)
```

Bag of words - cleaned title

In [42]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer = CountVectorizer(min_df=5)
vectorizer.fit(X_train['Cleaned_title'])
```

```
# we use the fitted CountVectorizer to convert the text to vector
X_train_cleaned_title_bow=vectorizer.transform(X_train['Cleaned_title'].values)
X_test_cleaned_title_bow=vectorizer.transform(X_test['Cleaned_title'].values)
X_cv_cleaned_title_bow=vectorizer.transform(X_cv['Cleaned_title'].values)


print("Shape of matrix after one hot encodig ",X_train_cleaned_title_bow.shape)
```

```
Shape of matrix after one hot encodig  (9800, 1102)
```

TFIDF vectorizer - essay

In [43]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf=vectorizer.transform(X_train['essay'].values)
X_test_essay_tfidf=vectorizer.transform(X_test['essay'].values)
X_cv_essay_tfidf=vectorizer.transform(X_cv['essay'].values)


print("Shape of matrix after one hot encodig ",X_train_essay_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (9800, 5000)
```

TFIDF vectorizer - cleaned tittle

In [44]:

```
# Similarly you can vectorize for title alsovectorizer = TfidfVectorizer(min_df=10)
vectorizer = TfidfVectorizer(min_df=5)

# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer.fit(X_train['Cleaned_title'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleaned_title_tfidf=vectorizer.transform(X_train['Cleaned_title'].values)
X_test_cleaned_title_tfidf=vectorizer.transform(X_test['Cleaned_title'].values)
X_cv_cleaned_title_tfidf=vectorizer.transform(X_cv['Cleaned_title'].values)


print("Shape of matrix after one hot encodig ",X_train_cleaned_title_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (9800, 1102)
```

Using Pretrained Models: Avg W2V

In [45]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
```

```
Output:

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

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

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

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

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


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)


'''
```

Out[45]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef
loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\',
encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n
word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\'
\'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the
coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus",
len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words tha
t are present in both glove vectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\r
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [46]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [47]:

```
# average Word2Vec
# compute average word2vec for each review.
def avg_w2v_vectors(preprocessed_essays):
    avg_w2v_vectors_text = []; # the avg-w2v for each sentence/review is stored in this list
```

```
    for sentence in tqdm(preprocessed_essays): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        cnt_words =0; # num of words with a valid vector in the sentence/review
        for word in sentence.split(): # for each word in a review/sentence
            if word in glove_words:
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_vectors_text.append(vector)
    return avg_w2v_vectors_text

X_train_essay_w2v=avg_w2v_vectors(X_train['essay'])
X_test_essay_w2v=avg_w2v_vectors(X_test['essay'])
X_cv_essay_w2v=avg_w2v_vectors(X_cv['essay'])

X_train_cleaned_title_w2v=avg_w2v_vectors(X_train['Cleaned_title'])
X_test_cleaned_title_w2v=avg_w2v_vectors(X_test['Cleaned_title'])
X_cv_cleaned_title_w2v=avg_w2v_vectors(X_cv['Cleaned_title'])
```

```
100%|████████████████████████████████████████████████████████████████| 9800/9800
[00:03<00:00, 2654.05it/s]
100%|████████████████████████████████████████████████████████████████| 6000/6000
[00:02<00:00, 2676.78it/s]
100%|████████████████████████████████████████████████████████████████| 4200/4200
[00:01<00:00, 2643.07it/s]
100%|████████████████████████████████████████████████████████████████| 9800/9800
[00:00<00:00, 40302.40it/s]
100%|████████████████████████████████████████████████████████████████| 6000/6000
[00:00<00:00, 41929.55it/s]
100%|████████████████████████████████████████████████████████████████| 4200/4200
[00:00<00:00, 43720.92it/s]
```

Using Pretrained Models: TFIDF weighted W2V - Essay

In [48]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words_essay = set(tfidf_model.get_feature_names())
print(len(tfidf_words_essay))
```

```
21992
```

In [49]:

```
# average Word2Vec
# compute average word2vec for each review.
def tfidf_w2v_vectors(tfidf_words,preprocessed_essays):
    tfidf_w2v_vectors_text = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in tqdm(preprocessed_essays): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        tf_idf_weight =0; # num of words with a valid vector in the sentence/review
        for word in sentence.split(): # for each word in a review/sentence
            if (word in glove_words) and (word in tfidf_words):
                vec = model[word] # getting the vector for each word
                # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
                tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
the tfidf value for each word
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors_text.append(vector)
    return tfidf_w2v_vectors_text

X_train_essay_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_essay,X_train['essay'])
X_test_essay_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_essay,X_test['essay'])
X_cv_essay_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_essay,X_cv['essay'])
```

```
X_cv_essay_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_essay,X_cv['essay'])
```

```
100%|████████████████████████████████████████████████████████████████| 9800/9800
[00:26<00:00, 374.65it/s]
100%|████████████████████████████████████████████████████████████████| 6000/6000
[00:16<00:00, 367.42it/s]
100%|████████████████████████████████████████████████████████████████| 4200/4200
[00:10<00:00, 382.89it/s]
```

Using Pretrained Models: TFIDF weighted W2V - Cleaned Title

In [50]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['Cleaned_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words_Cleaned_title = set(tfidf_model.get_feature_names())
print(len(tfidf_words_Cleaned_title))
```

```
5190
```

In [51]:

```
X_train_cleaned_title_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_Cleaned_title,X_train['Cleaned_title'
])
X_test_cleaned_title_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_Cleaned_title,X_test['Cleaned_title']
)
X_cv_cleaned_title_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_Cleaned_title,X_cv['Cleaned_title'])
```

```
100%|████████████████████████████████████████████████████████████████| 9800/9800
[00:00<00:00, 21106.53it/s]
100%|████████████████████████████████████████████████████████████████| 6000/6000
[00:00<00:00, 22044.09it/s]
100%|████████████████████████████████████████████████████████████████| 4200/4200
[00:00<00:00, 21635.02it/s]
```

In [52]:

```
X_train_prev_proj=X_train['teacher_number_of_previously_posted_projects'][:,np.newaxis]

X_test_prev_proj=X_test['teacher_number_of_previously_posted_projects'][:,np.newaxis]

X_cv_prev_proj=X_cv['teacher_number_of_previously_posted_projects'][:,np.newaxis]
```

Merging all the above features

BOW

In [53]:

```
# 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 matirx :)
X_train_bow = hstack((X_train_clean_categories,
X_train_clean_sub_categories,X_train_skl_state,X_train_teacher_prefix,
            X_train_project_grade_category,X_train_price_standardized,X_train_quantity_standardized
,X_train_prev_proj,
            X_train_essay_bow,X_train_cleaned_title_bow,X_train_essay_count_standardized,X_train_t
tle_count_standardized
            )).toarray()


X_test_bow = hstack((X_test_clean_categories,
X_test_clean_sub_categories,X_test_skl_state,X_test_teacher_prefix,
            X_test_project_grade_category,X_test_price_standardized,X_test_quantity_standardized,X
```

```
test_prev_proj,
            X_test_essay_bow,X_test_cleaned_title_bow,X_test_essay_count_standardized,X_test_title_
count_standardized
            )).toarray()


X_cv_bow = hstack((X_cv_clean_categories,
X_cv_clean_sub_categories,X_cv_skl_state,X_cv_teacher_prefix,
            X_cv_project_grade_category,X_cv_price_standardized,X_cv_quantity_standardized,X_cv_pre
v_proj,

X_cv_essay_bow,X_cv_cleaned_title_bow,X_cv_essay_count_standardized,X_cv_title_count_standardized
            )).toarray()


print(X_train_bow.shape)
print(X_test_bow.shape)
print(X_cv_bow.shape)
```

```
(9800, 6206)
(6000, 6206)
(4200, 6206)
```

TFIDF

In [54]:

```
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_tfidf = hstack((X_train_clean_categories, X_train_clean_sub_categories,X_train_skl_state,X
_train_teacher_prefix,
            X_train_project_grade_category,X_train_price_standardized,X_train_quantity_standardized
,X_train_prev_proj,
            X_train_essay_tfidf,X_train_cleaned_title_tfidf,X_train_essay_count_standardized,X_trai
n_title_count_standardized
            )).toarray()
```

In [55]:

```
X_test_tfidf = hstack((X_test_clean_categories,
X_test_clean_sub_categories,X_test_skl_state,X_test_teacher_prefix,
            X_test_project_grade_category,X_test_price_standardized,X_test_quantity_standardized,X_
test_prev_proj,
            X_test_essay_tfidf,X_test_cleaned_title_tfidf,X_test_essay_count_standardized,X_test_ti
tle_count_standardized
            )).toarray()
```

In [56]:

```
X_cv_tfidf = hstack((X_cv_clean_categories,
X_cv_clean_sub_categories,X_cv_skl_state,X_cv_teacher_prefix,
            X_cv_project_grade_category,X_cv_price_standardized,X_cv_quantity_standardized,X_cv_pre
v_proj,

X_cv_essay_tfidf,X_cv_cleaned_title_tfidf,X_cv_essay_count_standardized,X_cv_title_count_standardiz
d
            )).toarray()


print(X_train_tfidf.shape)
print(X_test_tfidf.shape)
print(X_cv_tfidf.shape)
```

```
(9800, 6206)
(6000, 6206)
(4200, 6206)
```

Word2Vec

In [57]:

In [57]:

```
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_w2v = hstack((X_train_clean_categories,
X_train_clean_sub_categories,X_train_skl_state,X_train_teacher_prefix,
            X_train_project_grade_category,X_train_price_standardized,X_train_quantity_standardized
,X_train_prev_proj,
            X_train_essay_w2v,X_train_cleaned_title_w2v,X_train_essay_count_standardized,X_train_t
tle_count_standardized
            )).toarray()


X_test_w2v = hstack((X_test_clean_categories,
X_test_clean_sub_categories,X_test_skl_state,X_test_teacher_prefix,
            X_test_project_grade_category,X_test_price_standardized,X_test_quantity_standardized,X_
test_prev_proj,
            X_test_essay_w2v,X_test_cleaned_title_w2v,X_test_essay_count_standardized,X_test_title_
count_standardized
            )).toarray()


X_cv_w2v = hstack((X_cv_clean_categories,
X_cv_clean_sub_categories,X_cv_skl_state,X_cv_teacher_prefix,
            X_cv_project_grade_category,X_cv_price_standardized,X_cv_quantity_standardized,X_cv_pre
v_proj,

X_cv_essay_w2v,X_cv_cleaned_title_w2v,X_cv_essay_count_standardized,X_cv_title_count_standardized
            )).toarray()


print(X_train_w2v.shape)
print(X_test_w2v.shape)
print(X_cv_w2v.shape)
```

```
(9800, 704)
(6000, 704)
(4200, 704)
```

TFIDF- WORD2VEC

In [58]:

```
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_tfidf_w2v = hstack((X_train_clean_categories,
X_train_clean_sub_categories,X_train_skl_state,X_train_teacher_prefix,
            X_train_project_grade_category,X_train_price_standardized,X_train_quantity_standardized
,X_train_prev_proj,

X_train_essay_tfidf_w2v,X_train_cleaned_title_tfidf_w2v,X_train_essay_count_standardized,X_train_ti
tle_count_standardized
            )).toarray()


X_test_tfidf_w2v = hstack((X_test_clean_categories, X_test_clean_sub_categories,X_test_skl_state,X
_test_teacher_prefix,
            X_test_project_grade_category,X_test_price_standardized,X_test_quantity_standardized,X_
test_prev_proj,
            X_test_essay_tfidf_w2v,X_test_cleaned_title_tfidf_w2v,X_test_essay_count_standardized,X
_test_title_count_standardized
            )).toarray()


X_cv_tfidf_w2v = hstack((X_cv_clean_categories,
X_cv_clean_sub_categories,X_cv_skl_state,X_cv_teacher_prefix,
            X_cv_project_grade_category,X_cv_price_standardized,X_cv_quantity_standardized,X_cv_pre
v_proj,
            X_cv_essay_tfidf_w2v,X_cv_cleaned_title_tfidf_w2v,X_cv_essay_count_standardized,X_cv_ti
tle_count_standardized
            )).toarray()


print(X_train_tfidf_w2v.shape)
print(X_test_tfidf_w2v.shape)
print(X_cv_tfidf_w2v.shape)
```

```
(9800, 704)
(6000, 704)
(4200, 704)
```

Assignment 3: Apply KNN

1. **[Task-1] Apply KNN(brute force version) on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)
   - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)
   - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)
2. **Hyper paramter tuning to find best K**
   - Find the best hyper parameter which results in the maximum AUC value
   - Find the best hyper paramter using k-fold cross validation (or) simple cross validation data
   - Use gridsearch-cv or randomsearch-cv or write your own for loops to do this task
3. **Representation of results**
   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure
   - Once you find the best hyper parameter, you need to train your model-M using the best hyper-param. Now, find the AUC on test data and plot the ROC curve on both train and test using model-M.
   - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points
4. **[Task-2]**
   - Select top 2000 features from feature Set 2 using `SelectKBest` and then apply KNN on top of these features
     - ```
       from sklearn.datasets import load_digits
       from sklearn.feature_selection import SelectKBest, chi2
       X, y = load_digits(return_X_y=True)
       X.shape
       X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
       X_new.shape
       ========
       output:
       (1797, 64)
       (1797, 20)
       ```
   - Repeat the steps 2 and 3 on the data matrix after feature selection
5. **Conclusion**
   - 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 link

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link.

K Nearest Neighbor

In [59]:
       ```

```
print(X_train_bow.shape)
print(X_cv_bow.shape)
print(X_test_bow.shape)

print('='*50)
print(X_train_tfidf.shape)
print(X_cv_tfidf.shape)
print(X_test_tfidf.shape)

print('='*50)
print(X_train_w2v.shape)
print(X_cv_w2v.shape)
print(X_test_w2v.shape)

print('='*50)
print(X_train_tfidf_w2v.shape)
print(X_cv_tfidf_w2v.shape)
print(X_test_tfidf_w2v.shape)

print('='*50)
print(y_train.shape)
print(y_test.shape)
print(y_cv.shape)
```

```
(9800, 6206)
(4200, 6206)
(6000, 6206)
==================================================
(9800, 6206)
(4200, 6206)
(6000, 6206)
==================================================
(9800, 704)
(4200, 704)
(6000, 704)
==================================================
(9800, 704)
(4200, 704)
(6000, 704)
==================================================
(9800,)
(6000,)
(4200,)
```

Appling KNN on different kind of featurization as mentioned in the instructions

Applying KNN brute force on BOW, SET 1

Hyperparameter tuning for finiding optimal k using ROC_AUC_Score

In [60]:

```
#applied AI course material
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs

    #print("data.shape[0]-->", data.shape[0])
    #print("*"*50)
    # predict the response
    y_data_pred = []
    for i in range(0,data.shape[0],1000): #(CHUNK of data)
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    y_data_pred = np.hstack(y_data_pred)
    #print("y_data_pred-->\n", y_data_pred)
    #print("*"*50)
#     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]:

```
# appliedAI course material
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

summary=[]
roc_auc_score_cv_bow_dict={}
roc_auc_score_train_bow_dict={}

for i in tqdm(range(1,50,8)):
    knn = KNeighborsClassifier(n_neighbors=i, algorithm = 'kd_tree')

    # 1.fitting the model on X_train
    knn.fit(X_train_bow, y_train)

    # 2.predict the response on the CV data
    pred_bow_cv = batch_predict(knn,X_cv_bow)

    # 3.evaluate CV roc_auc
    roc_auc_cv = roc_auc_score(y_cv,pred_bow_cv)

    # 4.insert into dict of CV data
    roc_auc_score_cv_bow_dict[i]=roc_auc_cv

    # 5.predict the response on the TRAIN data
    pred_bow_train = batch_predict(knn,X_train_bow)

    # 6.evaluate train roc_auc
    roc_auc_train =roc_auc_score(y_train,pred_bow_train)

    # 7.insert into dict of train data
    roc_auc_score_train_bow_dict[i]=roc_auc_train




print(roc_auc_score_cv_bow_dict)
print(roc_auc_score_train_bow_dict)
```

```
100%|██████████| 7/7 [1:48:06<00:00, 942.85s/it]
```

```
{1: 0.5191806389723057, 9: 0.571361291847403, 17: 0.5912568384356578, 25: 0.6039917956150593, 33:
0.6063202959904349, 41: 0.6159310352018685, 49: 0.6228856895089534}
{1: 1.0, 9: 0.7718414579726953, 17: 0.7235437331367935, 25: 0.705581117088909, 33:
0.6934564550516624, 41: 0.6852398772056354, 49: 0.6774157645078169}
```

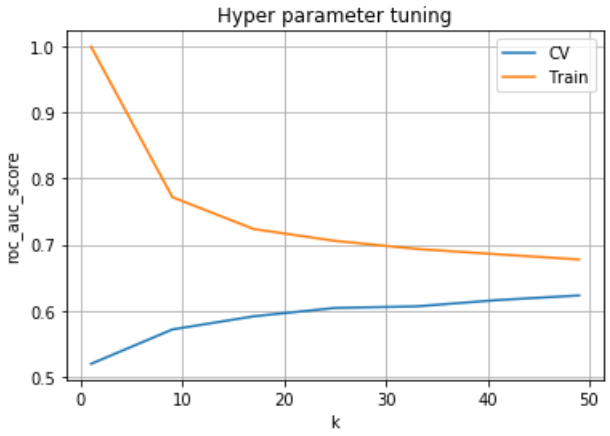
Plot ROC_AUC_score VS different K values (Train and CV set)

In [62]:

```
#https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-
values/37266356

lists1 = sorted(roc_auc_score_cv_bow_dict.items())
x1, y1 = zip(*lists1) # unpack a list of pairs into two tuples
lists2 = sorted(roc_auc_score_train_bow_dict.items())
x2, y2 = zip(*lists2) # unpack a list of pairs into two tuples
plt.xlabel('k')
plt.ylabel('roc_auc_score')
plt.title('Hyper parameter tuning')
plt.plot(x1, y1,label="CV")
plt.plot(x2, y2,label='Train')
plt.legend()
```

```
plt.grid()
plt.show()
```



Find the best K value from the result

In [63]:

```
# This function returns the key (K) with max value in the dictionary
def find_highest_k(mydict):
    #print dictionary contents
    print("Dictionary Contents-->\n",mydict)
    print("="*50)
    #find maximum value from dixtionary
    maxval=max(mydict.values())

    # make a list of keys
    keys = list(mydict.keys())
    print("Keys in the list are-->\n", keys)
    print("="*50)
    # make a list of values
    vals = list(mydict.values())
    print("Values in the list are-->\n", vals)
    print("="*50)
    # find the index of max value and use it to find corresponding key
    k=(keys[vals.index(maxval)])
    print("Maximum k is {0} ".format(k))
    print("="*50)
    return(k)

print(find_highest_k(roc_auc_score_cv_bow_dict))
```

```
Dictionary Contents-->
 {1: 0.5191806389723057, 9: 0.571361291847403, 17: 0.5912568384356578, 25: 0.6039917956150593, 33:
0.6063202959904349, 41: 0.6159310352018685, 49: 0.6228856895089534}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5191806389723057, 0.571361291847403, 0.5912568384356578, 0.6039917956150593,
0.6063202959904349, 0.6159310352018685, 0.6228856895089534]
==================================================
Maximum k is 49
==================================================
49
```

Train the model on the optimal K value and run the Test Dataset

-- Plot the ROC curve for BOW using the test and train Dataset

In [64]:

```
# train model on the best k
knn = KNeighborsClassifier(find_highest_k(roc_auc_score_cv_bow_dict), algorithm = 'kd_tree')
```

```
# fitting the model on train

knn.fit(X_train_bow, y_train)


#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_bow_test_scores=batch_predict(knn,X_test_bow)
print("pred_bow_test_scores-->\n", pred_bow_test_scores)

pred_bow_train_scores=batch_predict(knn,X_train_bow)
print("pred_bow_train_scores-->\n", pred_bow_train_scores)

# Calculate fpr, tpr for test
fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_bow_test_scores)

# Calculate fpr, tpr for train
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_bow_train_scores)

# Calculate auc from fpr, tpr for test
roc_auc_test = auc(fpr_test, tpr_test)

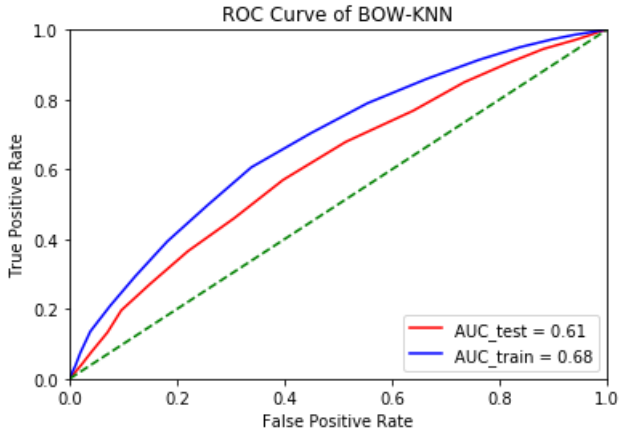
# Calculate auc from fpr, tpr for train
roc_auc_train = auc(fpr_train, tpr_train)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)

plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)

plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of BOW-KNN')
plt.show()
```

```
Dictionary Contents-->
 {1: 0.5191806389723057, 9: 0.571361291847403, 17: 0.5912568384356578, 25: 0.6039917956150593, 33:
0.6063202959904349, 41: 0.6159310352018685, 49: 0.6228856895089534}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5191806389723057, 0.571361291847403, 0.5912568384356578, 0.6039917956150593,
0.6063202959904349, 0.6159310352018685, 0.6228856895089534]
==================================================
Maximum k is 49
==================================================
pred_bow_test_scores-->
 [0.71428571 0.79591837 0.81632653 ... 0.75510204 0.69387755 0.75510204]
pred_bow_train_scores-->
 [0.65306122 0.83673469 0.79591837 ... 0.79591837 0.75510204 0.81632653]
```



Get the confusion matrix for the BOW - KNN

In [65]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshold, fpr, tpr):

    t = threshold[np.argmax(fpr*(1-tpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

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

In [66]:

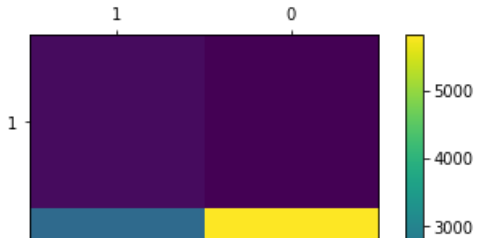
```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for BOW")
print("********************")
labels = [1,0]
cm =confusion_matrix(y_train,predict(pred_bow_train_scores,threshold_train,fpr_train,fpr_train))

print("Confusion Matrix")
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.show()
print("="*50)

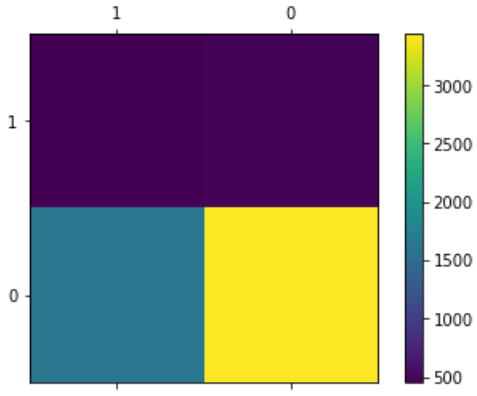

print("Testing CM for BOW")
labels = [1,0]
cm =confusion_matrix(y_test,predict(pred_bow_test_scores,threshold_test,fpr_test,fpr_test))
summary.append(['BoW',find_highest_k(roc_auc_score_cv_bow_dict),roc_auc_test])

print("Confusion Matrix")
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.show()
```

```
Training CM for BOW
********************
the maximum value of tpr*(1-fpr) 0.2473692809097722 for threshold 0.776
Confusion Matrix
[[ 833  678]
 [2457 5832]]
```




```
==================================================
Testing CM for BOW
the maximum value of tpr*(1-fpr) 0.24980290993567167 for threshold 0.776
Dictionary Contents-->
 {1: 0.5191806389723057, 9: 0.571361291847403, 17: 0.5912568384356578, 25: 0.6039917956150593, 33:
0.6063202959904349, 41: 0.6159310352018685, 49: 0.6228856895089534}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5191806389723057, 0.571361291847403, 0.5912568384356578, 0.6039917956150593,
0.6063202959904349, 0.6159310352018685, 0.6228856895089534]
==================================================
Maximum k is 49
==================================================
Confusion Matrix
[[ 450  476]
 [1629 3445]]
```



Applying KNN brute force on TFIDF, SET 2

Hyperparameter tuning for finiding optimal k using ROC_AUC_Score

In [67]:

```
roc_auc_score_cv_tfidf_dict={}
roc_auc_score_train_tfidf_dict={}

for i in tqdm(range(1,50,8)):
    knn = KNeighborsClassifier(n_neighbors=i, algorithm = 'kd_tree')

     # fitting the model on crossvalidation train
    knn.fit(X_train_tfidf, y_train)

    # predict the response on the crossvalidation train
    pred_tfidf_cv = batch_predict(knn,X_cv_tfidf)

    #evaluate CV roc_auc
    roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_cv)

    #insert into dict
    roc_auc_score_cv_tfidf_dict[i]=roc_auc_cv

    # predict the response on the train
    pred_tfidf_train = batch_predict(knn,X_train_tfidf)

    #evaluate train roc_auc
```

```
    roc_auc_train =roc_auc_score(y_train,pred_tfidf_train)

    #insert into dict
    roc_auc_score_train_tfidf_dict[i]=roc_auc_train



print(roc_auc_score_cv_tfidf_dict)
print(roc_auc_score_train_tfidf_dict)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 7/7
[1:01:39<00:00, 546.41s/it]
```

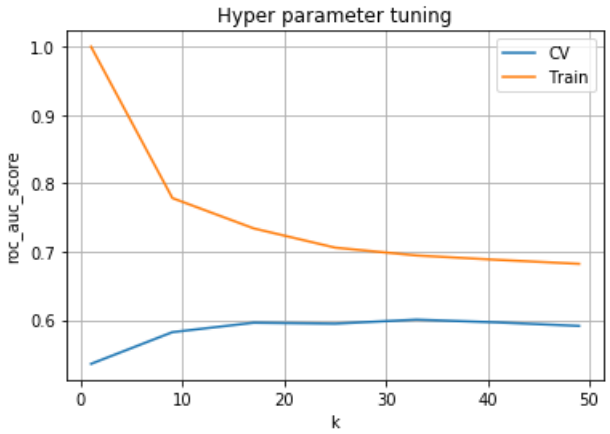
```
{1: 0.5356919419419419, 9: 0.5821216181459236, 17: 0.5959066705594483, 25: 0.5945233427872317, 33:
0.6004861198003559, 41: 0.5963311401679458, 49: 0.5912320740879768}
{1: 1.0, 9: 0.778367573332618, 17: 0.7339842003136368, 25: 0.7058542179005146, 33:
0.6943834249165188, 41: 0.687971643824165, 49: 0.6821252664439543}
```

Plot ROC_AUC_score VS different K values (Train and CV set)

In [68]:

```
#https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-
values/37266356

lists1 = sorted(roc_auc_score_cv_tfidf_dict.items())
x1, y1 = zip(*lists1) # unpack a list of pairs into two tuples
lists2 = sorted(roc_auc_score_train_tfidf_dict.items())
x2, y2 = zip(*lists2) # unpack a list of pairs into two tuples
plt.xlabel('k')
plt.ylabel('roc_auc_score')
plt.title('Hyper parameter tuning')
plt.plot(x1, y1,label="CV")
plt.plot(x2, y2,label='Train')
plt.legend()
plt.grid()
plt.show()
```



Find the best K value from the result

In [69]:

```
print(find_highest_k(roc_auc_score_cv_tfidf_dict))
```

```
Dictionary Contents-->
 {1: 0.5356919419419419, 9: 0.5821216181459236, 17: 0.5959066705594483, 25: 0.5945233427872317, 33
: 0.6004861198003559, 41: 0.5963311401679458, 49: 0.5912320740879768}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5356919419419419, 0.5821216181459236, 0.5959066705594483, 0.5945233427872317,
```

```
0.6004861198003559, 0.5963311401679458, 0.5912320740879768]
==================================================
Maximum k is 33
==================================================
33
```

Train the model on the optimal K value and run the Test Dataset

-- Plot the ROC curve for TFIDF using the test and train Dataset

In [70]:

```
# train model on the best k
knn = KNeighborsClassifier(n_neighbors=find_highest_k(roc_auc_score_cv_tfidf_dict), algorithm =
'kd_tree')

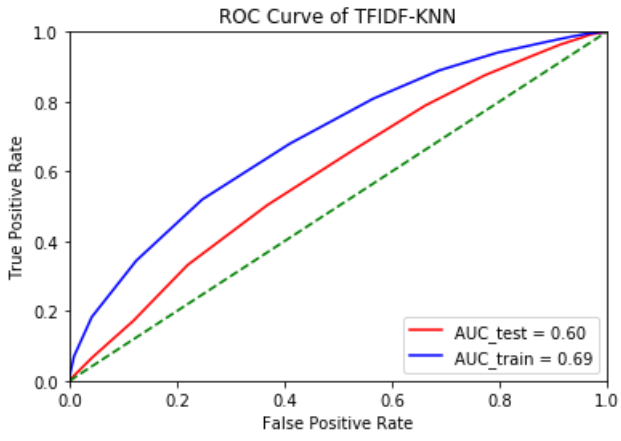
# fitting the model on crossvalidation train

knn.fit(X_train_tfidf, y_train)


#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_test_scores=batch_predict(knn,X_test_tfidf)
pred_tfidf_train_scores=batch_predict(knn,X_train_tfidf)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_test_scores)
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_train_scores)
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF-KNN')
plt.show()
```

```
Dictionary Contents-->
 {1: 0.5356919419419419, 9: 0.5821216181459236, 17: 0.5959066705594483, 25: 0.5945233427872317, 33
: 0.6004861198003559, 41: 0.5963311401679458, 49: 0.5912320740879768}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5356919419419419, 0.5821216181459236, 0.5959066705594483, 0.5945233427872317,
0.6004861198003559, 0.5963311401679458, 0.5912320740879768]
==================================================
Maximum k is 33
==================================================
```

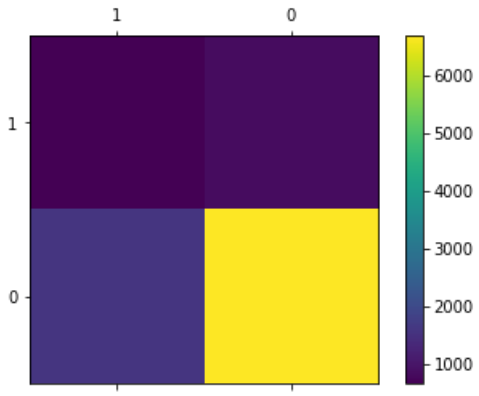


Get the confusion matrix for the TFIDF - KNN

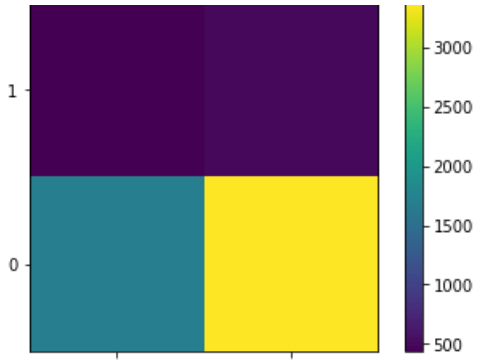
In [71]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF")
labels = [1,0]
cm =confusion_matrix(y_train,predict(pred_tfidf_train_scores,threshold_train,fpr_train,fpr_train))
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels(['']  labels)
ax.set_yticklabels(['']  labels)
plt.show()
print("="*50)
print("Testing CM for TFIDF")
labels = [1,0]
cm =confusion_matrix(y_test,predict(pred_tfidf_test_scores,threshold_test,fpr_test,fpr_test))
summary.append(['Tfidf',find_highest_k(roc_auc_score_cv_tfidf_dict),roc_auc_test])
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels(['']  labels)
ax.set_yticklabels(['']  labels)
plt.show()
```

```
Training CM for TFIDF
the maximum value of tpr*(1-fpr) 0.2457504442383912 for threshold 0.818
[[ 657  854]
 [1587 6702]]
```




```
==================================================
Testing CM for TFIDF
the maximum value of tpr*(1-fpr) 0.2488792689241448 for threshold 0.848
Dictionary Contents-->
 {1: 0.5356919419419419, 9: 0.5821216181459236, 17: 0.5959066705594483, 25: 0.5945233427872317, 33
: 0.6004861198003559, 41: 0.5963311401679458, 49: 0.5912320740879768}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5356919419419419, 0.5821216181459236, 0.5959066705594483, 0.5945233427872317,
0.6004861198003559, 0.5963311401679458, 0.5912320740879768]
==================================================
Maximum k is 33
==================================================
[[ 432  494]
 [1692 3382]]
```



Applying KNN brute force on AVG W2V, SET 3

Hyperparameter tuning for finiding optimal k using ROC_AUC_Score

In [72]:

```
roc_auc_score_cv_w2v_dict={}
roc_auc_score_train_w2v_dict={}

for i in tqdm(range(1,50,8)):
    knn = KNeighborsClassifier(n_neighbors=i, algorithm = 'kd_tree')

     # fitting the model on crossvalidation train
    knn.fit(X_train_w2v, y_train)

    # predict the response on the crossvalidation train
    pred_w2v_cv = batch_predict(knn,X_cv_w2v)

    #evaluate CV roc_auc
    roc_auc_cv =roc_auc_score(y_cv,pred_w2v_cv)

    #insert into dict
    roc_auc_score_cv_w2v_dict[i]=roc_auc_cv

    # predict the response on the train
    pred_w2v_train = batch_predict(knn,X_train_w2v)

    #evaluate train roc_auc
    roc_auc_train =roc_auc_score(y_train,pred_w2v_train)

    #insert into dict
    roc_auc_score_train_w2v_dict[i]=roc_auc_train



print(roc_auc_score_cv_w2v_dict)
print(roc_auc_score_train_w2v_dict)
```

```
100%|██████████| 7/7 [10:45<00:00, 94.21s/it]
```

```
{1: 0.5104896563229897, 9: 0.5671958851212323, 17: 0.5839489663274385, 25: 0.600804580622289, 33:
0.5960800209932154, 41: 0.6021531514153042, 49: 0.5999282702841731}
{1: 1.0, 9: 0.7946376509928919, 17: 0.7426160383032572, 25: 0.716222906790665, 33:
0.7015778607978697, 41: 0.6933584086266803, 49: 0.6870238351018817}
```

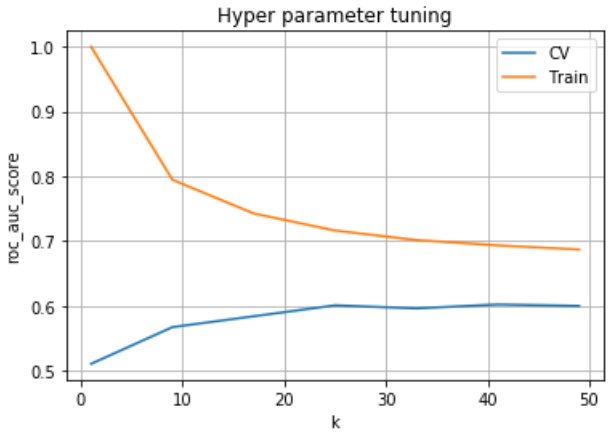
Plot ROC_AUC_score VS different K values (Train and CV set)

In [73]:

```
#https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-
values/37266356

lists1 = sorted(roc_auc_score_cv_w2v_dict.items())
x1, y1 = zip(*lists1) # unpack a list of pairs into two tuples
lists2 = sorted(roc_auc_score_train_w2v_dict.items())
```

```
x2, y2 = zip(*lists2) # unpack a list of pairs into two tuples
plt.xlabel('k')
plt.ylabel('roc_auc_score')
plt.title('Hyper parameter tuning')
plt.plot(x1, y1,label="CV")
plt.plot(x2, y2,label='Train')
plt.legend()
plt.grid()
plt.show()
```



Find the best K value from the result

In [74]:

```
print(find_highest_k(roc_auc_score_cv_w2v_dict))
```

```
Dictionary Contents-->
 {1: 0.5104896563229897, 9: 0.5671958851212323, 17: 0.5839489663274385, 25: 0.600804580622289, 33:
0.5960800209932154, 41: 0.6021531514153042, 49: 0.5999282702841731}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5104896563229897, 0.5671958851212323, 0.5839489663274385, 0.600804580622289,
0.5960800209932154, 0.6021531514153042, 0.5999282702841731]
==================================================
Maximum k is 41
==================================================
41
```

Train the model on the optimal K value and run the Test Dataset

-- Plot the ROC curve for w2v using the test and train Dataset

In [75]:

```
# train model on the best k
knn = KNeighborsClassifier(n_neighbors=find_highest_k(roc_auc_score_cv_w2v_dict), algorithm =
'kd_tree')

# fitting the model on crossvalidation train

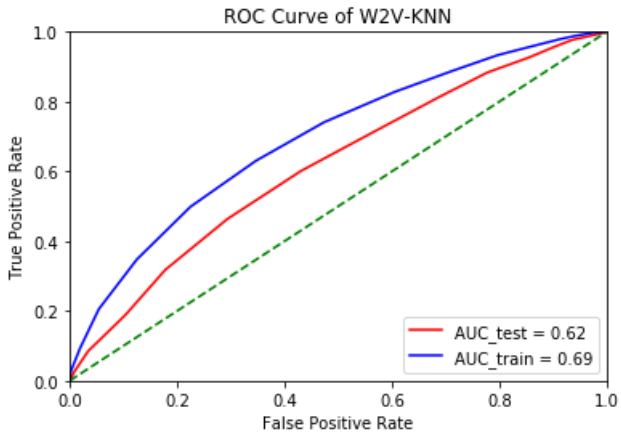
knn.fit(X_train_w2v, y_train)


#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_w2v_test_scores=batch_predict(knn,X_test_w2v)
pred_w2v_train_scores=batch_predict(knn,X_train_w2v)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_w2v_test_scores)
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_w2v_train_scores)
```

```
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of W2V-KNN')
plt.show()
```

```
Dictionary Contents-->
 {1: 0.5104896563229897, 9: 0.5671958851212323, 17: 0.5839489663274385, 25: 0.600804580622289, 33:
0.5960800209932154, 41: 0.6021531514153042, 49: 0.5999282702841731}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5104896563229897, 0.5671958851212323, 0.5839489663274385, 0.600804580622289,
0.5960800209932154, 0.6021531514153042, 0.5999282702841731]
==================================================
Maximum k is 41
==================================================
```



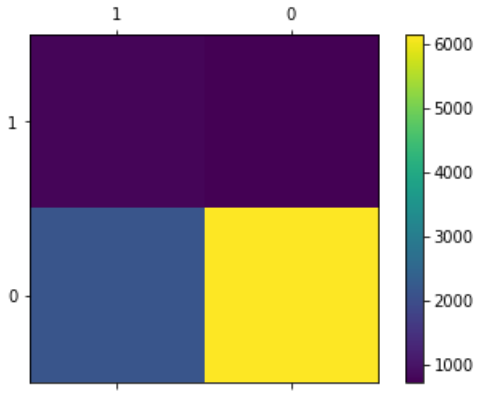
Get the confusion matrix for the W2V - KNN

In [76]:

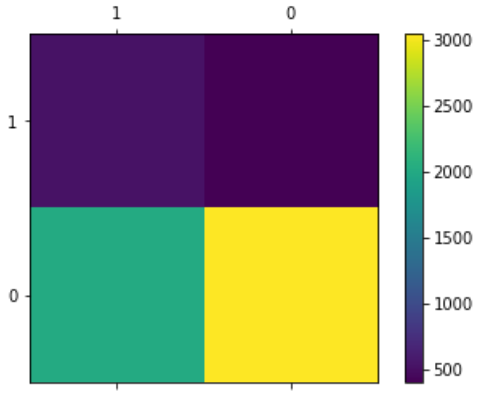
```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for W2V")
labels = [1,0]
cm =confusion_matrix(y_train,predict(pred_w2v_train_scores,threshold_train,fpr_train,fpr_train))
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.show()
print("="*50)
print("Testing CM for W2V")
labels = [1,0]
cm =confusion_matrix(y_test,predict(pred_w2v_test_scores,threshold_test,fpr_test,fpr_test))
summary.append(['W2v',find_highest_k(roc_auc_score_cv_w2v_dict),roc_auc_test])
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
```

```
plt.show()
```

```
Training CM for W2V
the maximum value of tpr*(1-fpr) 0.24931661528232626 for threshold 0.829
[[ 795  716]
 [2146 6143]]
```




```
==================================================
Testing CM for W2V
the maximum value of tpr*(1-fpr) 0.24522318992018433 for threshold 0.854
Dictionary Contents-->
 {1: 0.5104896563229897, 9: 0.5671958851212323, 17: 0.5839489663274385, 25: 0.600804580622289, 33:
0.5960800209932154, 41: 0.6021531514153042, 49: 0.5999282702841731}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5104896563229897, 0.5671958851212323, 0.5839489663274385, 0.600804580622289,
0.5960800209932154, 0.6021531514153042, 0.5999282702841731]
==================================================
Maximum k is 41
==================================================
[[ 527  399]
 [2022 3052]]
```



Applying KNN brute force on TFIDF W2V, SET 4

Hyperparameter tuning for finiding optimal k using ROC_AUC_Score

In [77]:

```
roc_auc_score_cv_tfidf_w2v_dict={}
roc_auc_score_train_tfidf_w2v_dict={}

for i in tqdm(range(1,50,8)):
    knn = KNeighborsClassifier(n_neighbors=i, algorithm = 'kd_tree')

     # fitting the model on crossvalidation train
```

```
    knn.fit(X_train_tfidf_w2v, y_train)

    # predict the response on the crossvalidation train
    pred_tfidf_w2v_cv = batch_predict(knn,X_cv_tfidf_w2v)

    #evaluate CV roc_auc
    roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_w2v_cv)

    #insert into dict
    roc_auc_score_cv_tfidf_w2v_dict[i]=roc_auc_cv

    # predict the response on the train
    pred_tfidf_w2v_train = batch_predict(knn,X_train_tfidf_w2v)

    #evaluate train roc_auc
    roc_auc_train =roc_auc_score(y_train,pred_tfidf_w2v_train)

    #insert into dict
    roc_auc_score_train_tfidf_w2v_dict[i]=roc_auc_train



print(roc_auc_score_cv_tfidf_w2v_dict)
print(roc_auc_score_train_tfidf_w2v_dict)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 7/7 [11
:05<00:00, 96.38s/it]
```

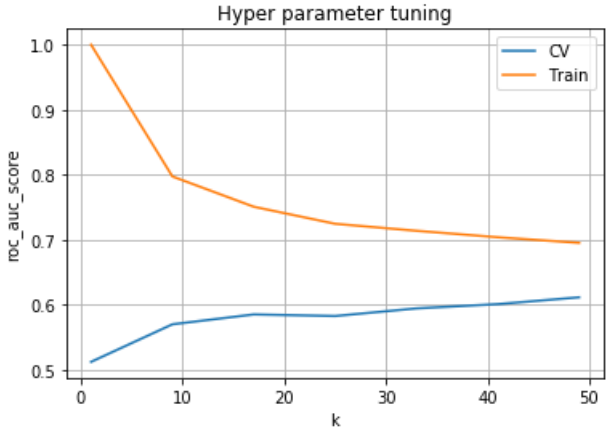
```
{1: 0.5125281531531531, 9: 0.5702584094511177, 17: 0.5854200554721388, 25: 0.582979029376599, 33:
0.5946947381409187, 41: 0.601266413983428, 49: 0.6117449915192972}
{1: 1.0, 9: 0.7974752885882346, 17: 0.7509509425351342, 25: 0.7247604509464873, 33:
0.7139739070358609, 41: 0.7040618366346953, 49: 0.6954426137388431}
```

Plot ROC_AUC_score VS different K values (Train and CV set)

In [78]:

```
#https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-
values/37266356

lists1 = sorted(roc_auc_score_cv_tfidf_w2v_dict.items())
x1, y1 = zip(*lists1) # unpack a list of pairs into two tuples
lists2 = sorted(roc_auc_score_train_tfidf_w2v_dict.items())
x2, y2 = zip(*lists2) # unpack a list of pairs into two tuples
plt.xlabel('k')
plt.ylabel('roc_auc_score')
plt.title('Hyper parameter tuning')
plt.plot(x1, y1,label="CV")
plt.plot(x2, y2,label='Train')
plt.legend()
plt.grid()
plt.show()
```



Find the best K value

In [79]:

```
print(find_highest_k(roc_auc_score_cv_tfidf_w2v_dict))
```

```
Dictionary Contents-->
 {1: 0.5125281531531531, 9: 0.5702584094511177, 17: 0.5854200554721388, 25: 0.582979029376599, 33:
0.5946947381409187, 41: 0.601266413983428, 49: 0.6117449915192972}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5125281531531531, 0.5702584094511177, 0.5854200554721388, 0.582979029376599,
0.5946947381409187, 0.601266413983428, 0.6117449915192972]
==================================================
Maximum k is 49
==================================================
49
```

Train the model on the optimal K value and run the Test Dataset

In [80]:

```
# train model on the best k
knn = KNeighborsClassifier(n_neighbors=find_highest_k(roc_auc_score_cv_tfidf_w2v_dict), algorithm
= 'kd_tree')

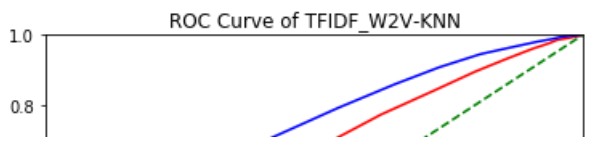
# fitting the model on crossvalidation train

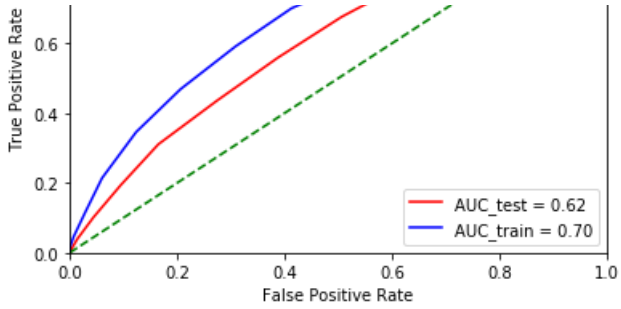
knn.fit(X_train_tfidf_w2v, y_train)


#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_w2v_test_scores=batch_predict(knn,X_test_tfidf_w2v)
pred_tfidf_w2v_train_scores=batch_predict(knn,X_train_tfidf_w2v)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_w2v_test_scores)
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_w2v_train_scores)
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF_W2V-KNN')
plt.show()
```

```
Dictionary Contents-->
 {1: 0.5125281531531531, 9: 0.5702584094511177, 17: 0.5854200554721388, 25: 0.582979029376599, 33:
0.5946947381409187, 41: 0.601266413983428, 49: 0.6117449915192972}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5125281531531531, 0.5702584094511177, 0.5854200554721388, 0.582979029376599,
0.5946947381409187, 0.601266413983428, 0.6117449915192972]
==================================================
Maximum k is 49
==================================================
```



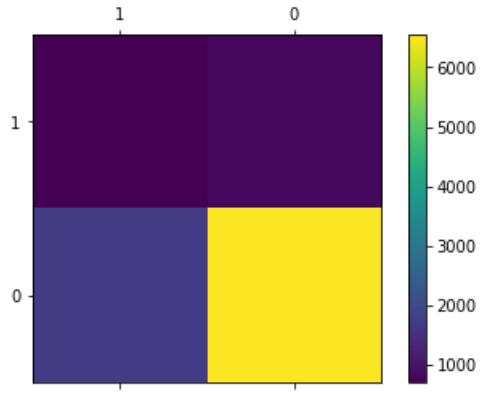


Get the confusion matrix for the TFIDF W2V

In [81]:

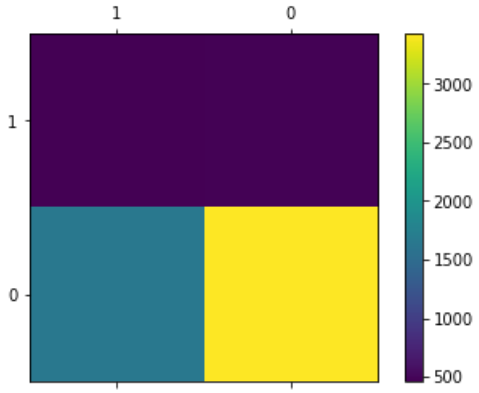
```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF_W2V")
labels = [1,0]
cm
=confusion_matrix(y_train,predict(pred_tfidf_w2v_train_scores,threshold_train,fpr_train,fpr_train)
)
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels(['']  labels)
ax.set_yticklabels(['']  labels)
plt.show()
print("="*50)
print("Testing CM for TFIDF_W2V")
labels = [1,0]
cm =confusion_matrix(y_test,predict(pred_tfidf_w2v_test_scores,threshold_test,fpr_test,fpr_test))
summary.append(['Tfidf_w2v',find_highest_k(roc_auc_score_cv_tfidf_w2v_dict),roc_auc_test])
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels(['']  labels)
ax.set_yticklabels(['']  labels)
plt.show()
```

```
Training CM for TFIDF_W2V
the maximum value of tpr*(1-fpr) 0.24806306805464975 for threshold 0.816
[[ 689  822]
 [1732 6557]]
```




```
==================================================
Testing CM for TFIDF_W2V
the maximum value of tpr*(1-fpr) 0.24995801631765788 for threshold 0.837
Dictionary Contents-->
 {1: 0.5125281531531531, 9: 0.5702584094511177, 17: 0.5854200554721388, 25: 0.582979029376599, 33:
0.5946947381409187, 41: 0.601266413983428, 49: 0.6117449915192972}
==================================================
```

```
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5125281531531531, 0.5702584094511177, 0.5854200554721388, 0.582979029376599,
0.5946947381409187, 0.601266413983428, 0.6117449915192972]
==================================================
Maximum k is 49
==================================================
[[ 457  469]
 [1643 3431]]
```



Feature selection with `SelectKBest`

In [82]:

```
#https://stats.stackexchange.com/questions/341332/how-to-scale-for-selectkbest-for-feature-selecti
on
from sklearn.feature_selection import SelectKBest, f_classif
f_tfidf_ds = SelectKBest(f_classif , k=2000).fit(X_train_tfidf,y_train)
X_train_tfidf_f=f_tfidf_ds.transform(X_train_tfidf)
X_test_tfidf_f=f_tfidf_ds.transform(X_test_tfidf)
X_cv_tfidf_f=f_tfidf_ds.transform(X_cv_tfidf)
```

In [83]:

```
roc_auc_score_cv_tfidf_dict={}
roc_auc_score_train_tfidf_dict={}

for i in tqdm(range(1,50,8)):
    knn = KNeighborsClassifier(n_neighbors=i, algorithm = 'kd_tree')

     # fitting the model on crossvalidation train
    knn.fit(X_train_tfidf_f, y_train)

    # predict the response on the crossvalidation train
    pred_tfidf_cv = batch_predict(knn,X_cv_tfidf_f)

    #evaluate CV roc_auc
    roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_cv)

    #insert into dict
    roc_auc_score_cv_tfidf_dict[i]=roc_auc_cv

    # predict the response on the train
    pred_tfidf_train = batch_predict(knn,X_train_tfidf_f)

    #evaluate train roc_auc
    roc_auc_train =roc_auc_score(y_train,pred_tfidf_train)

    #insert into dict
    roc_auc_score_train_tfidf_dict[i]=roc_auc_train



print(roc_auc_score_cv_tfidf_dict)
print(roc_auc_score_train_tfidf_dict)
```

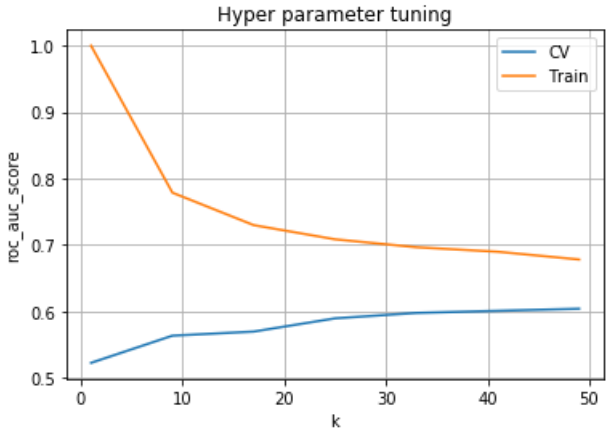
```
100%|██████████████████████████████████████████████████████████████████████████████| 7/7 [14:54<00:00, 134.92s/it]
```

```
{1: 0.5221523606940274, 9: 0.5631128089895452, 17: 0.5692858657268378, 25: 0.5892309410104549, 33: 0.5973827560199089, 41: 0.600560630074519, 49: 0.6037215601017684}
{1: 1.0, 9: 0.778580153631083, 17: 0.7296426918406451, 25: 0.7082781123572108, 33: 0.6963342134357295, 41: 0.6895549578556065, 49: 0.677903920731222}
```

In [84]:

```
#https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-values/37266356

lists1 = sorted(roc_auc_score_cv_tfidf_dict.items())
x1, y1 = zip(*lists1) # unpack a list of pairs into two tuples
lists2 = sorted(roc_auc_score_train_tfidf_dict.items())
x2, y2 = zip(*lists2) # unpack a list of pairs into two tuples
plt.xlabel('k')
plt.ylabel('roc_auc_score')
plt.title('Hyper parameter tuning')
plt.plot(x1, y1,label="CV")
plt.plot(x2, y2,label='Train')
plt.legend()
plt.grid()
plt.show()
```



In [85]:

```
print(find_highest_k(roc_auc_score_cv_tfidf_dict))
```

```
Dictionary Contents-->
 {1: 0.5221523606940274, 9: 0.5631128089895452, 17: 0.5692858657268378, 25: 0.5892309410104549, 33: 0.5973827560199089, 41: 0.600560630074519, 49: 0.6037215601017684}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5221523606940274, 0.5631128089895452, 0.5692858657268378, 0.5892309410104549, 0.5973827560199089, 0.600560630074519, 0.6037215601017684]
==================================================
Maximum k is 49
==================================================
49
```

In [86]:

```
# train model on the best k
knn = KNeighborsClassifier(n_neighbors=find_highest_k(roc_auc_score_cv_tfidf_dict), algorithm = 'kd_tree')

# fitting the model on crossvalidation train

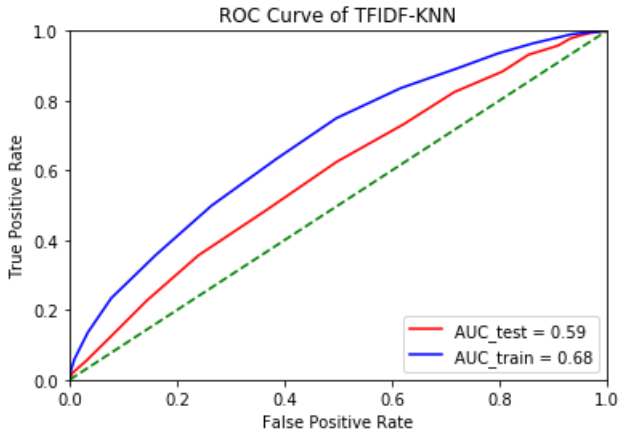
knn.fit(X_train_tfidf_f, y_train)
```

```
knn.fit(X_train_tfidf_f, y_train)


#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-algorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_test_scores=batch_predict(knn,X_test_tfidf_f)
pred_tfidf_train_scores=batch_predict(knn,X_train_tfidf_f)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_test_scores)
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_train_scores)
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF-KNN')
plt.show()
```

```
Dictionary Contents-->
 {1: 0.5221523606940274, 9: 0.5631128089895452, 17: 0.5692858657268378, 25: 0.5892309410104549, 33
: 0.5973827560199089, 41: 0.600560630074519, 49: 0.6037215601017684}
==================================================
Keys in the list are-->
 [1, 9, 17, 25, 33, 41, 49]
==================================================
Values in the list are-->
 [0.5221523606940274, 0.5631128089895452, 0.5692858657268378, 0.5892309410104549,
0.5973827560199089, 0.600560630074519, 0.6037215601017684]
==================================================
Maximum k is 49
==================================================
```

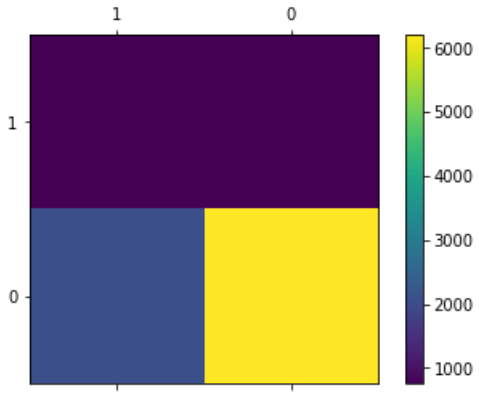


In [87]:

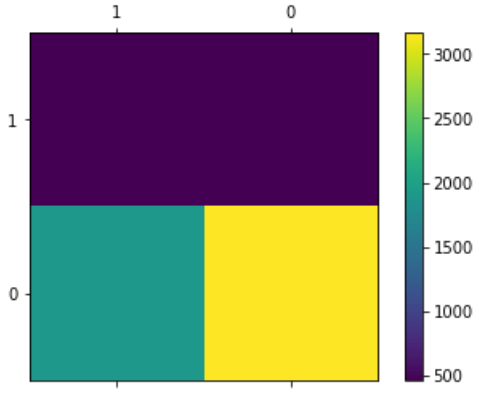
```
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF")
labels = [1,0]
cm =confusion_matrix(y_train,predict(pred_tfidf_train_scores,threshold_train,fpr_train,fpr_train))
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels(['']  labels)
ax.set_yticklabels(['']  labels)
plt.show()
print("="*50)
print("Testing CM for TFIDF")
labels = [1,0]
cm =confusion_matrix(y_test,predict(pred_tfidf_test_scores,threshold_test,fpr_test,fpr_test))
print(cm)
```

```
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
fig.colorbar(cax)
ax.set_xticklabels(['']  + labels)
ax.set_yticklabels(['']  + labels)
plt.show()
```

```
Training CM for TFIDF
the maximum value of tpr*(1-fpr) 0.24998675059271935 for threshold 0.837
[[ 761  750]
 [2076 6213]]
```




```
==================================================
Testing CM for TFIDF
the maximum value of tpr*(1-fpr) 0.24998950407941445 for threshold 0.857
[[ 466  460]
 [1907 3167]]
```



Conclusions

In [88]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Hyper parameter", "AUC"]

for each in summary:
    x.add_row(each)

print(x)
```

```
+------------+-----------------+--------------------+
| Vectorizer | Hyper parameter |        AUC         |
+------------+-----------------+--------------------+
|    BoW     |        49       | 0.6140952988640688 |
|   Tfidf    |        33       | 0.5950531486058176 |
|    W2v     |        41       | 0.616461148224421  |
```

```
| Tfidf_w2v  |        49       |  0.62064650515779  |
+------------+-----------------+--------------------+
```