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`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`

Feature	Description
`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 50K points as system becomes Unresponsive with higher no of point
s

project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
project_data= project_data.sample(n=50000, 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,"%")
```

```
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    42460
0     7540
Name: project_is_approved, dtype: int64
Number of projects approved for funding  42460 = 84.92 %
Number of projects not approved for funding  7540 = 15.079999999999998 %
```

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 (50000, 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
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 = []
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
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', "do
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%|████████████████████████████████████████████████████████████████████████████| 50000/50000
[00:34<00:00, 1459.01it/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	project_grade
75155	144107	p064182	7414165942b20a8d7fe5bcdc96244624	Ms.	NY	2017-02-05 17:49:01	
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	

Unnamed:

Preprocessing of `project_title`

In [22]:

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

```
100%|████████████████████████████████████████████████████████████████████████| 50000/50000
[00:01<00:00, 28991.31it/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-panda
s
X=project_data.loc[:, project_data.columns != 'project_is_approved']
y=project_data['project_is_approved']
X.shape
```

Out[29]:

```
(50000, 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)

# intentionally taking less data in cv ; will not use it
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)
```

```
(24500, 13)
(15000, 13)
(10500, 13)
(24500,)
(15000,)
(10500,)
```

In [31]:

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

Out[31]:

	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	teacher_number_of_previously_posted_projects	cle
0	Ms.	NY	2017-02-05 17:49:01	3_5	7	Lite
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)

feature_names_tfidf=[]
feature_names_tfidf_w2v=[]
feature_names_tfidf.extend(vectorizer.get_feature_names())
feature_names_tfidf_w2v.extend(vectorizer.get_feature_names())
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix after one hot encodig  (24500, 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)


feature_names_tfidf.extend(vectorizer.get_feature_names())
feature_names_tfidf_w2v.extend(vectorizer.get_feature_names())
```

```
['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  (24500, 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)


feature_names_tfidf.extend(vectorizer.get_feature_names())
feature_names_tfidf_w2v.extend(vectorizer.get_feature_names())
```

```
['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  (24500, 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)

feature_names_tfidf.extend(vectorizer.get_feature_names())
feature_names_tfidf_w2v.extend(vectorizer.get_feature_names())
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (24500, 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)


feature_names_tfidf.extend(vectorizer.get_feature_names())
feature_names_tfidf_w2v.extend(vectorizer.get_feature_names())
```

```
['3_5', '6_8', '9_12', 'PreK_2']
Shape of matrix after one hot encodig  (24500, 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))


feature_names_tfidf.append('price')
feature_names_tfidf_w2v.append('price')
```

```
Mean : 299.1188355102041, Standard deviation : 364.2361296208864
```

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))

feature_names_tfidf.append('quantity')
feature_names_tfidf_w2v.append('quantity')
```

```
Mean : 17.013551020408162, Standard deviation : 26.683954898457372
```

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
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))

feature_names_tfidf.append('essay_count')
feature_names_tfidf_w2v.append('essay_count')
```

```
Mean : 1015.7230612244898, Standard deviation : 278.40201060516114
```

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))

feature_names_tfidf.append('title_count')
feature_names_tfidf_w2v.append('title_count')
```

```
Mean : 25.659551020408163, Standard deviation : 11.612785556008768
```

Vectorizing Text data

TFIDF vectorizer - essay

In [41]:

```
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)
feature_names_tfidf.extend(vectorizer.get_feature_names())
```

```
Shape of matrix after one hot encodig  (24500, 5000)
```

TFIDF vectorizer - cleaned tittle

In [42]:

```
# 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)

feature_names_tfidf.extend(vectorizer.get_feature_names())
```

```
Shape of matrix after one hot encodig  (24500, 2098)
```

Using Pretrained Models: Avg W2V

In [43]:

```
'''
# 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[43]:

```
'\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]\n
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 [44]:

```
# 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 [45]:

```
# 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%|█████████████████████████████████████████████████████████████████████████| 24500/24500
[00:09<00:00, 2711.09it/s]
100%|█████████████████████████████████████████████████████████████████████████| 15000/15000
[00:05<00:00, 2773.93it/s]
100%|█████████████████████████████████████████████████████████████████████████| 10500/10500
[00:03<00:00, 2757.12it/s]
100%|████████████████████████████████████████████████████████████████████████| 24500/24500
[00:00<00:00, 45214.46it/s]
100%|████████████████████████████████████████████████████████████████████████| 15000/15000
[00:00<00:00, 45510.16it/s]
100%|████████████████████████████████████████████████████████████████████████| 10500/10500
[00:00<00:00, 44937.87it/s]
```

Using Pretrained Models: TFIDF weighted W2V - Essay

In [46]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
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))
```

```
31565
```

In [47]:

```
# 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'])

feature_names_tfidf_w2v.extend(tfidf_model.get_feature_names())
```

```
100%|████████████████████████████████████████████████████████████████████████████| 24500/24500 [01:
05<00:00, 373.93it/s]
100%|████████████████████████████████████████████████████████████████████████████| 15000/15000 [00:
39<00:00, 377.05it/s]
100%|████████████████████████████████████████████████████████████████████████████| 10500/10500 [00:
28<00:00, 376.82it/s]
```

Using Pretrained Models: TFIDF weighted W2V - Cleaned Title

In [48]:

```
# 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))
```

```
8267
```

In [49]:

```
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'])
feature_names_tfidf_w2v.extend(tfidf_model.get_feature_names())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24500/24500
[00:01<00:00, 21088.38it/s]
100%|██████████████████████████████████████████████████████████████████████████| 15000/15000
```

```
[00:00<00:00, 22729.27it/s]
100%|█████████████████████████████████████████████████████████████| 10500/10500
[00:00<00:00, 22044.10it/s]
```

In [50]:

```
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]

feature_names_tfidf.append("previously_posted ")
```

Merging all the above features

TFIDF

In [51]:

```
# 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_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_standardizec
,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 [52]:

```
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 [53]:

```
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)
```

```
(24500, 7202)
(15000, 7202)
```

TFIDF- WORD2VEC

In [54]:

```
# 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_standardizec
,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)
```

```
(24500, 704)
(15000, 704)
```

Apply DECISION TREE on TFIDF

Train model for various values

In [55]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import roc_auc_score

roc_auc_score_cv_tfidf_dict=[] # TO STORE DATA FOR CV HYPER PARMATERS AND AUC SCORE
roc_auc_score_train_tfidf_dict=[] # TO STORE DATA FOR TRAIN HYPER PARMATERS AND AUC SCORE

depth=[1, 10, 50, 100]
min_samples_split=[5, 10, 100, 500]

for d in tqdm(depth):
    for s in min_samples_split:
        #create instance of model
        dt=DecisionTreeClassifier(max_depth=d,min_samples_split=s)

        #Fit the model on the training set
        dt.fit(X_train_tfidf,y_train)


        # predict the response on the crossvalidation train
        pred_tfidf_cv = dt.predict_proba(X_cv_tfidf)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_cv[:,1])

        #insert into CV dictionary == Depth, Samples_Split and roc_auc scores
        roc_auc_score_cv_tfidf_dict.append([d,s,roc_auc_cv])

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

        # predict the response on the train
        pred_tfidf_train = dt.predict_proba(X_train_tfidf)
        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_tfidf_train[:,1])

        #insert into TRAIN dictionary == Depth, Samples_Split and roc_auc scores
        roc_auc_score_train_tfidf_dict.append([d,s,roc_auc_train])

print(roc_auc_score_cv_tfidf_dict)
```

```
100%|███████████████████████████████████████████████████████████████████████████████████| 4/4 [53:
10<00:00, 814.18s/it]
```

```
[[1, 5, 0.5505768754891305], [1, 10, 0.5505768754891305], [1, 100, 0.5505768754891305], [1, 500, 0
.5505768754891305], [10, 5, 0.6464162621086682], [10, 10, 0.6465636521153778], [10, 100,
0.653736278224159], [10, 500, 0.6621118632413433], [50, 5, 0.5107913146657271], [50, 10,
0.5258125914634513], [50, 100, 0.5579842771241004], [50, 500, 0.6176927091572586], [100, 5,
0.49748282238721364], [100, 10, 0.5100760427586167], [100, 100, 0.5346885444774584], [100, 500,
0.5968837622402601]]
```

3D Scatter Plot

In [56]:

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

x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in tqdm(roc_auc_score_cv_tfidf_dict):
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in tqdm(roc_auc_score_train_tfidf_dict):
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross Val')

trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Train')

data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(TFIDF)',scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='min_samples_split'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)

offline.iplot(fig, filename='3d-scatter-colorscale')
```

```
100%|███████████████████████████████████████████████████████████████████████████████| 16/16
[00:00<00:00, 31956.60it/s]
100%|███████████████████████████████████████████████████████████████████████████████| 16/16
[00:00<00:00, 15993.53it/s]
```

Find best Hyper-Parameter value to train model

In [57]:

```
from numpy import array

def find_best_params(input_list):
    optimal={}

    temp=pd.DataFrame(input_list)

    print(temp)
    print("="*50)

    print("Max auc score==>", max(temp[2]))
    print("*"*50)

    print("temp[2]==>",temp[2])
    print("@"*50)

    print("temp[temp[2]==max(temp[2])]==>",temp[temp[2]==max(temp[2])])
    print("^"*50)

    print("temp[temp[2]==max(temp[2])].iloc[0][0]==>",temp[temp[2]==max(temp[2])].iloc[0][0])
    print("#"*50)

    optimal_depth=int(temp[temp[2]==max(temp[2])].iloc[0][0])

    optimal_sample=int(temp[temp[2]==max(temp[2])].iloc[0][1])

    optimal['depth']=optimal_depth

    optimal['sample']=optimal_sample

    return optimal
```

In [58]:

```
find_best_params(roc_auc_score_cv_tfidf_dict)
```

```
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4    10    5  0.646416
5    10   10  0.646564
```

```
6    10  100  0.653736
7    10  500  0.662112
8    50    5  0.510791
9    50   10  0.525813
10   50  100  0.557984
11   50  500  0.617693
12  100    5  0.497483
13  100   10  0.510076
14  100  100  0.534689
15  100  500  0.596884
==================================================
Max auc score==> 0.6621118632413433
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.646416
5     0.646564
6     0.653736
7     0.662112
8     0.510791
9     0.525813
10    0.557984
11    0.617693
12    0.497483
13    0.510076
14    0.534689
15    0.596884
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0    1         2
7  10  500  0.662112
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
```

Out[58]:

```
{'depth': 10, 'sample': 500}
```

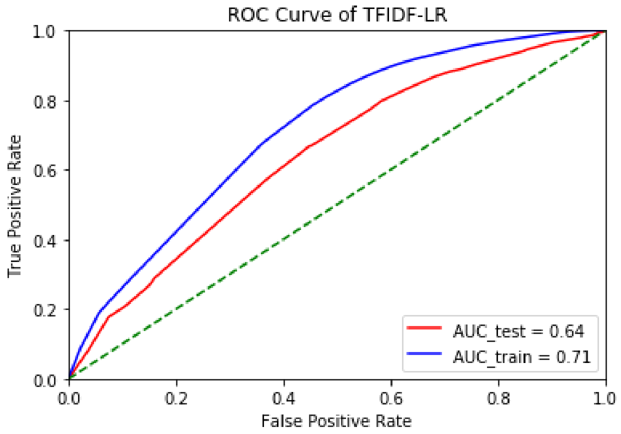
Use best Hyper-Parameter value to train model

In [59]:

```
# train model on the best alpha
model = DecisionTreeClassifier(max_depth=find_best_params(roc_auc_score_cv_tfidf_dict)['depth'],mi
n_samples_split=find_best_params(roc_auc_score_cv_tfidf_dict)['sample'])

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

# predict the response on the crossvalidation train
pred_tfidf_test = model.predict(X_test_tfidf)
pred_tfidf_train = model.predict(X_train_tfidf)

#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=model.predict_proba(X_test_tfidf)
pred_tfidf_train_scores=model.predict_proba(X_train_tfidf)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_train_scores[:, 1])

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-LR')
plt.show()
```

```
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4    10    5  0.646416
5    10   10  0.646564
6    10  100  0.653736
7    10  500  0.662112
8    50    5  0.510791
9    50   10  0.525813
10   50  100  0.557984
11   50  500  0.617693
12  100    5  0.497483
13  100   10  0.510076
14  100  100  0.534689
15  100  500  0.596884
==================================================
Max auc score==> 0.6621118632413433
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.646416
5     0.646564
6     0.653736
7     0.662112
8     0.510791
9     0.525813
10    0.557984
11    0.617693
12    0.497483
13    0.510076
14    0.534689
15    0.596884
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0    1         2
7  10  500  0.662112
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4    10    5  0.646416
5    10   10  0.646564
6    10  100  0.653736
7    10  500  0.662112
8    50    5  0.510791
9    50   10  0.525813
10   50  100  0.557984
11   50  500  0.617693
12  100    5  0.497483
13  100   10  0.510076
14  100  100  0.534689
15  100  500  0.596884
==================================================
Max auc score==> 0.6621118632413433
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.646416
5     0.646564
```

```
6      0.653736
7      0.662112
8      0.510791
9      0.525813
10     0.557984
11     0.617693
12     0.497483
13     0.510076
14     0.534689
15     0.596884
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0    1         2
7  10  500  0.662112
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
```

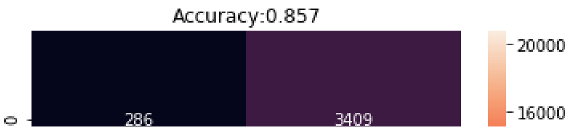


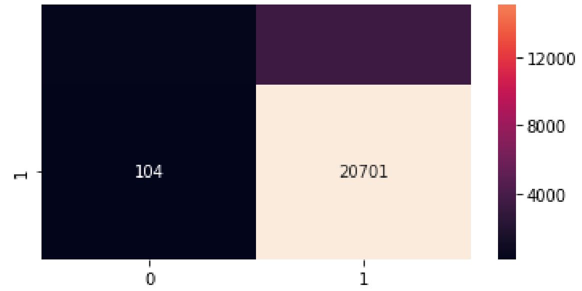
Plot Confusion Matrix

In [60]:

```
from sklearn.metrics import accuracy_score
summary = []
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix

print("Training CM for TFIDF")
cm =confusion_matrix(y_train, pred_tfidf_train, labels=None, sample_weight=None)

sns.heatmap(cm, annot=True,fmt="d")

plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_train, pred_tfidf_train)))

plt.show()

print("="*50)

cm =confusion_matrix(y_test, pred_tfidf_test, labels=None, sample_weight=None)

summary.append(['Tfidf',find_best_params(roc_auc_score_cv_tfidf_dict)['depth'],find_best_params(ro
c_auc_score_cv_tfidf_dict)['sample'],roc_auc_test])

print("="*50)
print("Testing CM for TFIDF")
sns.heatmap(cm, annot=True,fmt="d")
plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_test, pred_tfidf_test)))
plt.show()
```

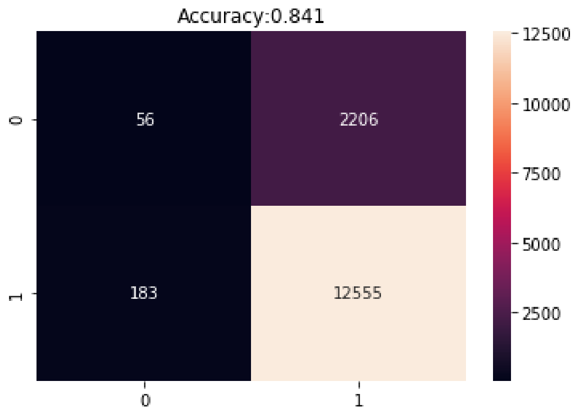
```
Training CM for TFIDF
```





```
==================================================
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4    10    5  0.646416
5    10   10  0.646564
6    10  100  0.653736
7    10  500  0.662112
8    50    5  0.510791
9    50   10  0.525813
10   50  100  0.557984
11   50  500  0.617693
12  100    5  0.497483
13  100   10  0.510076
14  100  100  0.534689
15  100  500  0.596884
==================================================
Max auc score==> 0.6621118632413433
**************************************************
temp[2]==> 0      0.550577
1      0.550577
2      0.550577
3      0.550577
4      0.646416
5      0.646564
6      0.653736
7      0.662112
8      0.510791
9      0.525813
10     0.557984
11     0.617693
12     0.497483
13     0.510076
14     0.534689
15     0.596884
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0    1         2
7  10  500  0.662112
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4    10    5  0.646416
5    10   10  0.646564
6    10  100  0.653736
7    10  500  0.662112
8    50    5  0.510791
9    50   10  0.525813
10   50  100  0.557984
11   50  500  0.617693
12  100    5  0.497483
13  100   10  0.510076
14  100  100  0.534689
15  100  500  0.596884
==================================================
Max auc score==> 0.6621118632413433
**************************************************
temp[2]==> 0      0.550577
```

```
temp[2] ==> 0     0.550577
1      0.550577
2      0.550577
3      0.550577
4      0.646416
5      0.646564
6      0.653736
7      0.662112
8      0.510791
9      0.525813
10     0.557984
11     0.617693
12     0.497483
13     0.510076
14     0.534689
15     0.596884
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0    1          2
7  10  500  0.662112
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
==================================================
Testing CM for TFIDF
```



Plot the Tree

In [61]:

```
from sklearn.tree import export_graphviz
# train model on the best alpha
lr =
DecisionTreeClassifier(max_depth=2,min_samples_split=find_best_params(roc_auc_score_cv_tfidf_dict)
['sample'])

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

dot_data = export_graphviz(lr,
                           feature_names=feature_names_tfidf,
                           class_names=["+","-"],
                           out_file='Tfidf_tree.dot',
                           filled=True,
                           rounded=True)
```

```
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4    10    5  0.646416
5    10   10  0.646564
6    10  100  0.653736
7    10  500  0.662112
8    50    5  0.510791
9    50   10  0.525813
10   50  100  0.557984
```

```
10   50  100  0.557984
11   50  500  0.617693
12  100    5  0.497483
13  100   10  0.510076
14  100  100  0.534689
15  100  500  0.596884
==================================================
Max auc score==> 0.6621118632413433
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.646416
5     0.646564
6     0.653736
7     0.662112
8     0.510791
9     0.525813
10    0.557984
11    0.617693
12    0.497483
13    0.510076
14    0.534689
15    0.596884
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0    1         2
7  10  500  0.662112
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
```

Apply SVM on TFIDFW2V

Train model for various values

In [62]:

```
roc_auc_score_cv_tfidf_w2v_dict=[]
roc_auc_score_train_tfidf_w2v_dict=[]

depth=[5, 10, 50, 100]
min_samples_split=[5, 10, 100, 500]

for d in tqdm(depth):
    for s in min_samples_split:
        #create instance of model
        dt=DecisionTreeClassifier(max_depth=d,min_samples_split=s)

        #Fit the model on the training set
        dt.fit(X_train_tfidf_w2v,y_train)


        # predict the response on the crossvalidation train
        pred_tfidf_w2v_cv = dt.predict_proba(X_cv_tfidf_w2v)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_w2v_cv[:,1])

        #insert into dict
        roc_auc_score_cv_tfidf_w2v_dict.append([d,s,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_tfidf_w2v, y_train)

        # predict the response on the train
        pred_tfidf_w2v_train = dt.predict_proba(X_train_tfidf_w2v)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_tfidf_w2v_train[:,1])

        #insert into dict
        roc_auc_score_train_tfidf_w2v_dict.append([d,s,roc_auc_train])
        roc_auc_score_train_tfidf_w2v_dict.append([d,s,roc_auc_train])

print(roc_auc_score_cv_tfidf_w2v_dict)
```

```
100%|██████████████████████████████████████████████████████████████████████████████████| 4/4 [14:
28<00:00, 213.53s/it]
```

```
[[5, 5, 0.6516413635938253], [5, 10, 0.6515225235379467], [5, 100, 0.650984147976308], [5, 500, 0.
6512533888897901], [10, 5, 0.6222956625823708], [10, 10, 0.6238328967835681], [10, 100,
0.6337199997931369], [10, 500, 0.6473760505301541], [50, 5, 0.5341185018487686], [50, 10,
0.545380890703208], [50, 100, 0.590941263541479], [50, 500, 0.6337681025638919], [100, 5,
0.5400041131765391], [100, 10, 0.5502895694702836], [100, 100, 0.5851024089570052], [100, 500,
0.6327460072397857]]
```

3D Scatter Plot

In [63]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_tfidf_w2v_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_tfidf_w2v_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(TFIDF_W2V)',scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='min_samples_split'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

Find best Hyper-Parameter value to train model

In [64]:

```
find_best_params(roc_auc_score_cv_tfidf_w2v_dict)
```

```
      0    1         2
0     5    5  0.651641
1     5   10  0.651523
2     5  100  0.650984
3     5  500  0.651253
4    10    5  0.622296
5    10   10  0.623833
6    10  100  0.633720
7    10  500  0.647376
8    50    5  0.534119
9    50   10  0.545381
10   50  100  0.590941
11   50  500  0.633768
12  100    5  0.540004
13  100   10  0.550290
14  100  100  0.585102
15  100  500  0.632746
==================================================
Max auc score==> 0.6516413635938253
**************************************************
temp[2]==> 0     0.651641
1     0.651523
2     0.650984
3     0.651253
4     0.622296
5     0.623833
6     0.633720
7     0.647376
8     0.534119
9     0.545381
10    0.590941
11    0.633768
12    0.540004
13    0.550290
14    0.585102
15    0.632746
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>    0  1         2
0  5  5  0.651641
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```

Out[64]:

```
{'depth': 5, 'sample': 5}
```

Use best Hyper-Parameter value to train model

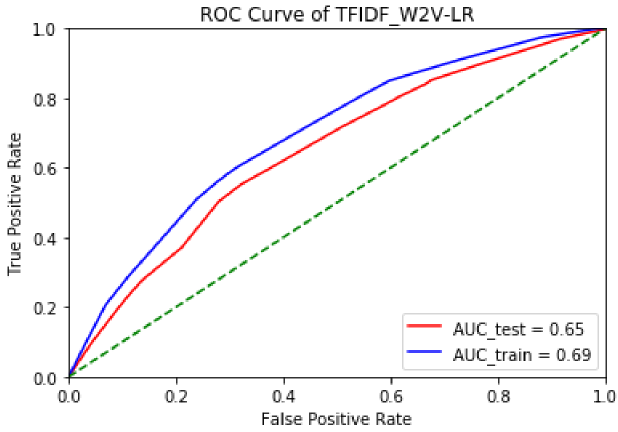
In [65]:

```
# train model on the best alpha
lr = DecisionTreeClassifier(max_depth=find_best_params(roc_auc_score_cv_tfidf_w2v_dict)['depth'],m
in_samples_split=find_best_params(roc_auc_score_cv_tfidf_w2v_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_tfidf_w2v, y_train)
```

```
# predict the response on the crossvalidation train
pred_tfidf_w2v_test = lr.predict(X_test_tfidf_w2v)
pred_tfidf_w2v_train = lr.predict(X_train_tfidf_w2v)

#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=lr.predict_proba(X_test_tfidf_w2v)
pred_tfidf_w2v_train_scores=lr.predict_proba(X_train_tfidf_w2v)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_w2v_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_w2v_train_scores[:, 1])

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-LR')
plt.show()
```

```
      0    1         2
0     5    5  0.651641
1     5   10  0.651523
2     5  100  0.650984
3     5  500  0.651253
4    10    5  0.622296
5    10   10  0.623833
6    10  100  0.633720
7    10  500  0.647376
8    50    5  0.534119
9    50   10  0.545381
10   50  100  0.590941
11   50  500  0.633768
12  100    5  0.540004
13  100   10  0.550290
14  100  100  0.585102
15  100  500  0.632746
==================================================
Max auc score==> 0.6516413635938253
**************************************************
temp[2]==> 0     0.651641
1     0.651523
2     0.650984
3     0.651253
4     0.622296
5     0.623833
6     0.633720
7     0.647376
8     0.534119
9     0.545381
10    0.590941
11    0.633768
12    0.540004
13    0.550290
14    0.585102
15    0.632746
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>    0  1         2
0  5  5  0.651641
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
      0    1         2
0     5    5  0.651641
1     5   10  0.651523
2     5  100  0.650984
3     5  500  0.651253
```

```
4    10    5  0.622296
5    10   10  0.623833
6    10  100  0.633720
7    10  500  0.647376
8    50    5  0.534119
9    50   10  0.545381
10   50  100  0.590941
11   50  500  0.633768
12  100    5  0.540004
13  100   10  0.550290
14  100  100  0.585102
15  100  500  0.632746
==================================================
Max auc score==> 0.6516413635938253
**************************************************
temp[2]==> 0     0.651641
1     0.651523
2     0.650984
3     0.651253
4     0.622296
5     0.623833
6     0.633720
7     0.647376
8     0.534119
9     0.545381
10    0.590941
11    0.633768
12    0.540004
13    0.550290
14    0.585102
15    0.632746
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>    0  1         2
0  5  5  0.651641
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```



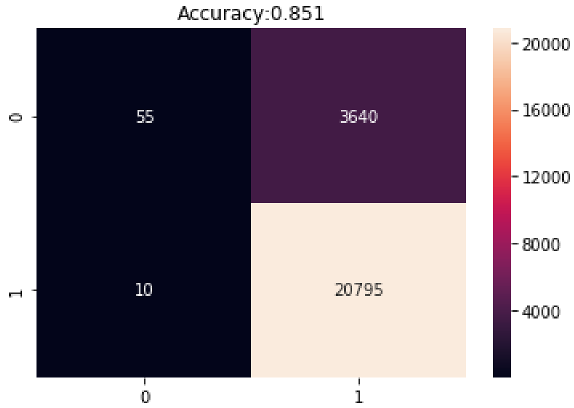
Plot Confusion Matrix

In [66]:

```
from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF_W2V")
cm =confusion_matrix(y_train, pred_tfidf_w2v_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_train, pred_tfidf_w2v_train)))
plt.show()

print("="*50)
print("Testing CM for TFIDF_W2V")

cm =confusion_matrix(y_test, pred_tfidf_w2v_test, labels=None, sample_weight=None)
```

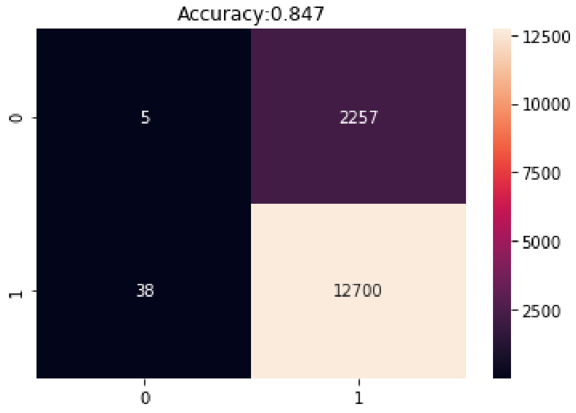
```
cm= confusion_matrix(y_test, pred_tfidf_w2v_test, labels=None, sample_weight=None)
summary.append(['Tfidf_w2v',find_best_params(roc_auc_score_cv_tfidf_w2v_dict)
['depth'],find_best_params(roc_auc_score_cv_tfidf_w2v_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")

print("="*50)
print("Testing CM for TFIDF_W2V")
plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_test, pred_tfidf_w2v_test)))
plt.show()
```

```
Training CM for TFIDF_W2V
```



```
==================================================
Testing CM for TFIDF_W2V
      0    1         2
0     5    5  0.651641
1     5   10  0.651523
2     5  100  0.650984
3     5  500  0.651253
4    10    5  0.622296
5    10   10  0.623833
6    10  100  0.633720
7    10  500  0.647376
8    50    5  0.534119
9    50   10  0.545381
10   50  100  0.590941
11   50  500  0.633768
12  100    5  0.540004
13  100   10  0.550290
14  100  100  0.585102
15  100  500  0.632746
==================================================
Max auc score==> 0.6516413635938253
**************************************************
temp[2]==> 0     0.651641
1     0.651523
2     0.650984
3     0.651253
4     0.622296
5     0.623833
6     0.633720
7     0.647376
8     0.534119
9     0.545381
10    0.590941
11    0.633768
12    0.540004
13    0.550290
14    0.585102
15    0.632746
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
0  5  5  0.651641
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
      0    1         2
0     5    5  0.651641
1     5   10  0.651523
```

```
1      5    10   0.651523
2      5   100   0.650984
3      5   500   0.651253
4     10     5   0.622296
5     10    10   0.623833
6     10   100   0.633720
7     10   500   0.647376
8     50     5   0.534119
9     50    10   0.545381
10    50   100   0.590941
11    50   500   0.633768
12   100     5   0.540004
13   100    10   0.550290
14   100   100   0.585102
15   100   500   0.632746
==================================================
Max auc score==> 0.6516413635938253
**************************************************
temp[2]==> 0      0.651641
1      0.651523
2      0.650984
3      0.651253
4      0.622296
5      0.623833
6      0.633720
7      0.647376
8      0.534119
9      0.545381
10     0.590941
11     0.633768
12     0.540004
13     0.550290
14     0.585102
15     0.632746
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
0  5  5  0.651641
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
==================================================
Testing CM for TFIDF_W2V
```



New Set

Select Best 5K features from set 2

In [68]:

```
print(len(feature_names_tfidf))
print(X_train_tfidf.shape)
```

```
7202
(24500, 7202)
```

In [69]:

```
DT = DecisionTreeClassifier(random_state=0)
tfidf_pd_train = pd.DataFrame(X_train_tfidf, columns=feature_names_tfidf)
tfidf_pd_test = pd.DataFrame(X_test_tfidf, columns=feature_names_tfidf)
tfidf_pd_cv = pd.DataFrame(X_cv_tfidf, columns=feature_names_tfidf)
DT = DT.fit(tfidf_pd_train, y_train)

temp=dict(zip(tfidf_pd_train.columns, DT.feature_importances_))
```

Feature Importnace

In [70]:

```
temp = {k:v for k,v in temp.items() if v != 0}
print(len(temp))
```

```
1058
```

Pick top 5000

In [71]:

```
imp_features_train=[]
temp=Counter(temp)
for k, v in temp.most_common(5000):
    imp_features_train.append(k)
```

In [72]:

```
imp_tfidf_df_train=tfidf_pd_train[imp_features_train]
imp_tfidf_df_test=tfidf_pd_test[imp_features_train]
imp_tfidf_df_cv=tfidf_pd_cv[imp_features_train]
print(imp_tfidf_df_train.shape)
print(imp_tfidf_df_test.shape)
print(imp_tfidf_df_cv.shape)
```

```
(24500, 1185)
(15000, 1185)
(10500, 1185)
```

Apply Decision Tree on New Set

In [73]:

```
depth=[1, 5, 10, 50, 100]
min_samples_split=[5, 10, 100, 500]


roc_auc_score_cv_imp_tfidf_dict=[]
roc_auc_score_train_imp_tfidf_dict=[]
for d in tqdm(depth):
    for s in min_samples_split:
        #create instance of model
        dt=DecisionTreeClassifier(max_depth=d,min_samples_split=s)

        #Fit the model on the training set
        dt.fit(imp_tfidf_df_train,y_train)


        # predict the response on the crossvalidation train
        pred_imp_tfidf_cv = dt.predict_proba(imp_tfidf_df_cv)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_imp_tfidf_cv[:,1])
        #insert into dict
        roc_auc_score_cv_imp_tfidf_dict.append([d,s,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(imp_tfidf_df_train, y_train)

        # predict the response on the train
        pred_imp_tfidf_train = dt.predict_proba(imp_tfidf_df_train)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_imp_tfidf_train[:,1])

        #insert into dict
        roc_auc_score_train_imp_tfidf_dict.append([d,s,roc_auc_train])

print(roc_auc_score_cv_imp_tfidf_dict)
```

```
100%|█████████████████████████████████████████████████████████████████████| 5/5 [09:
56<00:00, 148.11s/it]
```

```
[[1, 5, 0.5505768754891305], [1, 10, 0.5505768754891305], [1, 100, 0.5505768754891305], [1, 500, 0
.5505768754891305], [5, 5, 0.6471222535106698], [5, 10, 0.6465990384688272], [5, 100,
0.6466689964748956], [5, 500, 0.6470814122038359], [10, 5, 0.6443323282286542], [10, 10,
0.6447465858899059], [10, 100, 0.64551761167122], [10, 500, 0.6458500096099277], [50, 5,
0.49854501516087396], [50, 10, 0.5110775580313172], [50, 100, 0.5442055607794802], [50, 500,
0.6075825552291007], [100, 5, 0.49093383205303687], [100, 10, 0.5212842717187375], [100, 100,
0.5473835316090816], [100, 500, 0.5896231130200457]]
```

3D Scatter Plot

In [74]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_imp_tfidf_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_imp_tfidf_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(IMP_TFIDF)',scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='min_samples_split'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)

offline.iplot(fig, filename='3d-scatter-colorscale')
```

Find best Hper-parameter to train the model

In [75]:

```
find_best_params(roc_auc_score_cv_imp_tfidf_dict)
```

```
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4     5    5  0.647122
5     5   10  0.646599
6     5  100  0.646669
7     5  500  0.647081
8    10    5  0.644332
9    10   10  0.644747
10   10  100  0.645518
11   10  500  0.645850
12   50    5  0.498545
13   50   10  0.511078
14   50  100  0.544206
15   50  500  0.607583
16  100    5  0.490934
17  100   10  0.521284
18  100  100  0.547384
19  100  500  0.589623
==================================================
Max auc score==> 0.6471222535106698
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.647122
5     0.646599
6     0.646669
7     0.647081
8     0.644332
9     0.644747
10    0.645518
11    0.645850
12    0.498545
13    0.511078
14    0.544206
15    0.607583
16    0.490934
17    0.521284
18    0.547384
19    0.589623
Name: 2, dtype: float64
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
4  5  5  0.647122
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```

Out[75]:

```
{'depth': 5, 'sample': 5}
```

Use best Hper-parameter to train the model

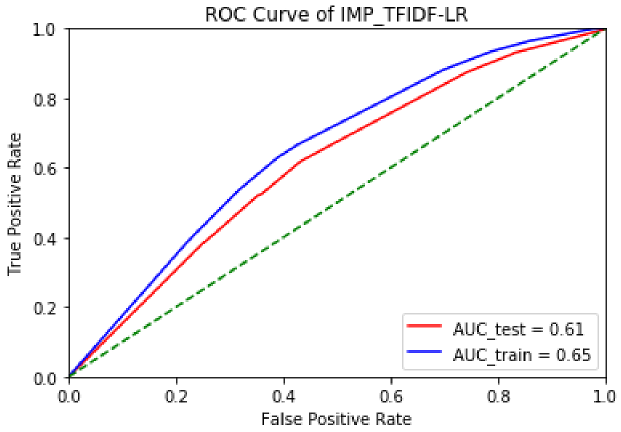
In [76]:

```
# train model on the best alpha
lr = DecisionTreeClassifier(max_depth=find_best_params(roc_auc_score_cv_imp_tfidf_dict)['depth'],m
in_samples_split=find_best_params(roc_auc_score_cv_imp_tfidf_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(imp_tfidf_df_train, y_train)

# predict the response on the crossvalidation train
pred_imp_tfidf_test = lr.predict(imp_tfidf_df_test)
pred_imp_tfidf_train = lr.predict(imp_tfidf_df_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_imp_tfidf_test_scores=lr.predict_proba(imp_tfidf_df_test)
pred_imp_tfidf_train_scores=lr.predict_proba(imp_tfidf_df_train)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_imp_tfidf_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_imp_tfidf_train_scores[:, 1])

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 IMP_TFIDF-LR')
plt.show()
```

```
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4     5    5  0.647122
5     5   10  0.646599
6     5  100  0.646669
7     5  500  0.647081
8    10    5  0.644332
9    10   10  0.644747
10   10  100  0.645518
11   10  500  0.645850
12   50    5  0.498545
13   50   10  0.511078
14   50  100  0.544206
15   50  500  0.607583
16  100    5  0.490934
17  100   10  0.521284
18  100  100  0.547384
19  100  500  0.589623
==================================================
Max auc score==> 0.6471222535106698
**************************************************
```

```
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.647122
5     0.646599
6     0.646669
7     0.647081
8     0.644332
9     0.644747
10    0.645518
11    0.645850
12    0.498545
13    0.511078
14    0.544206
15    0.607583
16    0.490934
17    0.521284
18    0.547384
19    0.589623
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>    0  1         2
4  5  5  0.647122
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4     5    5  0.647122
5     5   10  0.646599
6     5  100  0.646669
7     5  500  0.647081
8    10    5  0.644332
9    10   10  0.644747
10   10  100  0.645518
11   10  500  0.645850
12   50    5  0.498545
13   50   10  0.511078
14   50  100  0.544206
15   50  500  0.607583
16  100    5  0.490934
17  100   10  0.521284
18  100  100  0.547384
19  100  500  0.589623
==================================================
Max auc score==> 0.6471222535106698
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.647122
5     0.646599
6     0.646669
7     0.647081
8     0.644332
9     0.644747
10    0.645518
11    0.645850
12    0.498545
13    0.511078
14    0.544206
15    0.607583
16    0.490934
17    0.521284
18    0.547384
19    0.589623
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>    0  1         2
4  5  5  0.647122
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
```

```
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```

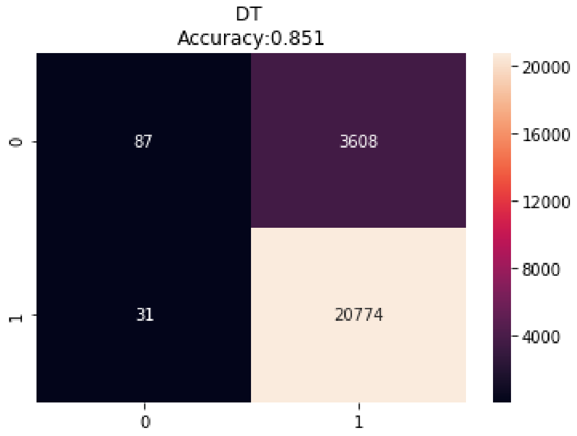


Plot Confusion Matrix

In [77]:

```python
from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for IMP_TFIDF")
cm =confusion_matrix(y_train, pred_imp_tfidf_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT \nAccuracy:{0:.3f}'.format(accuracy_score(y_train, pred_imp_tfidf_train)))
plt.show()

print("="*50)
print("Testing CM for IMP_TFIDF")

cm =confusion_matrix(y_test, pred_imp_tfidf_test, labels=None, sample_weight=None)
summary.append(['Imp_tfidf',find_best_params(roc_auc_score_cv_imp_tfidf_dict)
['depth'],find_best_params(roc_auc_score_cv_imp_tfidf_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")


print("="*50)
print("Testing CM for IMP_TFIDF")
plt.title('DT  \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, pred_imp_tfidf_test)))
plt.show()
```

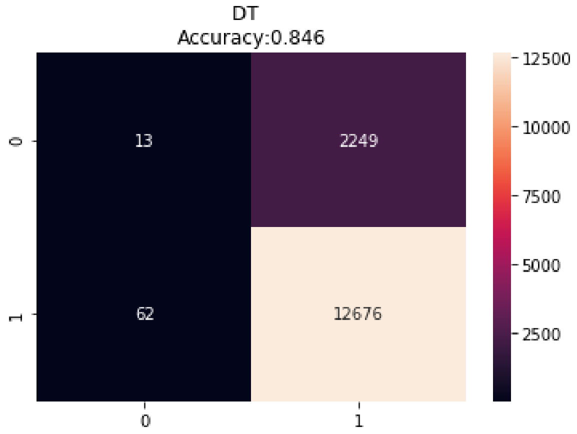
```
Training CM for IMP_TFIDF
```



```
==================================================
Testing CM for IMP_TFIDF
       0    1         2
0      1    5  0.550577
1      1   10  0.550577
```

```
2     1  100  0.550577
3     1  500  0.550577
4     5    5  0.647122
5     5   10  0.646599
6     5  100  0.646669
7     5  500  0.647081
8    10    5  0.644332
9    10   10  0.644747
10   10  100  0.645518
11   10  500  0.645850
12   50    5  0.498545
13   50   10  0.511078
14   50  100  0.544206
15   50  500  0.607583
16  100    5  0.490934
17  100   10  0.521284
18  100  100  0.547384
19  100  500  0.589623
==================================================
Max auc score==> 0.6471222535106698
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.647122
5     0.646599
6     0.646669
7     0.647081
8     0.644332
9     0.644747
10    0.645518
11    0.645850
12    0.498545
13    0.511078
14    0.544206
15    0.607583
16    0.490934
17    0.521284
18    0.547384
19    0.589623
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>    0  1         2
4  5  5  0.647122
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
      0    1         2
0     1    5  0.550577
1     1   10  0.550577
2     1  100  0.550577
3     1  500  0.550577
4     5    5  0.647122
5     5   10  0.646599
6     5  100  0.646669
7     5  500  0.647081
8    10    5  0.644332
9    10   10  0.644747
10   10  100  0.645518
11   10  500  0.645850
12   50    5  0.498545
13   50   10  0.511078
14   50  100  0.544206
15   50  500  0.607583
16  100    5  0.490934
17  100   10  0.521284
18  100  100  0.547384
19  100  500  0.589623
==================================================
Max auc score==> 0.6471222535106698
**************************************************
temp[2]==> 0     0.550577
1     0.550577
2     0.550577
3     0.550577
4     0.647122
```

```
5     0.646599
6     0.646669
7     0.647081
8     0.644332
9     0.644747
10    0.645518
11    0.645850
12    0.498545
13    0.511078
14    0.544206
15    0.607583
16    0.490934
17    0.521284
18    0.547384
19    0.589623
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>    0  1         2
4  5  5  0.647122
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
==================================================
Testing CM for IMP_TFIDF
```



False Positive Data Analysis

False Positives

In [78]:

```
y_test=y_test.values
X_test_price=X_test['price'].values
X_test_essay=X_test['essay'].values
```

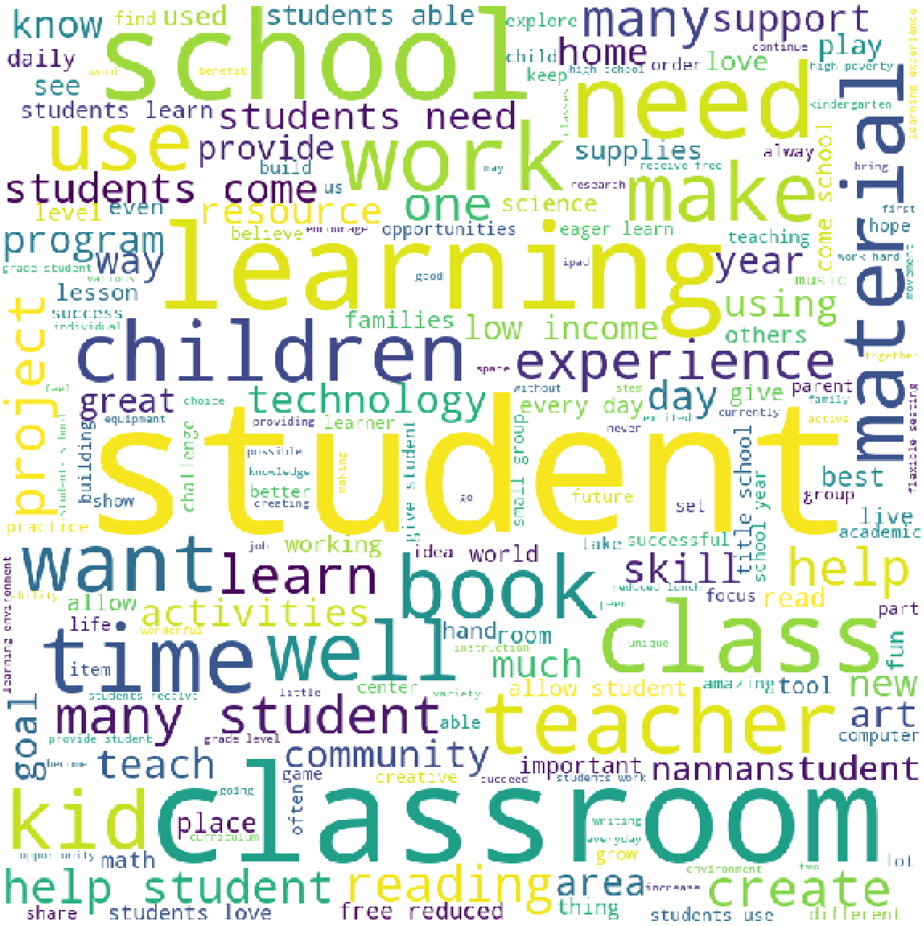
In [79]:

```
fpr=[]
essay=''
price_box=[]
X_test_prev=[]
for i in range(len(y_test)):
    if (int((y_test[i]) == 0) and (int(pred_tfidf_test[i]) == 1)):
        fpr.append(1)
        essay+=X_test_essay[i]
        price_box.append(X_test_price[i])
        X_test_prev.append(X_test_prev_proj[i])
    else :
        fpr.append(0)
```

Draw Word Cloud

In [80]:

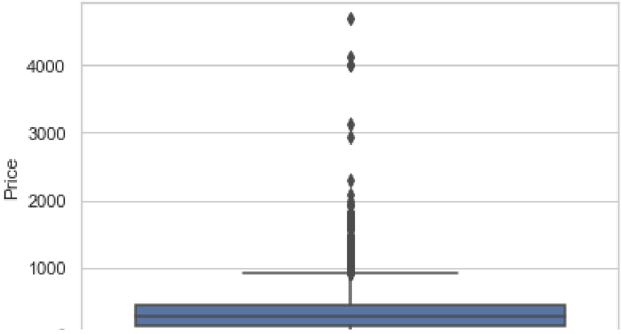
```
from wordcloud import WordCloud
wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(essay)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

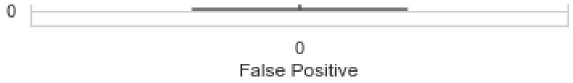


Box Plot for Price

In [81]:

```
sns.set(context='notebook', style='whitegrid')
plt.xlabel("False Positive")
plt.ylabel("Price")
sns.boxplot(data=price_box)
plt.show()
```

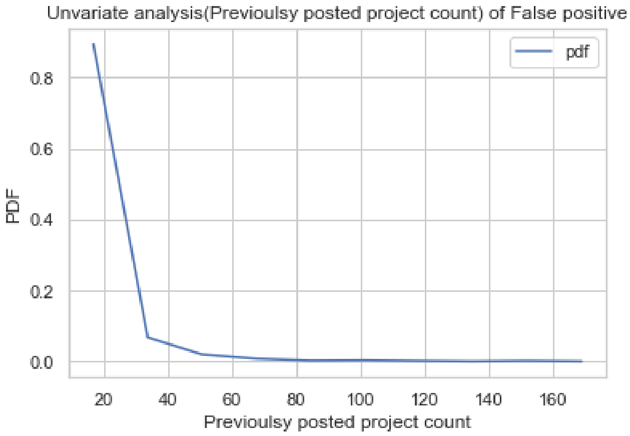




PDF for Previusly Posted

In [82]:

```
count,bin_edges=np.histogram(X_test_prev,bins=10,density=True)

#pdf
pdf=count/(sum(count))

p=plt.plot(bin_edges[1:],pdf,label='pdf')

plt.ylabel('PDF')
plt.xlabel('Previoulsy posted project count')
plt.legend()
plt.title('Unvariate analysis(Previoulsy posted project count) of False positive')
plt.show()
```



Conclusion

In [83]:

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Depth","Sample", "AUC"]

for each in summary:
    x.add_row(each)

print(x)
```

```
+------------+-------+--------+--------------------+
| Vectorizer | Depth | Sample |        AUC         |
+------------+-------+--------+--------------------+
|   Tfidf    |   10  |  500   | 0.6433122542198834 |
| Tfidf_w2v  |   5   |   5    | 0.6472079822982092 |
| Imp_tfidf  |   5   |   5    | 0.6135126883518878 |
+------------+-------+--------+--------------------+
```