DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom p
number of volunteers is needed to manually screen each submission before it's approved to be po

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

- How to scale current manual processes and resources to screen 500,000 projects so that th
as possible
- How to increase the consistency of project vetting across different volunteers to improve the
- 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 subı
the text of project descriptions as well as additional metadata about the project, teacher, and scho
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 • `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 • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Examp**

Feature	Description
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the pro • `Literacy` • `Literature & Writing, Social Science`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy m`
`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-`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example**
`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 t

* 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 ea resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p0365(`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds,`
`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 tra 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 projec

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 fo following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific d neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of projec

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_

Enter your authorization code:
··········
Mounted at /content/drive
```

```
!ls drive/'My Drive'/data/train_data.csv
```

```
'drive/My Drive/data/train_data.csv'
```

```
# Latest version 3.1.1 of matplotlib is breaking the seaborn library hence going
pip install matplotlib==3.1.0
```

```
Collecting matplotlib==3.1.0
  Downloading https://files.pythonhosted.org/packages/da/83/d989ee20c78117c737
     |████████████████████████████████| 13.1MB 2.8MB/s
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /us
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/d
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-pac
ERROR: plotnine 0.6.0 has requirement matplotlib>=3.1.1, but you'll have matpl
ERROR: mizani 0.6.0 has requirement matplotlib>=3.1.1, but you'll have matplot
ERROR: albumentations 0.1.12 has requirement imgaug<0.2.7,>=0.2.5, but you'll
Installing collected packages: matplotlib
  Found existing installation: matplotlib 3.1.2
    Uninstalling matplotlib-3.1.2:
      Successfully uninstalled matplotlib-3.1.2
Successfully installed matplotlib-3.1.0
WARNING: The following packages were previously imported in this runtime:
  [matplotlib,mpl_toolkits]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
%matplotlib inline
import warnings
```

```
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
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


import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string


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

from tqdm import tqdm
import os


from collections import Counter
```

1.1 Reading Data

```
project_data = pd.read_csv('drive/My Drive/data/train_data.csv')
resource_data = pd.read_csv('drive/My Drive/data/resources.csv')
```

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'sch
 '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']
```

```
print('The Columns with their nan values counts are below ')
for col in project_data.columns:
    print('{col} '.format(col=col),project_data[col].isnull().sum())
```

```
The Columns with their nan values counts are below
Unnamed: 0  0
id  0
teacher_id  0
teacher_prefix  3
school_state  0
project_submitted_datetime  0
project_grade_category  0
project_subject_categories  0
project_subject_subcategories  0
project_title  0
project_essay_1  0
project_essay_2  0
project_essay_3  105490
project_essay_4  105490
project_resource_summary  0
teacher_number_of_previously_posted_projects  0
project_is_approved  0
```

```
# removing 3 nan values from teacher prefix column as they seems to be outliers
# DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
project_data.dropna(subset=['teacher_prefix'],inplace=True)
```

```
#how to replace elements in list python: https://stackoverflow.com/a/2582163/408
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_d


#sort dataframe based on time pandas python: https://stackoverflow.com/a/4970249
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/408
project_data = project_data[cols]


project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

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

1.2 preprocessing of `project_subject_categories`

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-
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",
        if 'The' in j.split(): # this will split each of the catogory based on s
            j=j.replace('The','') # if we have the words "The" we are going to r
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trail
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())


# Considering each mix of category as a unique category and to stop repeating we
project_data['clean_categories'] = ["__".join(sorted(x.split())).strip() for x i

project_data.drop(['project_subject_categories'], axis=1, inplace=True) #0 for i

#counting the occurence of word

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]))

```

```
project_data['clean_categories'].value_counts()
```

```
Literacy_Language                          23654
Math_Science                               17072
Literacy_Language__Math_Science            16923
Health_Sports                              10177
Music_Arts                                  5180
SpecialNeeds                                4226
Literacy_Language__SpecialNeeds             3961
AppliedLearning                             3771
AppliedLearning__Literacy_Language          2827
AppliedLearning__Math_Science               2272
History_Civics__Literacy_Language           2230
History_Civics                              1851
Math_Science__SpecialNeeds                  1840
Literacy_Language__Music_Arts               1757
Math_Science__Music_Arts                    1642
AppliedLearning__SpecialNeeds               1467
Health_Sports__SpecialNeeds                 1433
Care_Hunger__Warmth                         1309
History_Civics__Math_Science                 974
Health_Sports__Literacy_Language             875
AppliedLearning__Health_Sports               800
AppliedLearning__Music_Arts                  768
Health_Sports__Math_Science                  685
Music_Arts__SpecialNeeds                     440
History_Civics__Music_Arts                   330
History_Civics__SpecialNeeds                 252
AppliedLearning__History_Civics              220
Health_Sports__Music_Arts                    174
Health_Sports__History_Civics                 56
Care_Hunger__SpecialNeeds__Warmth             23
Care_Hunger__Health_Sports__Warmth            23
Care_Hunger__Math_Science__Warmth             11
AppliedLearning__Care_Hunger__Warmth          10
Care_Hunger__Literacy_Language__Warmth         9
Care_Hunger__Music_Arts__Warmth                2
Care_Hunger__History_Civics__Warmth            1
Name: clean_categories, dtype: int64
```

```
1 project_data['clean_categories'].unique()
```

```
array(['Math_Science', 'SpecialNeeds', 'Literacy_Language',
       'AppliedLearning', 'History_Civics__Math_Science',
       'Literacy_Language__Math_Science', 'AppliedLearning__Music_Arts',
       'AppliedLearning__Math_Science',
       'History_Civics__Literacy_Language',
       'AppliedLearning__Health_Sports', 'Math_Science__Music_Arts',
       'AppliedLearning__Literacy_Language', 'Music_Arts',
       'Health_Sports', 'Literacy_Language__SpecialNeeds',
       'Math_Science__SpecialNeeds', 'AppliedLearning__History_Civics',
       'AppliedLearning__SpecialNeeds',
       'Health_Sports__Literacy_Language',
       'Literacy_Language__Music_Arts', 'Health_Sports__SpecialNeeds',
       'History_Civics__Music_Arts', 'Health_Sports__Math_Science',
       'Music_Arts__SpecialNeeds', 'Health_Sports__History_Civics',
       'History_Civics', 'History_Civics__SpecialNeeds',
       'Health_Sports__Music_Arts', 'Care_Hunger__Health_Sports__Warmth',
       'Care_Hunger__History_Civics__Warmth',
       'Care_Hunger__Math_Science__Warmth',
       'Care_Hunger__SpecialNeeds__Warmth', 'Care_Hunger__Warmth',
       'Care_Hunger__Literacy_Language__Warmth',
       'Care_Hunger__Music_Arts__Warmth',
       'AppliedLearning__Care_Hunger__Warmth'], dtype=object)
```

```
1
2 cat_dict.items()
```

```
dict_items([('Math_Science', 17072), ('SpecialNeeds', 4226), ('Literacy_Langua
```

1.3 preprocessing of `project_subject_subcategories`

```
1 sub_catogories = list(project_data['project_subject_subcategories'].values)
2 # remove special characters from list of strings python: https://stackoverflow.c
3
4 # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
5 # https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from
6 # https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-
7
8 sub_cat_list = []
9 for i in sub_catogories:
10     temp = ""
11     # consider we have text like this "Math & Science, Warmth, Care & Hunger"
12     for j in i.split(','): # it will split it in three parts ["Math & Science",
13         if 'The' in j.split(): # this will split each of the catogory based on s
14             j=j.replace('The','') # if we have the words "The" we are going to r
15         j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty
16         temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trail
17         temp = temp.replace('&','_')
18     sub_cat_list.append(temp.strip())
19
20 # Considering each mix of category as a unique category and to stop repeating we
21 project_data['clean_subcategories'] = ["__".join(sorted(x.split())).strip() for
22 project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
23
```

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/
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]))
```

```
(project_data['clean_subcategories'].value_counts()==1).sum()
```

```
24
```

```
mask = project_data['clean_subcategories'].value_counts()==1
list(project_data['clean_subcategories'].value_counts()[mask].index)
```

```
['ESL__Economics',
 'Care_Hunger__Gym_Fitness__Warmth',
 'CommunityService__Music',
 'CommunityService__Gym_Fitness',
 'ESL__TeamSports',
 'Care_Hunger__College_CareerPrep__Warmth',
 'FinancialLiteracy__PerformingArts',
 'Economics__NutritionEducation',
 'Gym_Fitness__ParentInvolvement',
 'Care_Hunger__Other__Warmth',
 'Gym_Fitness__SocialSciences',
 'Economics__Other',
 'Civics_Government__ForeignLanguages',
 'Economics__Music',
 'Civics_Government__NutritionEducation',
 'Literature_Writing__NutritionEducation',
 'ParentInvolvement__TeamSports',
 'Care_Hunger__ParentInvolvement__Warmth',
 'Extracurricular__FinancialLiteracy',
 'FinancialLiteracy__ForeignLanguages',
 'CommunityService__FinancialLiteracy',
 'Care_Hunger__History_Geography__Warmth',
 'Civics_Government__ParentInvolvement',
 'Economics__ForeignLanguages']
```

```
len(project_data['clean_subcategories'].unique())
```

```
401
```

```
# removing spaces and - from Grades 3-5 into Grades_3-5
project_data['project_grade_category'] = project_data['project_grade_category'].

project_data.head()
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	

1.3 Text preprocessing

```
# 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)
```

```
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

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

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom a
==================================================
I teach high school English to students with learning and behavioral disabilit
==================================================
\"Life moves pretty fast. If you don't stop and look around once in awhile, yo
==================================================
Some of my students come from difficult family lives, but they don't let that
==================================================
\"This is how mathematicians do it! Remember we are all mathematicians in this
==================================================
```

```
# 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
```

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

```
Some of my students come from difficult family lives, but they do not let that
==================================================
```

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

```
sent = sent.replace('\\n', ' ')
print(sent)
```

```
Some of my students come from difficult family lives, but they do not let that
```

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

```
Some of my students come from difficult family lives but they do not let that
```

```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itse
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 't
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than'
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should'v
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "d
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn
            'won', "won't", 'wouldn', "wouldn't"]
```

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

```
100%|██████████| 109245/109245 [00:57<00:00, 1893.23it/s]
```

```
# after preprocesing
preprocessed_essays[20000]
```

```
'students come difficult family lives not let stop built community classroom a
```

1.4 Preprocessing of `project_title`

```
# similarly you can preprocess the titles also
preprocessed_title = []

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

```
100%|██████████| 109245/109245 [00:02<00:00, 41257.89it/s]
```

1.5 Preprocessing of `Teacher Prefix`

```
# let's check the distribution of this prefix with having period and  special ch
# https://www.geeksforgeeks.org/python-program-check-string-contains-special-cha
import re
regex = re.compile('[@_!#$%^&*()<>?/\|}{~:.]')
project_data.teacher_prefix.map(lambda x: regex.search(x)== None).value_counts()
```

```
False    106885
True       2360
Name: teacher_prefix, dtype: int64
```

```
# https://stackoverflow.com/questions/50444346/fast-punctuation-removal-with-par
# cleaning the teacher prefix columns as the cells have periods associated with
# python's str.translate function is implemented in C, and is therefore very fas

def clean_col(col):
    import string
    punct = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{}~'   # `|` is not present in teache
    transtab = str.maketrans(dict.fromkeys(punct, ''))

    col = '|'.join(col.tolist()).translate(transtab).split('|')
    return col
```

```
preprocessed_prefix = clean_col(project_data['teacher_prefix'])
```

```
# verifying if any special char are present or not
c = list(map(lambda x : regex.search(x) == None,preprocessed_prefix)).count(True
print(c)
```

```
109245
```

1.5 Preparing data for models

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

Assignment 7: SVM

1. **[Task-1] Apply Support Vector Machines(SGDClassifier with hinge loss: Linear SVM) on these feature se**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BO
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TF
   - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay
   - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassa

2. **The hyper paramter tuning (best alpha in range [10^-4 to 10^4], and the best penalty among 'l1', 'l2')**
   - Find the best hyper parameter which will give the maximum AUC value
   - Find the best hyper paramter using k-fold cross validation or simple cross validation da
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do th

3. **Representation of results**
    - You need to plot the performance of model both on train data and cross validation data figure. 
    - Once after you found the best hyper parameter, you need to train your model with it, and curve on both train and test. 
    - Along with plotting ROC curve, you need to print the confusion matrix with predicted and visualize your confusion matrices using seaborn heatmaps. 

4. **[Task-2] Apply the Support Vector Machines on these features by finding the best hyper paramter as suc**
    - Consider these set of features Set 5 :
        - **school_state** : categorical data
        - **clean_categories** : categorical data
        - **clean_subcategories** : categorical data
        - **project_grade_category** :categorical data
        - **teacher_prefix** : categorical data
        - **quantity** : numerical data
        - **teacher_number_of_previously_posted_projects** : numerical data
        - **price** : numerical data
        - **sentiment score's of each of the essay** : numerical data
        - **number of words in the title** : numerical data
        - **number of words in the combine essays** : numerical data
        - **Apply TruncatedSVD on TfidfVectorizer of essay text, choose the number of com method** : numerical data

    - **Conclusion**
        - You need to summarize the results at the end of the notebook, summarize it in th refer to this prettytable library link 

Note: Data Leakage

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

2. Support Vector Machines

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

Pandas Dataframe Reordering

Reordering the pandas dataframe with pre processed essays,title and relevant columns for classifi

```
1 project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

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

```
1 project_data.head()
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_stat
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	C
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	U
2	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	C
3	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	G
4	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	W

```
# https://stackoverflow.com/questions/45747589/copying-a-column-from-one-datafra

project_data_ = pd.DataFrame({c: project_data[c].to_numpy() for c in ('school_st
                                                                      'teacher_nu
project_data_.head(3)


```

	school_state	project_grade_category	clean_categories	clean_subc
0	CA	Grades_PreK_2	Math_Science	AppliedSciences__Health_
1	UT	Grades_3_5	SpecialNeeds	S
2	CA	Grades_PreK_2	Literacy_Language	

```
project_data_ = project_data_.assign(teacher_prefix=preprocessed_prefix,essay =
project_data_.head(3)
```

	school_state	project_grade_category	clean_categories	clean_subc
0	CA	Grades_PreK_2	Math_Science	AppliedSciences__Health_
1	UT	Grades_3_5	SpecialNeeds	S
2	CA	Grades_PreK_2	Literacy_Language	

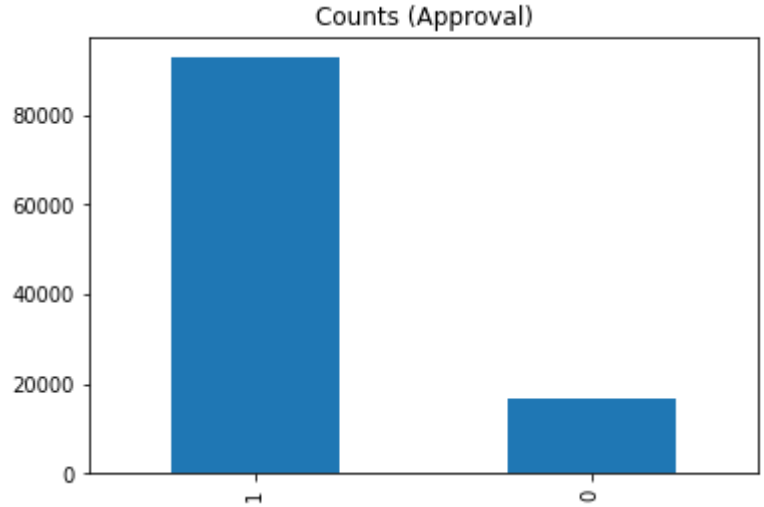
Creating the dataframe for new features first.

```
# Before splitting model let's check if dataset is balanced or not.
print("Negative reviews count = ",np.sum(project_data_.approved==0))
print("positive reviews count = ",np.sum(project_data_.approved==1))
```

```
Negative reviews count =  16542
positive reviews count =  92703
```

```
project_data_.approved.value_counts().plot(kind='bar',title='Counts (Approval)')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93b8328cc0>
```



As we can clearly see that this dataset higly imbalanced towards positive reviews that means mos
on this platform. But to solve this problem we need to oversample the negative reviews.

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.trai
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = None,None,None,None # clearing the variables
# splitting of train and test data with 80:20 ratio

x_train,x_test,y_train,y_test = train_test_split(project_data_.iloc[:,:project_c
#X_train,X_cv,y_train,y_cv = train_test_split(X_traincv,y_traincv,test_size=.2,s
x_train,x_cv,y_train,y_cv = train_test_split(x_train,y_train,random_state=0,test


print("Train Data shape : ",x_train.shape, y_train.shape)
print("Cross Validation Data shape :", x_cv.shape, y_cv.shape)
print("Test Data shape :", x_test.shape, y_test.shape)

print("="*100)

```

```
Train Data shape :  (69916, 10) (69916,)
Cross Validation Data shape : (17480, 10) (17480,)
Test Data shape : (21849, 10) (21849,)
====================================================================================
```

```
# With stratify = class label

print("x train distribution = \n", y_train.value_counts())
print("x cv distribution = \n", y_cv.value_counts())
print("x test distribution = \n", y_test.value_counts())
```

```
x train distribution =
 1    59329
0    10587
Name: approved, dtype: int64
x cv distribution =
 1    14833
0     2647
Name: approved, dtype: int64
x test distribution =
 1    18541
0     3308
Name: approved, dtype: int64
```

```
x_train.head(3)
```

	school_state	project_grade_category	clean_categories	clean_subcateg
32554	IL	Grades_6_8	Literacy_Language	Literacy__Literature_
21860	TX	Grades_9_12	AppliedLearning	College_CareerPrep_
37198	AR	Grades_PreK_2	Literacy_Language	Literacy__Literature_

Functions Declaration:

Declaration of functions for which is further used in computational process like

- Vectorization
- Hyperparamater Tuning
- Model Generalisation score on Test Data
- Printing Dimenionality info of input matrix list
- Retrive the vocabulary words for vectorization purposes

```
def retreive_vocab(_data=None):
    ls = []
    for word in _data:
        if len(word) != 1:
            for w in word.split():
                ls.append(w)
        else:
            ls.append(word)
    return list(set(ls))
```

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-t
# make sure you have the glove_vectors file
with open('drive/My Drive/data/glove_vectors.dms', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

```
def compute_avg_w2v(_text):
```

```
    global model,glove_words
    avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in th
    for sentence in tqdm(_text): # 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.append(vector)

    print(len(avg_w2v_vectors))
    print(len(avg_w2v_vectors[0]))

    return avg_w2v_vectors
```

```
def compute_tfidf_w2v(tfidf_model,_text):
    global model,glove_words
    # we are converting a dictionary with word as a key, and the idf as a value
    idf_value = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)
    tfidf_words = set(tfidf_model.get_feature_names())
    # average Word2Vec
    # compute average word2vec for each review.
    tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in
    for sentence in tqdm(_text): # 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/rev
        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 = idf_value[word]*(sentence.count(word)/len(sentence.spli
                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.append(vector)

    print("The number of rows are: ",len(tfidf_w2v_vectors))
    print("The number of columns are: ",len(tfidf_w2v_vectors[0]))
    return tfidf_w2v_vectors
```

```
# Defining a function to compute BOW, TFIDF and Word2Vec

def vectorize_text(encoding_type=None,**kwargs):
    # Validation for proper argument names
    try:
        checklist = ['test_text','cv_text','train_text']
        for k,v in kwargs.items():
            if k in checklist:
                checklist.remove(k)
```

```
        if not checklist:
            # do nothing
            pass
        else:
            raise ValueError("You haven't passed the matrices in the described f
    except ValueError as e:
        print("Error : ", e)

    text_train = kwargs['train_text']
    text_cv = kwargs['cv_text']
    text_test = kwargs['test_text']

    if "BOW" in encoding_type.upper():
        #Compute BOW
        # We are considering only the words which appeared in at least 10 docume
        vectorizer = CountVectorizer(ngram_range=(2,2),min_df=10,max_features=50
        vectorizer.fit(text_train)
        return vectorizer.transform(text_train),vectorizer.transform(text_cv),ve

    elif "TFIDF" in encoding_type.upper() and "W2V" not in encoding_type.upper()
        #Compute TFIDF
        from sklearn.feature_extraction.text import TfidfVectorizer
        vectorizer = TfidfVectorizer(ngram_range=(2,2),min_df=10,max_features=50
        vectorizer.fit(text_train)
        return vectorizer.transform(text_train),vectorizer.transform(text_cv),ve

    elif "AVGW2V" in encoding_type.upper():
        # compute average word2vec for each review.
        train = np.array(compute_avg_w2v(_text = text_train))
        cv = np.array(compute_avg_w2v(_text = text_cv))
        test = np.array(compute_avg_w2v(_text = text_test))
        return train,cv,test

    elif "TFIDFW2V" in encoding_type.upper():
        from sklearn.feature_extraction.text import TfidfVectorizer
        vectorizer = TfidfVectorizer(ngram_range=(2,2),min_df=10,max_features=50
        vectorizer.fit(text_train)
        train = np.array(compute_tfidf_w2v(tfidf_model=vectorizer,_text = text_t
        cv = np.array(compute_tfidf_w2v(tfidf_model = vectorizer,_text = text_cv
        test = np.array(compute_tfidf_w2v(tfidf_model = vectorizer,_text = text_
        return train,cv,test

    else:
        raise ValueError('Please give the encoding type from the following: BOW,


```

```
def _hypertuning(x_train,y_train,x_cv,y_cv,tune_type,penalty_type='l2',max_epoch
  hyper_param = np.arange(10**-4,10**4,100)
  #[x for x in xrange(10**-1,10**2,10)]
  from sklearn.linear_model import SGDClassifier
# SGDClassifier(loss='hinge',penalty=penalty_type,alpha=0.00001,max_iter=10000,l
  if tune_type.lower() == 'custom':

    from sklearn.metrics import roc_auc_score
```

```
    batch_size = 100
    #y_train_pred = []
    train_auc_score = []
    cv_auc_score = []

    for i in hyper_param:
        svc = SGDClassifier(loss='hinge',penalty=penalty_type,alpha=i,max_iter=m
        svc.fit(x_train,y_train)
        #y_train_pred_k = list()
        #y_cv_pred_k = list()
        y_train_pred = []
        y_cv_pred = []

        for k in range(0,y_train.shape[0]-batch_size,batch_size):
            y_train_pred.extend(svc.decision_function(x_train[k:k+batch_size,:])
        for k in range(0,y_cv.shape[0]-batch_size,batch_size):
            y_cv_pred.extend(svc.decision_function(x_cv[k:k+100,:]))

    # return value of decision_function : array of shape = [n_samples, n_classes
        #y_train_pred.extend(svc.decision_function(x_train))
        #y_cv_pred.extend(svc.decision_function(x_cv))
        train_auc_score.append(roc_auc_score(y_train[0:len(y_train_pred)],y_trai
        cv_auc_score.append(roc_auc_score(y_cv[0:len(y_cv_pred)],y_cv_pred))

    plt.plot(np.log10(hyper_param),train_auc_score, label='Train AUC')
    #plt.scatter(K,train_auc_score)
    plt.plot(np.log10(hyper_param), cv_auc_score, label='CV AUC')
    #plt.scatter(K,cv_auc_score)
    plt.legend()
    plt.xlabel("C (1/lambda): hyperparameter")
    plt.ylabel("AUC")
    plt.title("ERROR PLOTS "+penalty_type)
    plt.show()


  elif tune_type.lower() == 'gridsearch':

    # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.
    from sklearn.model_selection import GridSearchCV

    svc = SGDClassifier(loss='hinge',penalty=penalty_type,max_iter=max_epoch,lea
    parameters = {'alpha':hyper_param}
    clf = GridSearchCV(svc, parameters, cv=10, scoring='roc_auc',n_jobs=-1,retur
    clf.fit(x_train, y_train)

    train_auc = clf.cv_results_['mean_train_score']
    train_auc_std = clf.cv_results_['std_train_score']
    cv_auc = clf.cv_results_['mean_test_score']
    cv_auc_std= clf.cv_results_['std_test_score']

    plt.plot(np.log10(parameters['alpha']), train_auc, label='Train AUC')
      # this code is copied from here: https://stackoverflow.com/a/48803361/4084
    plt.gca().fill_between(np.log10(parameters['alpha']),train_auc - train_auc_s

    plt.plot(np.log10(parameters['alpha']), cv_auc, label='CV AUC')
```

```
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084
    plt.gca().fill_between(np.log10(parameters['alpha']),cv_auc - cv_auc_std,cv_
    plt.legend()
    plt.xlabel("C (1 / lambda): hyperparameter")
    plt.ylabel("AUC")
    plt.title("ERROR PLOTS "+penalty_type)
    plt.show()
```

```
#https://www.ritchieng.com/machine-learning-evaluate-classification-model/

def evaluate_threshold(clf,x_cv,y_cv,first=0,last=0):
    y_cv_pred = []
    y_cv_pred.extend(clf.decision_function(x_cv))
    from sklearn.metrics import roc_curve
    #import pdb
    #pdb.set_trace()
    fpr,tpr,thresholds = roc_curve(y_true = y_cv,y_score = y_cv_pred)
    if first==0 and last==0:
        initial = np.array(y_cv_pred).min()
        final = np.array(y_cv_pred).max()
    else:
        initial = first
        final = last
    #t_val = [0.1,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.8,0.9] # di
    t_val = np.linspace(initial,final,num=20)
    sn,sp =0,0

    ss_score = list()
    for i in t_val:
        sn = tpr[thresholds > i][-1]
        sp = 1 - fpr[thresholds > i][-1]
        ss_score.append((sn,sp,i))
    from prettytable import PrettyTable
    x = PrettyTable()
    x.field_names = ['Sensitivity/Recall (TPR)','Specificity (1-FPR)','Threshold
    for val in ss_score:
        sn,sp,th = val
        x.add_row([sn,sp,th])
    print(x)
    plt.figure(figsize=(10,5))
    plt.plot(thresholds,tpr,label='Sensitivity/Recall (TPR)')
    plt.plot(thresholds,1-fpr,label='Specificity (1-FPR)')
    plt.legend()
    plt.xlabel('Thresholds')
    plt.ylabel('Performance rates')
    plt.title('Sensitivity/Specificity vs Thresholds')

    plt.show()

```

```
# https://stackoverflow.com/questions/19984957/scikit-predict-default-threshold

def model_gen_score(svc,x_train,y_train,x_test,y_test,best_alpha,cutoff_val):
    from sklearn.metrics import roc_curve, auc
    from sklearn.metrics import confusion_matrix
    #from sklearn.linear_model import SGDClassifier

    #svc = SGDClassifier(loss='hinge',penalty=penalty_type,alpha=best_alpha,max_
    #svc.fit(x_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
# not the predicted outputs
    y_train_pred_prob = []
    #y_train_pred = []
    y_test_pred_prob = []
    #y_test_pred = []

    y_train_pred_prob.extend(svc.decision_function(x_train))
    y_test_pred_prob.extend(svc.decision_function(x_test))

    train_fpr, train_tpr, train_thresholds = roc_curve(y_true=y_train,y_score=y_
    test_fpr, test_tpr, test_thresholds = roc_curve(y_true=y_test,y_score=y_test

    plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_
    #plt.scatter(train_fpr, np.exp(train_thresholds))
    plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr))
    #plt.scatter(test_fpr, np.exp(test_thresholds))
    plt.legend()
    plt.xlabel("FPR (1 - Specificity)")
    plt.ylabel("TPR (Sensitivity)")
    plt.title("ROC Curve")
    plt.show()


    # Confusion matrix evaluations

    print("="*100)
    y_train_pred = (np.array(y_train_pred_prob) >= cutoff_val).astype(int)
    y_test_pred = (np.array(y_test_pred_prob) >= cutoff_val).astype(int)
    f, (ax1, ax2) = plt.subplots(2, 1,figsize=[8,8])
    print("Confusion matrix \n")
    akws = {"ha": 'left',"va": 'top'}
    sns.heatmap(data=confusion_matrix(y_train, y_train_pred),annot=True,annot_kw
    ax1.set_title('Train confusion matrix')
    sns.heatmap(data=confusion_matrix(y_test, y_test_pred),annot=True,fmt="",ax=
    ax2.set_title('Test confusion matrix')
    plt.show()
    return (auc(train_fpr, train_tpr),auc(test_fpr, test_tpr))


```

```
def print_dimension_info(_obj,_name):
  data_list= ['Training count : ','Cross Validation count : ','Test count : '] *
  col_num = []
```

```
  row_num = list()
  for i in _obj:
    row_num.append(i.shape[0])
    col_num.append(i.shape[1])
  print("The Values for : ", _name)
  print("\nRow Values are : ",list(zip(data_list,row_num)))
  print("\nColumn Values are : ",list(zip(data_list,col_num)))
  print("\nType of matrices: ",[type(x) for x in _obj])
  print("*"*100)
```

```

def one_hot_encoder(df_col_train,df_col_cv,df_col_test,vocab=None,case=False,_bi
    encoder_obj = CountVectorizer(vocabulary = vocab,lowercase=case,binary=_bin)
    encoder_obj.fit(df_col_train)
    print("features are : \n",encoder_obj.get_feature_names())

    return encoder_obj.transform(df_col_train.values),encoder_obj.transform(df_c

```

2.2 Make Data Model Ready: encoding numerical, categorical feature

```
# please write all the code with proper documentation, and proper titles for eac
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debuggi
# make sure you featurize train and test data separatly

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

```
# Vectorizing the teacher prefix input
x_train_tp,x_cv_tp,x_test_tp,tp_feat_names = one_hot_encoder(df_col_train=x_trai
                                                df_col_cv=x_cv['teacher_prefix'],
                                                df_col_test=x_test['teacher_prefi

```

```
features are :
 ['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
```

```
# vectorizing the school state column
x_train_ss,x_cv_ss,x_test_ss,ss_feat_names = one_hot_encoder(df_col_train=x_trai
                                                df_col_test=x_test['school_state'

```

```
features are :
 ['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA'
```

```
# Vectorizing the project_grade_category
```

```
# vectorizing the project_grade_category
x_train_pgc,x_cv_pgc,x_test_pgc,pgc_feat_names = one_hot_encoder(df_col_train=x_
                                                 df_col_test=x_test['project_gr
```

```
features are :
 ['Grades_6_8', 'Grades_9_12', 'Grades_PreK_2', 'Grades_3_5']
```

```
# Vectorizing the project subject category
x_train_cat,x_cv_cat,x_test_cat,cat_feat_names = one_hot_encoder(df_col_train=x_
                                                 df_col_test=x_test['clean_cate
```

```
features are :
 ['Care_Hunger__Math_Science__Warmth', 'Health_Sports', 'History_Civics__Liter
```

```
# Vectorizing the project subject sub category
x_train_sub,x_cv_sub,x_test_sub,sub_feat_names = one_hot_encoder(df_col_train=x_
                                                 df_col_test=x_test['clean_subc
```

```
features are :
 ['ESL__ForeignLanguages', 'AppliedSciences__EarlyDevelopment', 'College_Caree
```

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

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

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

# Now standardize the data with above maen and variance.
x_train_price_std = price_scalar.transform(x_train['price'].values.reshape(-1, 1
x_cv_price_std = price_scalar.transform(x_cv['price'].values.reshape(-1, 1))
x_test_price_std = price_scalar.transform(x_test['price'].values.reshape(-1, 1))
```

```
Mean : 297.4479599805481, Standard deviation : 363.1683554071137
```

```
# we will be doing the standardization of teacher_number_of_previously_posted_pr
teacher_pp_count = StandardScaler()
teacher_pp_count.fit(x_train.teacher_number_of_previously_posted_projects.values
print(f"Mean : {teacher_pp_count.mean_[0]}, Standard deviation : {np.sqrt(teache

x_train_pp_count_std = teacher_pp_count.transform(x_train.teacher_number_of_prev
x_cv_pp_count_std = teacher_pp_count.transform(x_cv.teacher_number_of_previously
x_test_pp_count_std = teacher_pp_count.transform(x_test.teacher_number_of_previo

```

```
10
```

```
Mean : 11.080081812460667, Standard deviation : 27.730880870367283
```

```
# we will be doing the standardization of teacher_number_of_previously_posted_pr
quantity_count = StandardScaler()
quantity_count.fit(x_train.quantity.values.reshape(-1,1))
print(f"Mean : {quantity_count.mean_[0]}, Standard deviation : {np.sqrt(quantity

x_train_q_count_std = quantity_count.transform(x_train.quantity.values.reshape(-
x_cv_q_count_std = quantity_count.transform(x_cv.quantity.values.reshape(-1,1))
x_test_q_count_std = quantity_count.transform(x_test.quantity.values.reshape(-1,


```

```
Mean : 17.038017049030266, Standard deviation : 26.18982439542945
```

```
print_dimension_info(_obj=[x_train_tp,x_cv_tp,x_test_tp],_name='Teacher Prefix')
print_dimension_info(_obj=[x_train_ss,x_cv_ss,x_test_ss],_name='Schol State Colu
print_dimension_info(_obj=[x_train_pgc,x_cv_pgc,x_test_pgc],_name = 'Project Gra
print_dimension_info(_obj=[x_train_cat,x_cv_cat,x_test_cat],_name= 'Project subj
print_dimension_info(_obj=[x_train_sub,x_cv_sub,x_test_sub],_name= 'Project subj
print_dimension_info(_obj=[x_train_price_std,x_cv_price_std,x_test_price_std],_n
print_dimension_info(_obj=[x_train_pp_count_std,x_cv_pp_count_std,x_test_pp_coun
print_dimension_info(_obj=[x_train_q_count_std,x_cv_q_count_std,x_test_q_count_s
```

```
The Values for :  Teacher Prefix

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 5), ('Cross Validation count : ',

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
The Values for :  Schol State Column

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 51), ('Cross Validation count : ',

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
The Values for :  Project Grade Category

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 4), ('Cross Validation count : ',

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
The Values for :  Project subject category

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 36), ('Cross Validation count : ',

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
The Values for :  Project subject sub category

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 391), ('Cross Validation count : '

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
The Values for :  Price

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 1), ('Cross Validation count : ',

Type of matrices:  [<class 'numpy.ndarray'>, <class 'numpy.ndarray'>, <class '
**************************************************************************
The Values for :  Count of previous project submitted by teacher

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 1), ('Cross Validation count : ',

Type of matrices:  [<class 'numpy.ndarray'>, <class 'numpy.ndarray'>, <class '
**************************************************************************
The Values for :  Quantity Count of resources

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 1), ('Cross Validation count : ',
```

```
Type of matrices:  [<class 'numpy.ndarray'>, <class 'numpy.ndarray'>, <class '
**************************************************************************
```

```
# 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
x_train_vec = hstack((x_train_tp, x_train_ss, x_train_pgc, x_train_cat, x_train_
x_cv_vec = hstack((x_cv_tp, x_cv_ss, x_cv_pgc, x_cv_cat, x_cv_sub, x_cv_price_st
x_test_vec = hstack((x_test_tp, x_test_ss, x_test_pgc, x_test_cat, x_test_sub, x

print_dimension_info(_obj=[x_train_vec,x_cv_vec,x_test_vec],_name='Stacked spars

# stacking all their features also so that we can interpret the features importa
stacked_feature_list = [tp_feat_names,ss_feat_names,pgc_feat_names,cat_feat_name
feature_list = list()
for features in stacked_feature_list:
    feature_list.extend(features)
feature_list.append('price')
feature_list.append('pp_count')
feature_list.append('quantity')
print("The length of feature list is: ", len(feature_list))
#print("All general features are: \n",feature_list)

# hstack only works with matrices. so can't stack text and matrices
```

```
The Values for :  Stacked sparse matrices dimensions

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 490), ('Cross Validation count : '

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
The length of feature list is:  490
```

2.3 Make Data Model Ready: encoding eassay, and project_title

```
# please write all the code with proper documentation, and proper titles for eac
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debuggi
# make sure you featurize train and test data separatly

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

```
x_train.columns
```

```
Index(['school_state', 'project_grade_category', 'clean_categories',
       'clean_subcategories', 'teacher_number_of_previously_posted_projects',
       'price', 'quantity', 'teacher_prefix', 'essay', 'title'],
```

```
# Bag of words vectorization for train,cv and test data
bow_title_train,bow_title_cv,bow_title_test,bow_title_features = vectorize_text(
                                                                cv_text =x_cv['titl

print_dimension_info(_obj=[bow_title_train,bow_title_cv,bow_title_test],_name='D

bow_essay_train,bow_essay_cv,bow_essay_test,bow_essay_features = vectorize_text(
                                                                cv_text =x_cv['essa
print_dimension_info(_obj=[bow_essay_train,bow_essay_cv,bow_essay_test],_name='D

```

```
The Values for :  Dimensions after BOW on Title :

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 1888), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
The Values for :  Dimensions after BOW on essay

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 5000), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
```

```
# Stacking all the BOW models with existing data frame -

final_x_train = hstack((x_train_vec,bow_title_train,bow_essay_train),format='csr
final_x_cv = hstack((x_cv_vec,bow_title_cv,bow_essay_cv),format='csr')
final_x_test = hstack((x_test_vec,bow_title_test,bow_essay_test),format='csr')

# stacking for features for bow model
bow_feature_list = list()
bow_feature_list.extend(feature_list)
for features in [bow_title_features,bow_essay_features]:
    bow_feature_list.extend(features)
bow_feature_list = np.array(bow_feature_list)

print_dimension_info(_obj=[final_x_train,final_x_cv,final_x_test],_name='The Fir
```

```
The Values for :  The Final Matrix dimension info after BOW

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 7378), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**************************************************************************
```

2.4 Appling SVM on different kind of featurization as mentioned in th

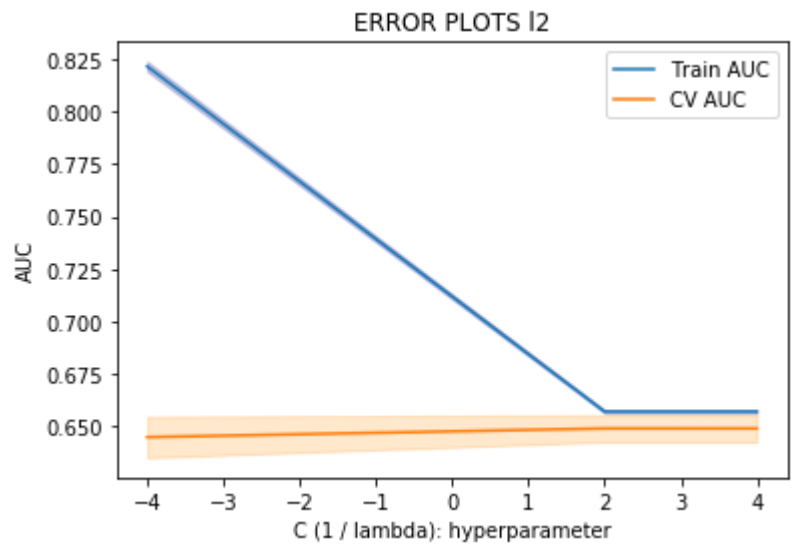
Apply SVM on different kind of featurization as mentioned in the instructions

For Every model that you work on make sure you do the step 2 and step 3 of instructions

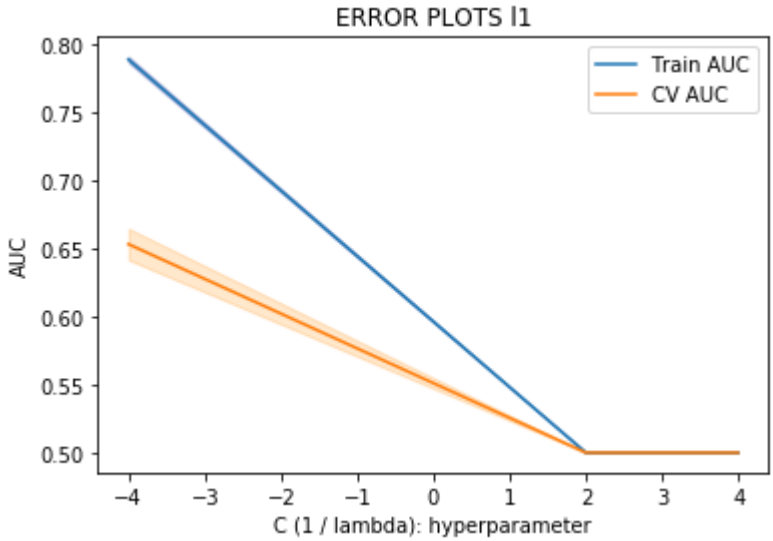
```
# please write all the code with proper documentation, and proper titles for eac
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debuggi

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

2.4.1 Applying SVM brute force on BOW, SET 1

```
# _hypertuning(x_train,y_train,x_cv,y_cv,tune_type,penalty_type='l2',max_epoch =
# With L2 Regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv


```




```
# with L1 regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv


```

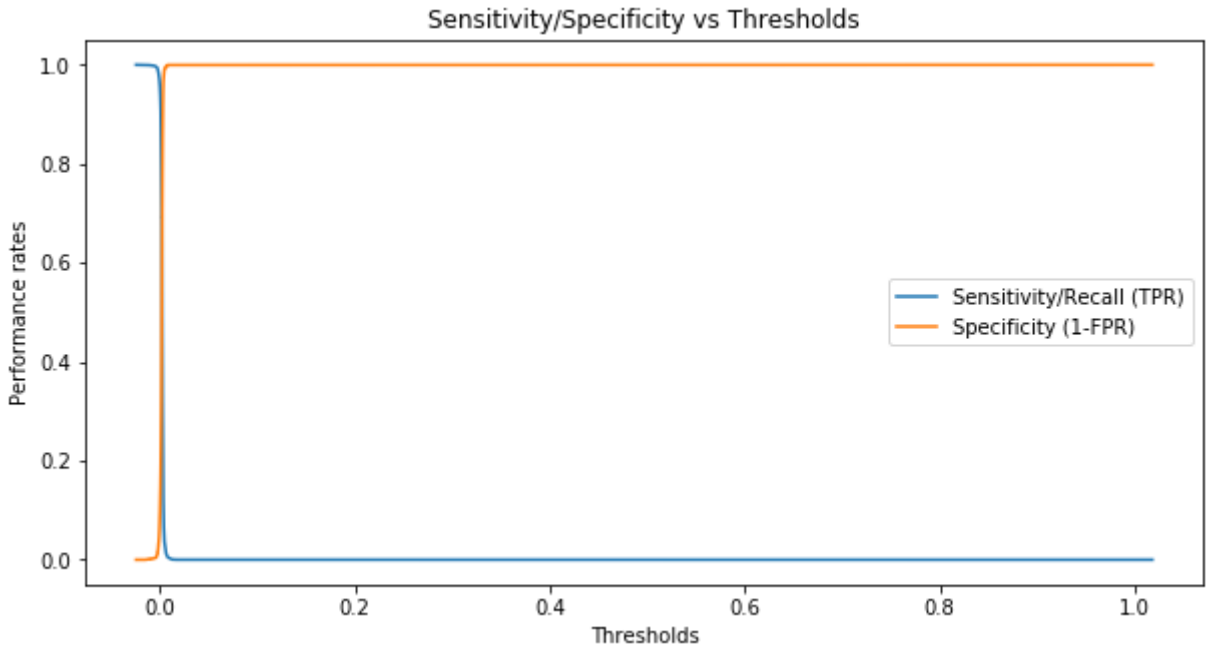



```
1
2 # fitting the SVM with the best c
3
4 from sklearn.linear_model import SGDClassifier
5 bow_c = 10**2
6 svc_clf = SGDClassifier(loss='hinge',penalty='l2',alpha=bow_c,max_iter=10000,lea
7 svc_clf.fit(final_x_train,y_train)
```

```
SGDClassifier(alpha=100, average=False, class_weight='balanced',
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=10000, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=None, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

```
1 # Deciding appropriate threshold with cross validation dataset
2 # evaluate_threshold(_c,x_train,y_train,x_cv,y_cv)
3 evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values)
```

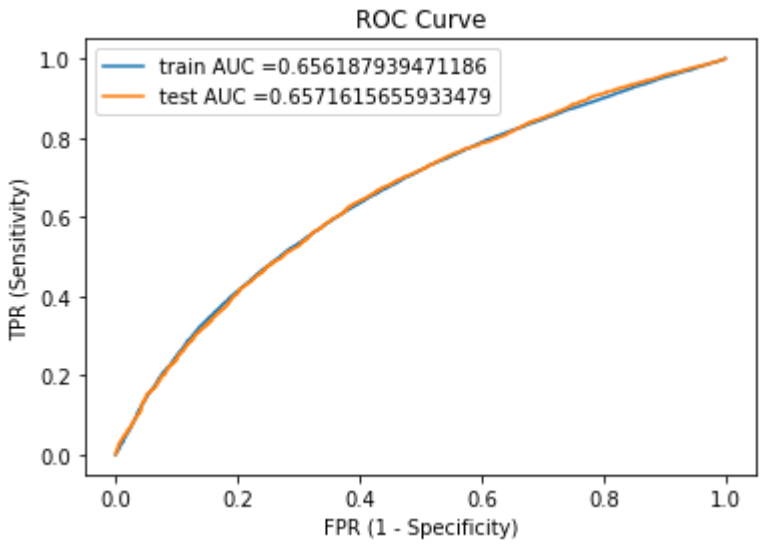
Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.9995280792826805	0.0	-0.02406338265393198
0.9995280792826805	0.0	-0.021867333025638785
0.9995280792826805	0.0	-0.01967128339734559
0.9995280792826805	0.0	-0.017475233769052398
0.9995280792826805	0.0	-0.015279184140759206
0.9994606620373492	0.0007555572346052169	-0.013083134512466012
0.9993258275466864	0.0007555572346052169	-0.01088708488417282
0.9986516550933728	0.0018889308651303116	-0.008691035255879626
0.9981797343760533	0.0026445032111824807	-0.006494985627586432 5
0.9968988067147576	0.0037778617302606232	-0.004298935999293239
0.9917076788242433	0.013978088401964461	-0.002102886371000045
0.9639317737477247	0.07820173781639594	9.316325729314887e-05
0.6837457021506101	0.5100113335851908	0.002289212885586339
0.054945054945054944	0.9814884775217227	0.004485262513879533
0.012674442122294884	0.9958443520967133	0.006681312142172727
0.0005393379626508461	0.9996222138269739	0.00887736177046592
0.0005393379626508461	0.9996222138269739	0.011073411398759114
6.741724533135576e-05	1.0	0.013269461027052308
6.741724533135576e-05	1.0	0.015465510655345502
0.0	1.0	0.017661560283638692



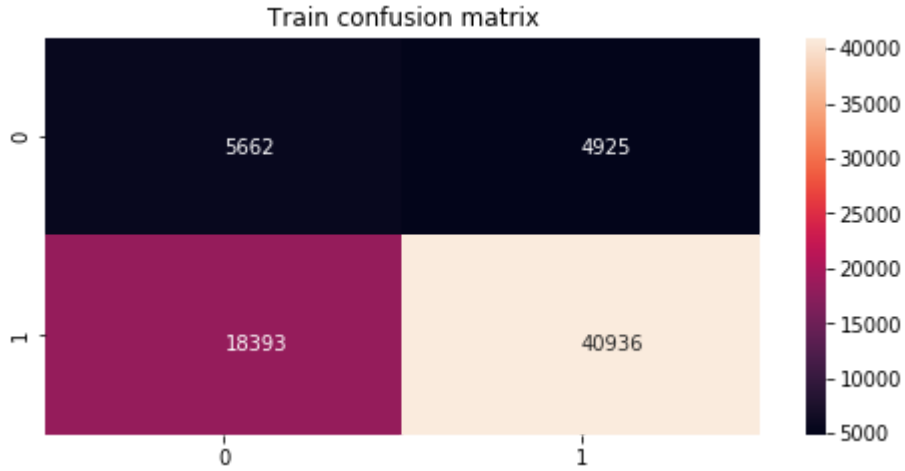
Now by looking at the table above we can go for the value which maximizes both sensitivity and sp towards false neagtive as we don't want our model wrongly classify the neagtive class as it will im choose platform as well as it can hurt the sentiments of projects submitter. So Less False neagtiv means high Recall.

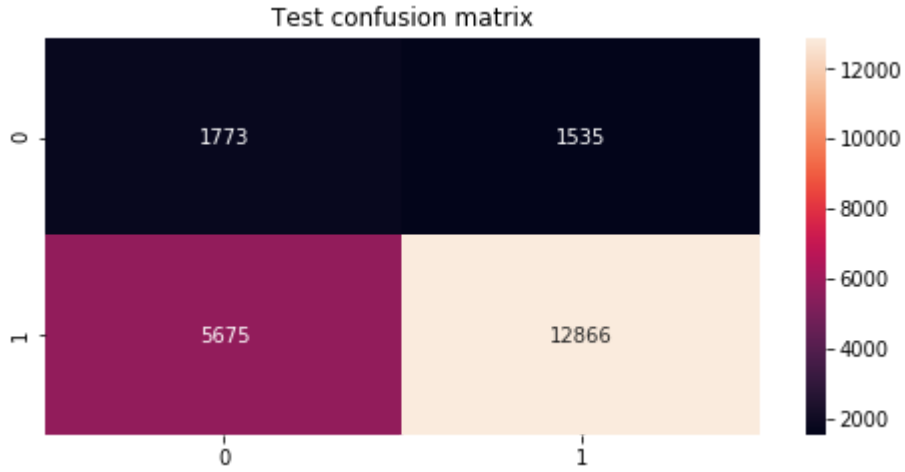
Why at every run of algorithms, I am getting different threshold, Is it due to stochastic nature of Alg

```
# model_gen_score(x_train,y_train,x_test,y_test,best_c,cutoff_val,clf,features_r

bow_train_auc,bow_test_auc = model_gen_score(svc=svc_clf,x_train = final_x_train
                                             x_test=final_x_test,y_test=y_test.v

```




```
==========================================================================================
Confusion matrix
```





2.4.2 Applying SVM brute force on TFIDF, SET 2

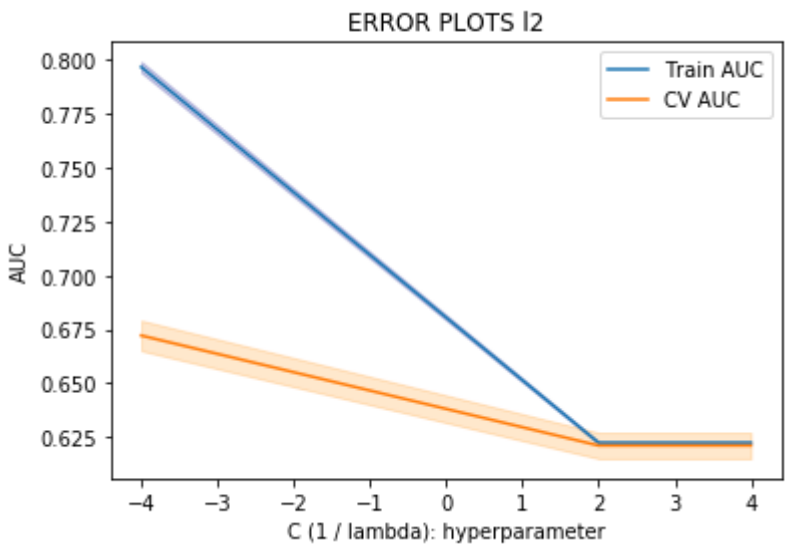
```
# Please write all the code with proper documentation

tfidf_essay_train,tfidf_essay_cv,tfidf_essay_test,tfidf_features_essay = vectori
                                                                  cv_text =x_cv['essa
```

```
# Please write all the code with proper documentation

tfidf_title_train,tfidf_title_cv,tfidf_title_test,tfidf_features_title = vectori
                                                                  cv_text =x_cv['titl
```

```
final_x_train = hstack((x_train_vec,tfidf_title_train,tfidf_essay_train),format=
final_x_cv = hstack((x_cv_vec,tfidf_title_cv,tfidf_essay_cv),format='csr')
final_x_test = hstack((x_test_vec,tfidf_title_test,tfidf_essay_test),format='csr

# stacking for features for bow model
tfidf_feature_list = list()
tfidf_feature_list.extend(feature_list)
for features in [tfidf_features_title,tfidf_features_essay]:
    tfidf_feature_list.extend(features)
tfidf_feature_list = np.array(tfidf_feature_list)
print_dimension_info(_obj=[final_x_train,final_x_cv,final_x_test],_name='The Fir
```

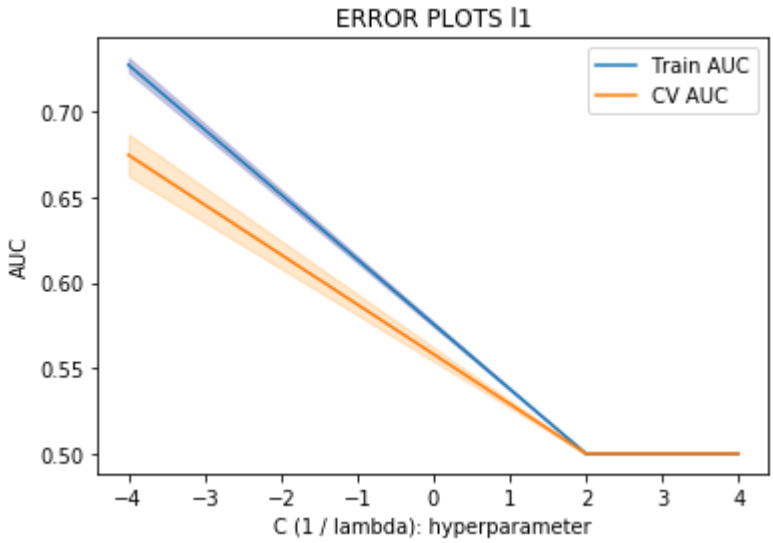
```
The Values for :  The Final Matrix dimension info after TFIDF

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 7378), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.spars
**********************************************************************
```

```
# _hypertuning(x_train,y_train,x_cv,y_cv,tune_type,penalty_type='l2',max_epoch =
# With L2 Regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv

```



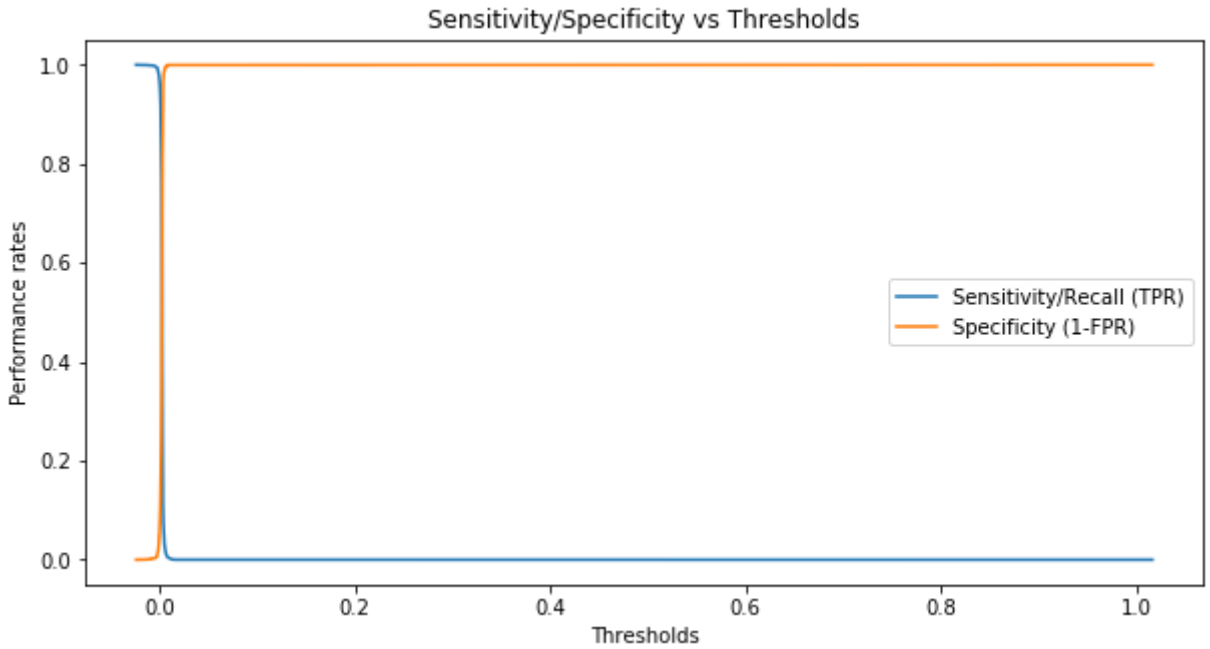

```
# With L1 Regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv
```

```
3
```



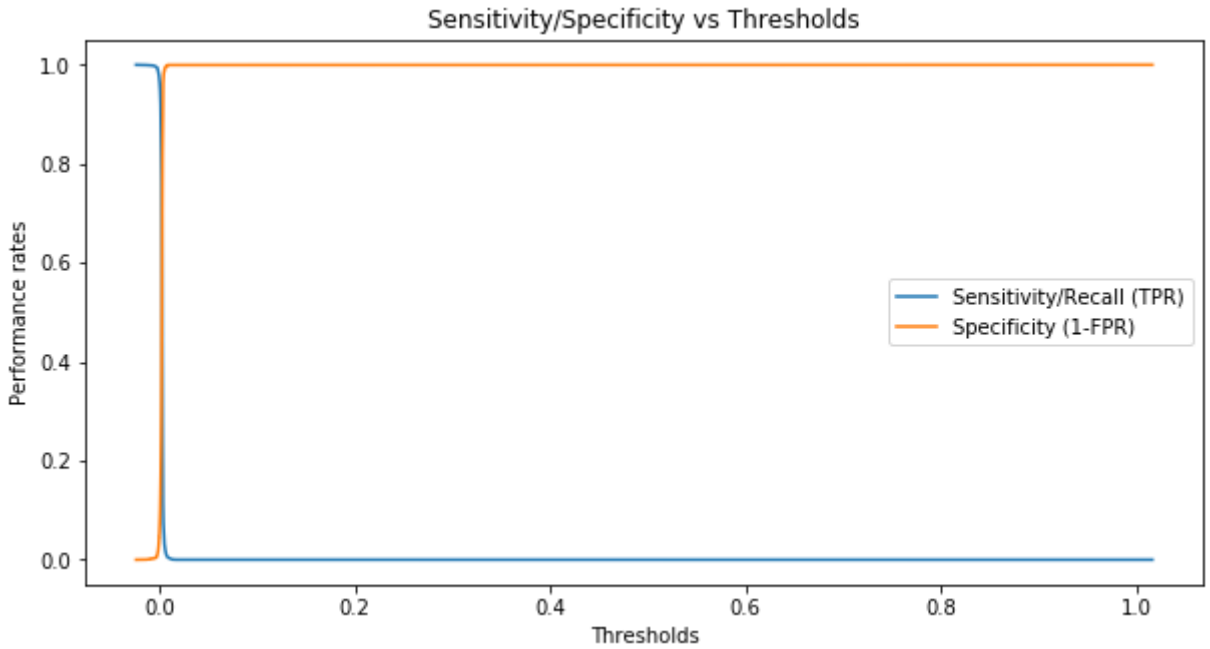

```
# fitting the SVM with the best c
from sklearn.linear_model import SGDClassifier
tfidf_c = 10**2
svc_clf = SGDClassifier(loss='hinge',penalty='l2',alpha=tfidf_c,max_iter=10000,l
svc_clf.fit(final_x_train,y_train)
# Deciding appropriate threshold with cross validation dataset
# evaluate_threshold(_c,x_train,y_train,x_cv,y_cv)
evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values)
```

Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.9995280792826805	0.0	-0.024464474949996033
0.9995280792826805	0.0	-0.022292529385211677
0.9995280792826805	0.0	-0.02012058382042732
0.9995280792826805	0.0	-0.017948638255642968
0.9995280792826805	0.0	-0.01577669269085861
0.9994606620373492	0.000755572346052169	-0.013604747126074254
0.9993932447920177	0.0011333585190782536	-0.0114328015612899
0.9986516550933728	0.0018889308651303116	-0.009260855996505543
0.9983819861120474	0.0026445032111824807	-0.007088910431721187
0.9966965549787635	0.0037778617302606232	-0.004916964866936831
0.9932582754668644	0.010200226671703838	-0.002745019302152474 5
0.971010584507517	0.05515678126180579	-0.0005730737373681183
0.7838603114676734	0.3543634302984511	0.0015988718274162345
0.056225982606350706	0.9799773328296184	0.0037708173922005907
0.011056428234342344	0.9962221382697394	0.005942762956984947
0.0043821209465381246	0.9992444276539478	0.008114708521769307
0.0005393379626508461	0.9996222138269739	0.01028665408655366
0.0005393379626508461	0.9996222138269739	0.012458599651338012
6.741724533135576e-05	1.0	0.014630545216122372
0.0	1.0	0.016802490780906728

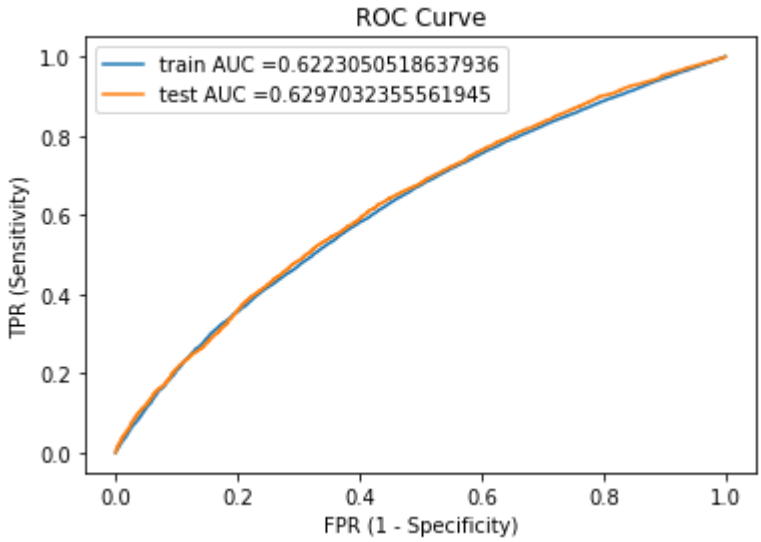



```
1 evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values,first=0.0015,las
```

Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.8077260163149733	0.3211182470721572	0.0015
0.7777253421425201	0.3607857952398942	0.0016157894736842106
0.7434773815141913	0.399319984888553	0.0017315789473684 21
0.704375379222005	0.454854552323385	0.0018473684210526316
0.6613631767006	0.5047223271628258	0.0019631578947368 42
0.6089125598328052	0.5640347563279184	0.0020789473684210526
0.5500573046585316	0.6301473366074801	0.0021947368421052 63
0.4829097283085013	0.6943709860219116	0.0023105263157894 74
0.4127957931638913	0.7563279183981866	0.0024263157894736844
0.3403222544326839	0.8133736305251228	0.0025421052631578 95
0.27209600215735186	0.8677748394408764	0.0026578947368421056
0.21283624351109015	0.904042312051379	0.0027736842105263162
0.16591384076046653	0.9293539856441254	0.0028894736842105264
0.13146362839614373	0.9463543634302984	0.0030052631578947 37
0.1088114339648082	0.9542878730638459	0.0031210526315789476
0.094181891727904	0.9622213826973933	0.0032368421052631578
0.08049619092563878	0.9656214582546279	0.0033526315789473684
0.0732825456751837	0.9705326785039667	0.0034684210526315 79
0.06472055551810153	0.9743105402342275	0.0035842105263157896
0.05905750691026765	0.9780884019644881	0.0037

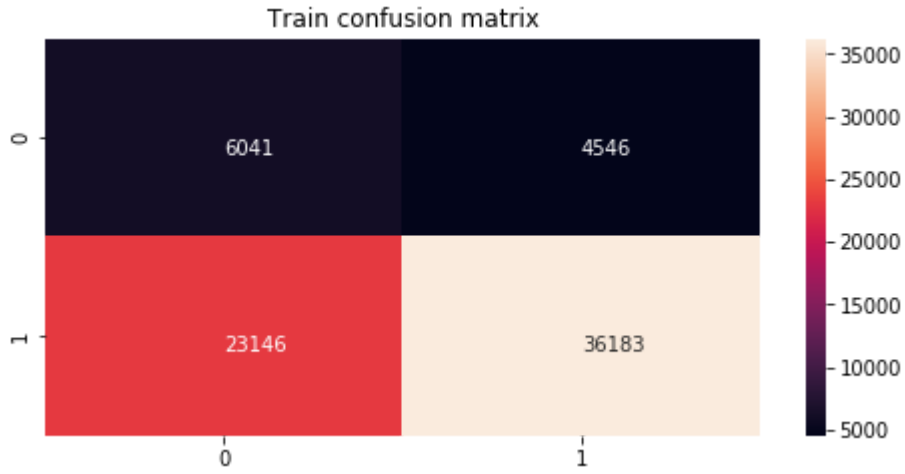


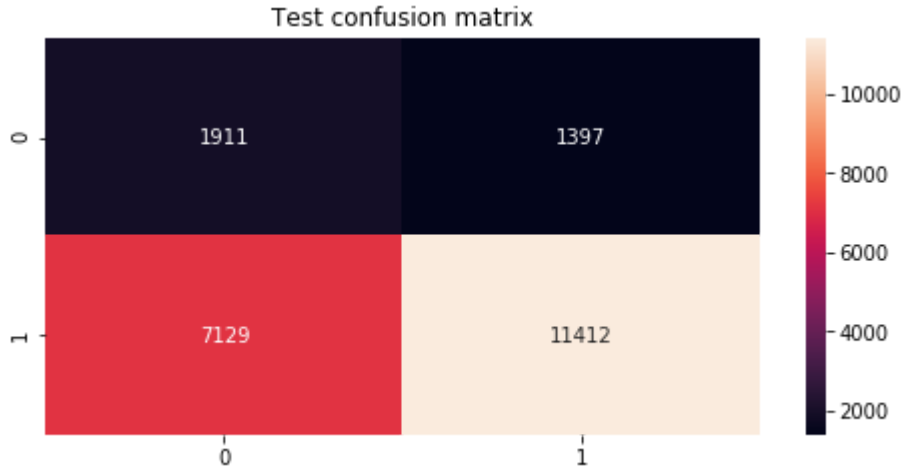

```
# model_gen_score(x_train,y_train,x_test,y_test,best_c,cutoff_val,clf,features_r
tfidf_train_auc,tfidf_test_auc = model_gen_score(svc=svc_clf,x_train = final_x_t
                                                 x_test=final_x_test,y_test=y_test.v

```



==========

Confusion matrix





2.4.3 Applying SVM brute force on AVG W2V, SET 3

```
1 # Using the pretrained word2vec glove model
2 # stronging variables into pickle files python: http://www.jessicayung.com/how-t
```

```
# make sure you have the glove_vectors file
with open('drive/My Drive/data/glove_vectors.dms', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

```

avgw2v_title_train,avgw2v_title_cv,avgw2v_title_test = vectorize_text(encoding_t
                                                                      cv_text =x_cv['titl
avgw2v_essay_train,avgw2v_essay_cv,avgw2v_essay_test = vectorize_text(encoding_t
                                                                      cv_text =x_cv['essa

```

```
100%|██████████| 69916/69916 [00:01<00:00, 59098.74it/s]
69916
300
100%|██████████| 17480/17480 [00:00<00:00, 66727.62it/s]
 29%|██▉       | 6444/21849 [00:00<00:00, 64429.62it/s]17480
300
100%|██████████| 21849/21849 [00:00<00:00, 68897.05it/s]
  1%|          | 366/69916 [00:00<00:19, 3656.08it/s]21849
300
100%|██████████| 69916/69916 [00:19<00:00, 3678.76it/s]
69916
300
100%|██████████| 17480/17480 [00:04<00:00, 3622.93it/s]
  0%|          | 0/21849 [00:00<?, ?it/s]17480
300
100%|██████████| 21849/21849 [00:05<00:00, 3726.03it/s]
21849
300
```

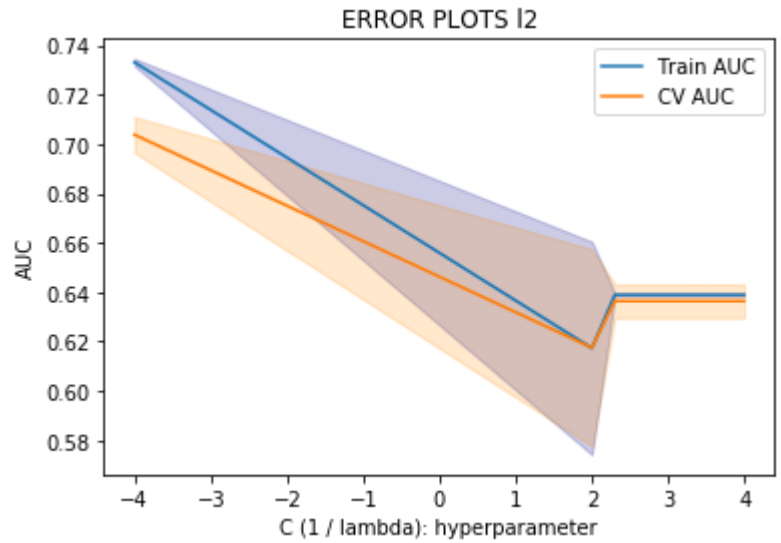
```
print_dimension_info(_obj=[avgw2v_title_train,avgw2v_title_cv,avgw2v_title_test,
```

```
The Values for :  Average W2V dim

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 300), ('Cross Validation count : '

Type of matrices:  [<class 'numpy.ndarray'>, <class 'numpy.ndarray'>, <class '
**************************************************************************
```

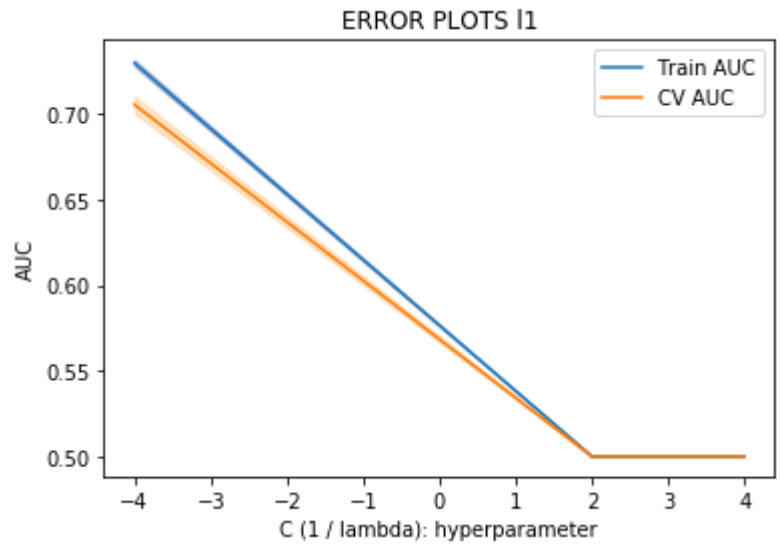
```
# This time we have np.hstack because we are not dealing with the sparse matrice

final_x_train = np.hstack((x_train_vec.todense(),avgw2v_title_train,avgw2v_essay
final_x_cv = np.hstack((x_cv_vec.todense(),avgw2v_title_cv,avgw2v_essay_cv))
final_x_test = np.hstack((x_test_vec.todense(),avgw2v_title_test,avgw2v_essay_te
print_dimension_info([final_x_train,final_x_cv,final_x_test],_name='The final ma
```

```
The Values for :  The final matrix of AVGW2V
```

```
# With L2 Regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv
```

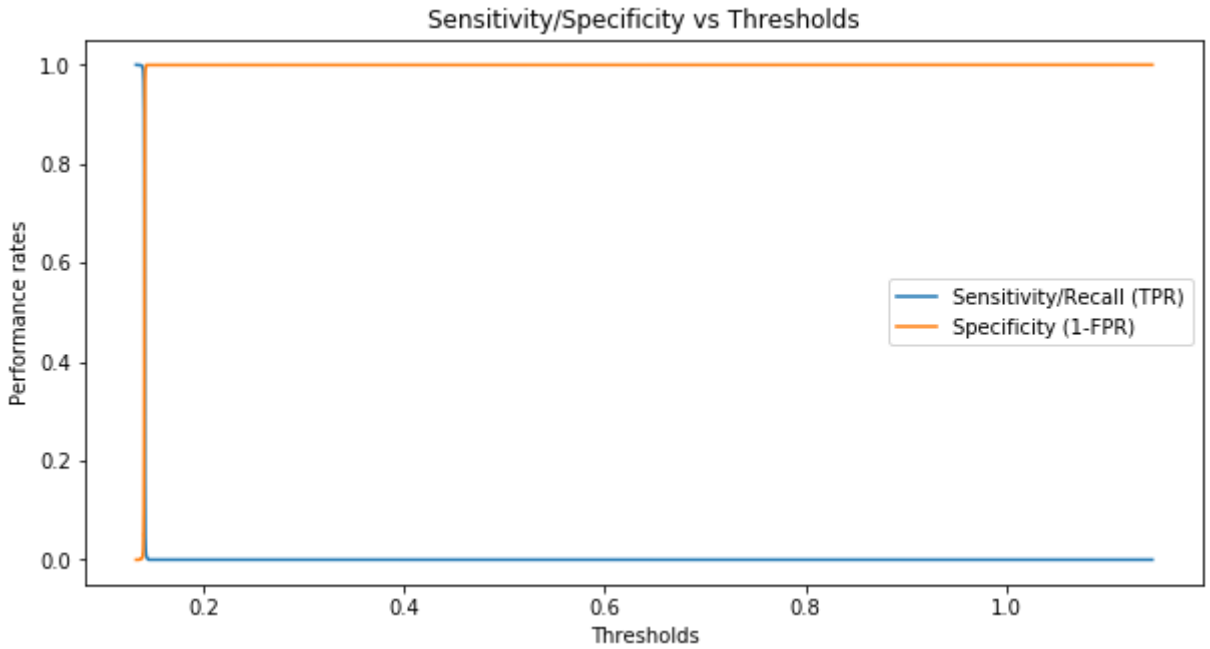



```
# With L1 Regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv
```



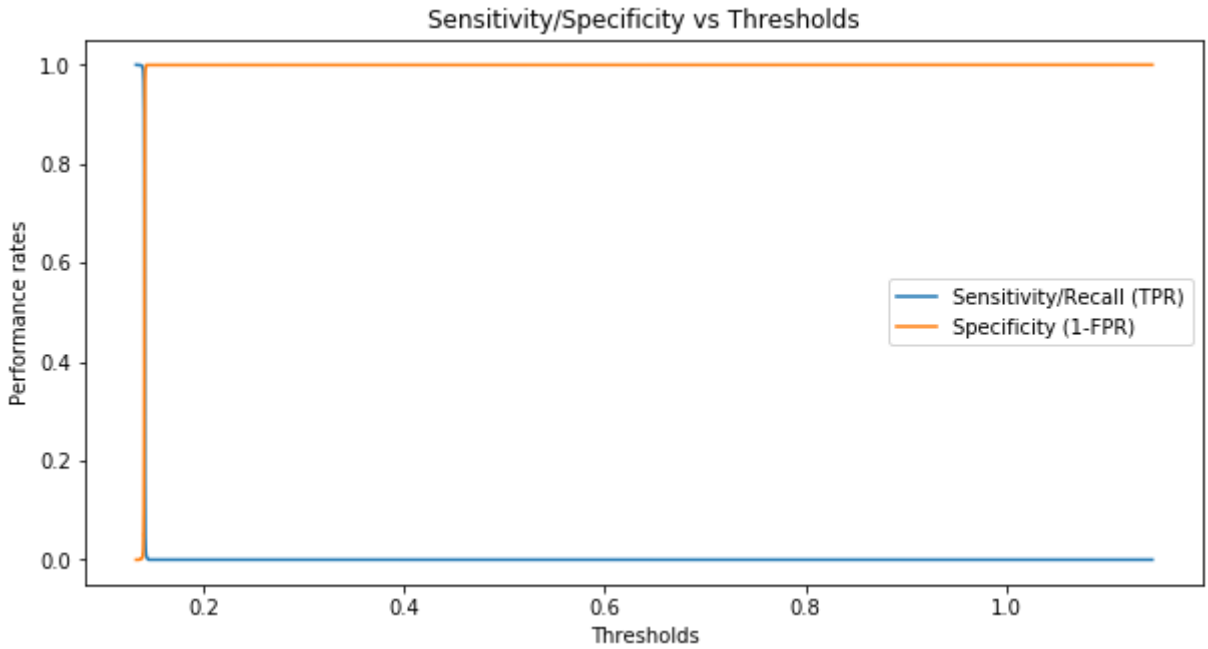

```
# fitting the Support vector machine with the best c

from sklearn.linear_model import SGDClassifier
avgw2v_c = 10**2.5
svc_clf = SGDClassifier(loss='hinge',penalty='l2',alpha=avgw2v_c,max_iter=10000,
svc_clf.fit(final_x_train,y_train)
# Deciding appropriate threshold with cross validation dataset
# evaluate_threshold(_c,x_train,y_train,x_cv,y_cv)
evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values)
```

Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.9994606620373492	0.0	0.13289348796076011
0.9994606620373492	0.0	0.1335763451277138
0.9994606620373492	0.0	0.13425920229466748
0.9994606620373492	0.0	0.1349420594616212
0.9994606620373492	0.0	0.13562491662857487
0.9994606620373492	0.000755572346052169	0.13630777379552855
0.9993932447920177	0.0011333585190782536	0.13699063096248223
0.9987864895840356	0.0018889308651303116	0.13767348812943592
0.9983145688667161	0.0026445032111824807	0.13835634529638963
0.9968988067147576	0.0037778617302606232	0.1390392024633433
0.9935279444481898	0.01133358519078198	0.139722059630297
0.971010584507517	0.06535700793350963	0.14040491679725067
0.7752309040652599	0.3940309784661882	0.14108777396420435
0.15141913301422505	0.938420853796751	0.14177063113115806
0.019281332164767746	0.9931998488855308	0.14245348829811175
0.0006067552079822018	0.9996222138269739	0.14313634546506543
0.0006067552079822018	0.9996222138269739	0.1438192026320191
0.0006067552079822018	0.9996222138269739	0.14450205979897282
6.741724533135576e-05	1.0	0.1451849169659265
0.0	1.0	0.14586777413288018

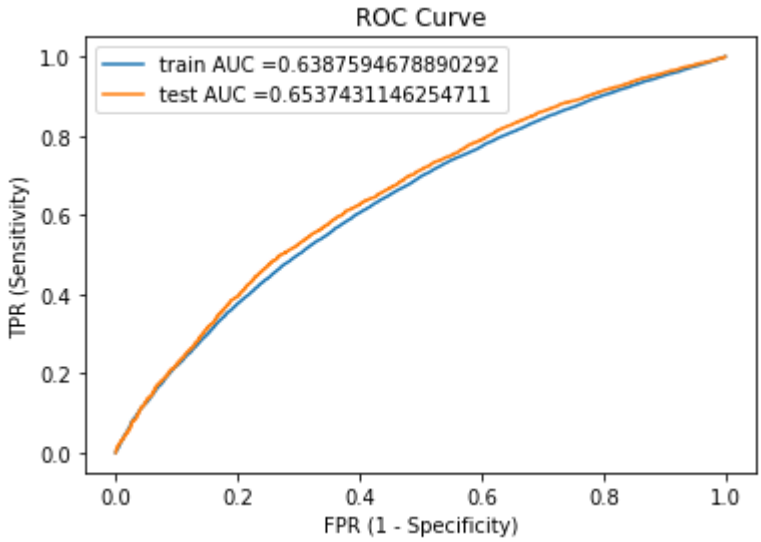



```
evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values,first=0.1410,las
```

Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.8334119867862199	0.3162070268228183	0.141
0.810490123373559	0.347941065357008	0.14103684210526315
0.7843322321849929	0.376275028333963	0.1410736842105263
0.758781096204409	0.41669814884775214	0.14111052631578946
0.730533270410571	0.4571212693615414	0.14114736842105263
0.6952740511022719	0.49754438987533056	0.14118421052631577
0.6604193352659611	0.5345674348318852	0.14122105263157894
0.625362367693656	0.5719682659614658	0.14125789473684208
0.5865300343827952	0.6127691726482811	0.14129473684210525
0.54500101125868	0.6471477143936533	0.14133157894736842
0.5076518573451089	0.6845485455232339	0.14136842105263156
0.4681453515809344	0.7196826596146582	0.14140526315789473
0.4268859974381447	0.7438609746883265	0.1414421052631579
0.38825591586327785	0.7752172270494899	0.14147894736842104
0.3505022584777186	0.8073290517567058	0.1415157894736842
0.31490595294276275	0.8341518700415564	0.14155263157894735
0.28045574057843997	0.8621080468454855	0.14158947368421052
0.24789321108339513	0.8862863619191538	0.14162631578947368
0.22038697498820198	0.898753305629014	0.14166315789473682
0.19261106991168342	0.915753683415187	0.1417

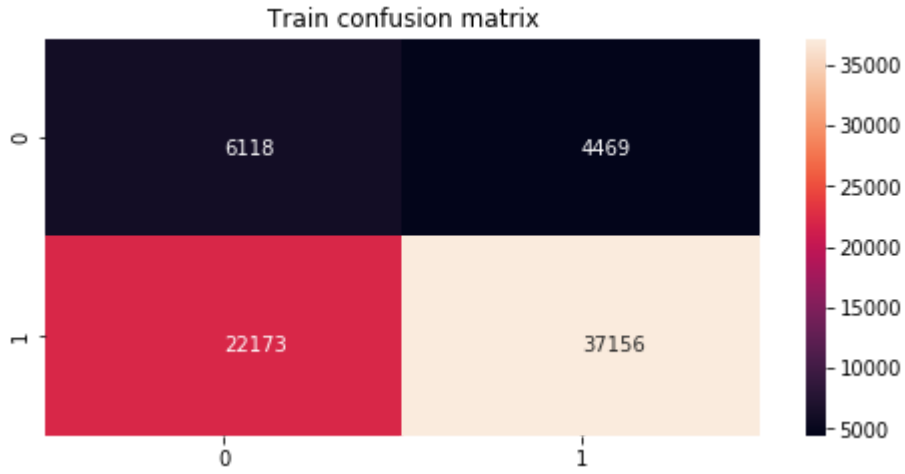


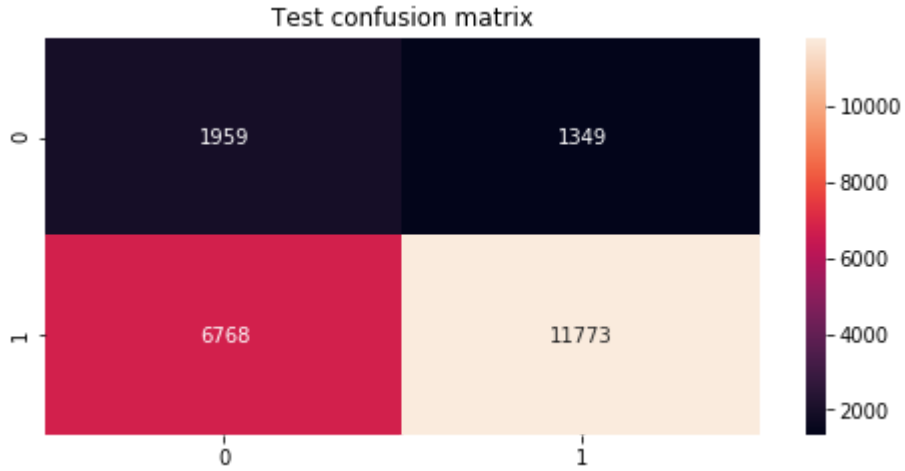

```
1
2 # model generalization on test data
3 avgw2v_train_auc,avgw2v_test_auc = model_gen_score(svc= svc_clf,x_train = final_
4                                                    x_test=final_x_test,y_test=y_test.v
5
```



==========

Confusion matrix





2.4.4 Applying SVM brute force on TFIDF W2V, SET 4

```
# Please write all the code with proper documentation
```

```
tfidfw2v_title_train,tfidfw2v_title_cv,tfidfw2v_title_test = vectorize_text(encc
                                                             cv_text =x_cv['titl
tfidfw2v_essay_train,tfidfw2v_essay_cv,tfidfw2v_essay_test = vectorize_text(encc
                                                             cv_text =x_cv['essa
```

```
100%|██████████| 69916/69916 [00:00<00:00, 244343.59it/s]
The number of rows are:  69916
The number of columns are:  300
100%|██████████| 17480/17480 [00:00<00:00, 258298.34it/s]
100%|██████████| 21849/21849 [00:00<00:00, 263174.61it/s]
The number of rows are:  17480
The number of columns are:  300
The number of rows are:  21849
The number of columns are:  300
100%|██████████| 69916/69916 [00:02<00:00, 26398.71it/s]
The number of rows are:  69916
The number of columns are:  300
100%|██████████| 17480/17480 [00:00<00:00, 26241.76it/s]
 11%|█         | 2300/21849 [00:00<00:00, 22992.62it/s]The number of rows are:
The number of columns are:  300
100%|██████████| 21849/21849 [00:00<00:00, 24707.93it/s]
The number of rows are:  21849
The number of columns are:  300
```

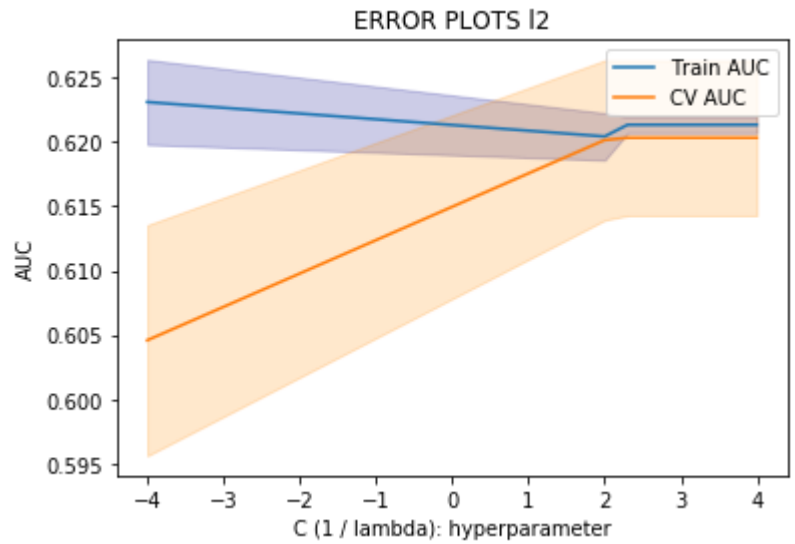
```
print_dimension_info(_obj=[tfidfw2v_title_train,tfidfw2v_title_cv,tfidfw2v_title
                    _name='TFIDF W2V Dimensions')
```

```
The Values for :  TFIDF W2V Dimensions

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 300), ('Cross Validation count : '

Type of matrices:  [<class 'numpy.ndarray'>, <class 'numpy.ndarray'>, <class '
**************************************************************************
```

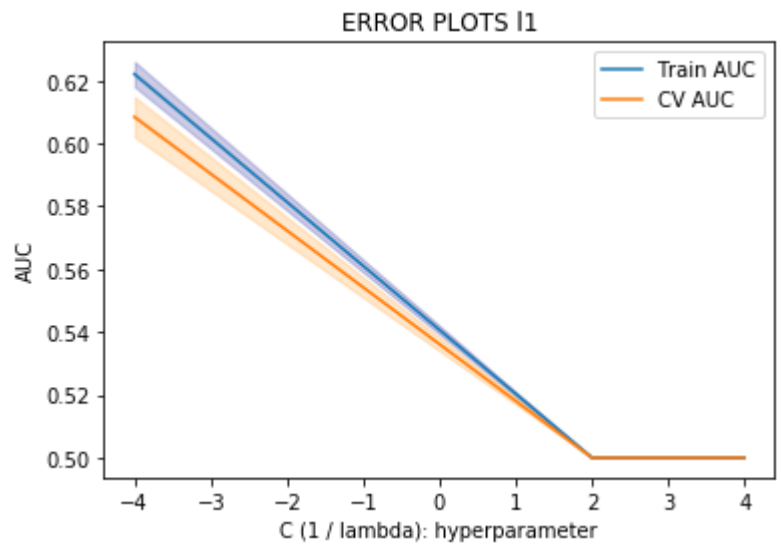
```
# This time we have np.hstack because we are not dealing with the sparse matrice

final_x_train = np.hstack((x_train_vec.todense(),tfidfw2v_title_train,tfidfw2v_e
final_x_cv = np.hstack((x_cv_vec.todense(),tfidfw2v_title_cv,tfidfw2v_essay_cv))
final_x_test = np.hstack((x_test_vec.todense(),tfidfw2v_title_test,tfidfw2v_essa
print_dimension_info([final_x_train,final_x_cv,final_x_test],_name='The final ma
```

```
The Values for :  The final matrix of AVGW2V

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 1090), ('Cross Validation count :

Type of matrices:  [<class 'numpy.matrix'>, <class 'numpy.matrix'>, <class 'nu
**************************************************************************
```

```
# With L2 Regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv

```

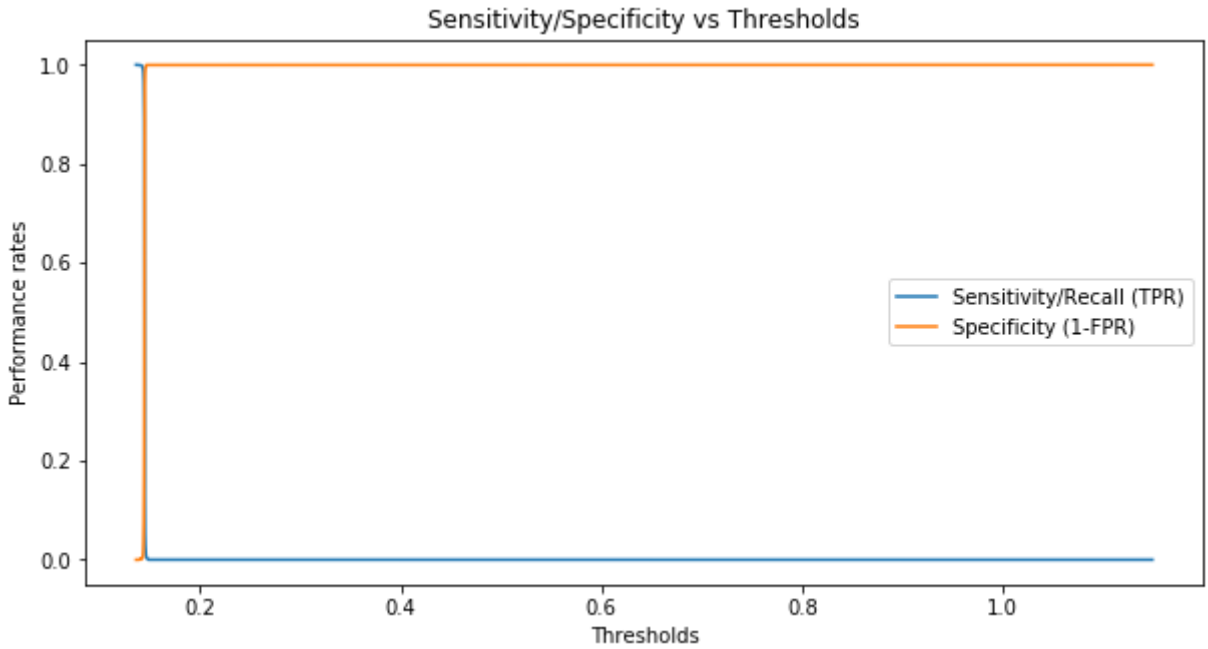



```
.th L1 Regularizer
ertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv=y_
```



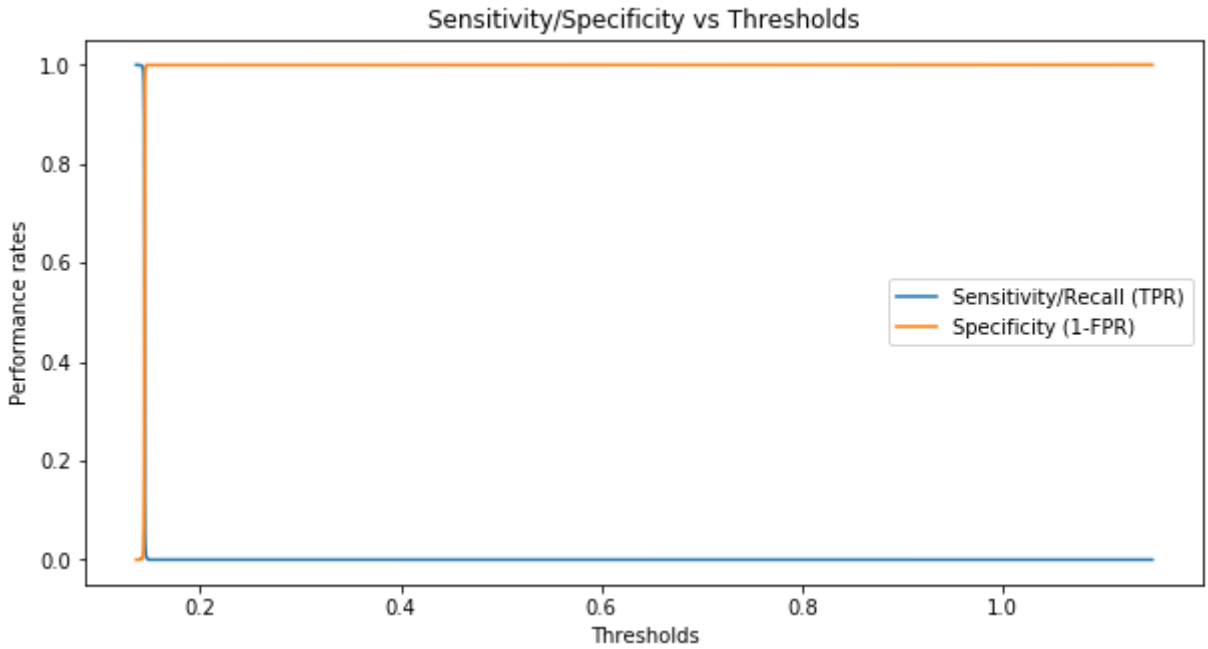

```
# fitting the Support vector machine with the best c

from sklearn.linear_model import SGDClassifier
tfidfw2v_c = 10**2.5
svc_clf = SGDClassifier(loss='hinge',penalty='l2',alpha=tfidfw2v_c,max_iter=100
svc_clf.fit(final_x_train,y_train)
# Deciding appropriate threshold with cross validation dataset
# evaluate_threshold(_c,x_train,y_train,x_cv,y_cv)
evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values)
```

Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.9995280792826805	0.0	0.136166128993629
0.9995280792826805	0.0	0.13685283499432255
0.9995280792826805	0.0	0.13753954099501609
0.9995280792826805	0.0	0.13822624699570962
0.9995280792826805	0.0	0.13891295299640316
0.9994606620373492	0.000755572346052169	0.13959965899709673
0.9993932447920177	0.0011333585190782536	0.14028636499779026
0.9986516550933728	0.0018889308651303116	0.1409730709984838
0.9983819861120474	0.0026445032111824807	0.14165977699917734
0.9966965549787635	0.0037778617302606232	0.14234648299987088
0.9938650306748467	0.010200226671703838	0.14303318900056441
0.9711454189981797	0.0547789950887797	0.14371989500125795
0.7844670666756556	0.3509633547412164	0.1444066010019515
0.056225982606350706	0.9795995466565923	0.14509330700264503
0.011056428234342344	0.9962221382697394	0.1457800130033386
0.0043821209465381246	0.9992444276539478	0.14646671900403213
0.0005393379626508461	0.9996222138269739	0.14715342500472567
0.0005393379626508461	0.9996222138269739	0.1478401310054192
6.741724533135576e-05	1.0	0.14852683700611274
0.0	1.0	0.14921354300680628

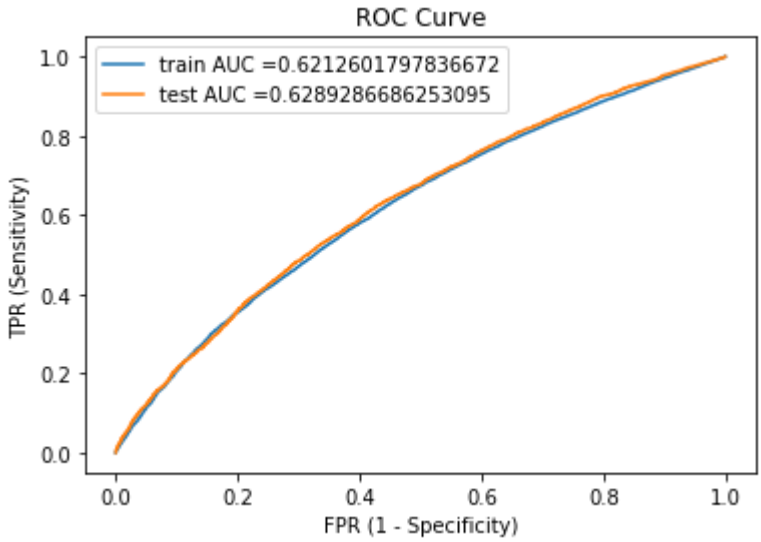



```
1 evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values,first=0.144,last
```

Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.94181891727904	0.11258027956176808	0.144
0.9327175891593069	0.12995844352096708	0.14405263157894735
0.9204476505090002	0.1564034756327919	0.14410526315789474
0.9056158565361019	0.17491499811106914	0.1441578947368421
0.8896379693925707	0.19682659614658105	0.14421052631578946
0.8698847165104834	0.22742727616169245	0.14426315789473684
0.8440639115485741	0.2678503966754817	0.1443157894736842
0.8150744960560912	0.3120513789195315	0.14436842105263156
0.770511696892065	0.3675859463543635	0.14442105263157895
0.7172520730802939	0.43369852663392516	0.1444736842105263
0.6565765522820738	0.5096335474121647	0.14452631578947367
0.5779006269803816	0.5976577257272384	0.14457894736842103
0.48392098698847164	0.6921042689837552	0.14463157894736842
0.38394121216207105	0.7797506611258028	0.14468421052631578
0.28200633722106117	0.8624858330185116	0.14473684210526314
0.19928537719948763	0.9123536078579524	0.14478947368421052
0.1408346254972022	0.9425765017000378	0.14484210526315788
0.10618216139688533	0.9550434454098979	0.14489473684210524
0.08568731881615317	0.9637325273894976	0.14494736842105263
0.073147711184521	0.9705326785039667	0.145

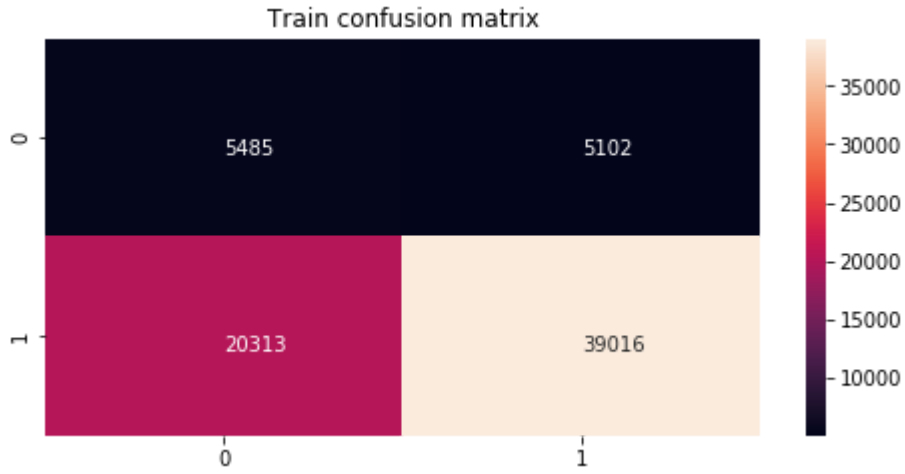


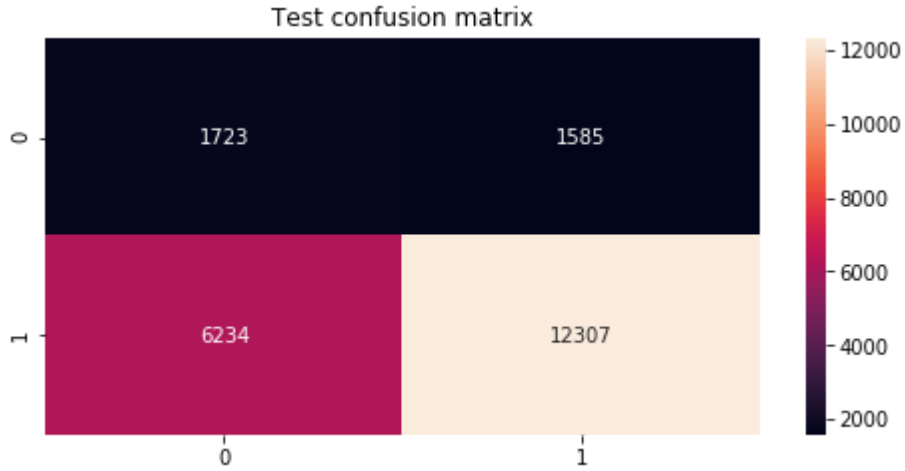

```
1
2 # model generalization on test data
3 tfidfw2v_train_auc,tfidfw2v_test_auc = model_gen_score(svc=svc_clf,x_train = fir
4                                                        x_test=final_x_test,y_test=y_test.v
5
```



==========

Confusion matrix





2.5 Support Vector Machines with added Features `Set 5`

```
1 project_data_.head(2)
```

	school_state	project_grade_category	clean_categories	clean_subc
0	CA	Grades_PreK_2	Math_Science	AppliedSciences__Health_
1	UT	Grades_3_5	SpecialNeeds	S

```
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
# calculating the polarity score for train essay
xtrain_sent = pd.DataFrame(list(x_train['essay'].map(lambda x : sid.polarity_sco
xcv_sent = pd.DataFrame(list(x_cv['essay'].map(lambda x : sid.polarity_scores(x)
xtest_sent_df = pd.DataFrame(list(x_test['essay'].map(lambda x : sid.polarity_sc

```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

```
# number of words in title
xtrain_title_count = x_train['title'].map(lambda x : len(x.split())).values.resh
xcv_title_count = x_cv['title'].map(lambda x : len(x.split())).values.reshape(-1
xtest_title_count = x_test['title'].map(lambda x : len(x.split())).values.reshap
```

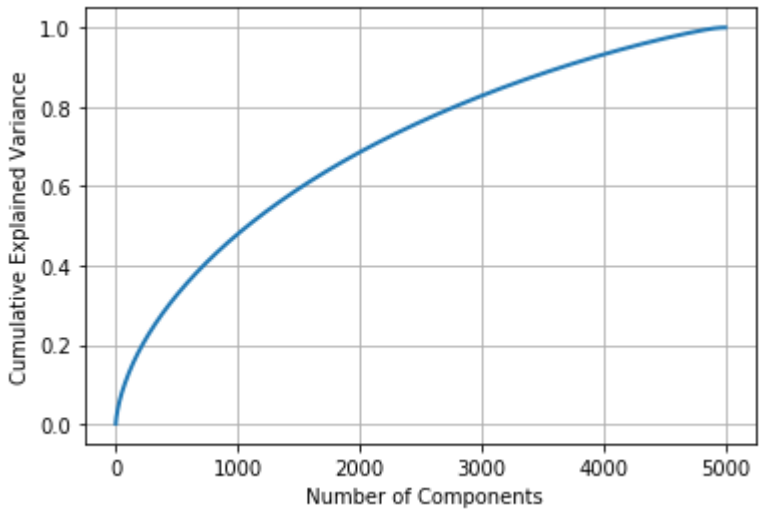
```
# number of words count in essay
xtrain_essay_count = x_train['essay'].map(lambda x : len(x.split())).values.resh
xcv_essay_count = x_cv['essay'].map(lambda x : len(x.split())).values.reshape(-1
xtest_essay_count = x_test['essay'].map(lambda x : len(x.split())).values.reshap
```

Truncated SVD on TFIDF Vectorizer set

```
from sklearn.decomposition import TruncatedSVD
```

```

svd = TruncatedSVD(n_components=tfidf_essay_train.shape[1]-1)
svd.components_ = tfidf_essay_train.shape[1]
# fitting to  Truncated SVD class
svd_data = svd.fit_transform(tfidf_essay_train,y_train.values)
```

```
svd_data = svd.fit_transform(tfidf_essay_train,y_train.values)

percentage_var_explained = svd.explained_variance_ratio_
cum_var_explained = np.cumsum(percentage_var_explained)

# plotting the cum_var_explained
plt.figure(1,figsize=(6,4))
plt.clf() # clear the current figure
plt.plot(cum_var_explained,linewidth=2)
plt.axis('tight')
plt.grid()
plt.ylabel("Cumulative Explained Variance")
plt.xlabel(" Number of Components")
plt.show()
```




```
# transforming the fitted Truncated SVD and with n_components with elbow method
svd_trunc = TruncatedSVD(n_components = 2800) # preserving the variance more tha
svd_trunc.fit(tfidf_essay_train,y_train.values)
# Transformation with required components
svd_train = svd_trunc.transform(tfidf_essay_train)
svd_cv = svd_trunc.transform(tfidf_essay_cv)
svd_test = svd_trunc.transform(tfidf_essay_test)
```

```
type(svd_train)
```

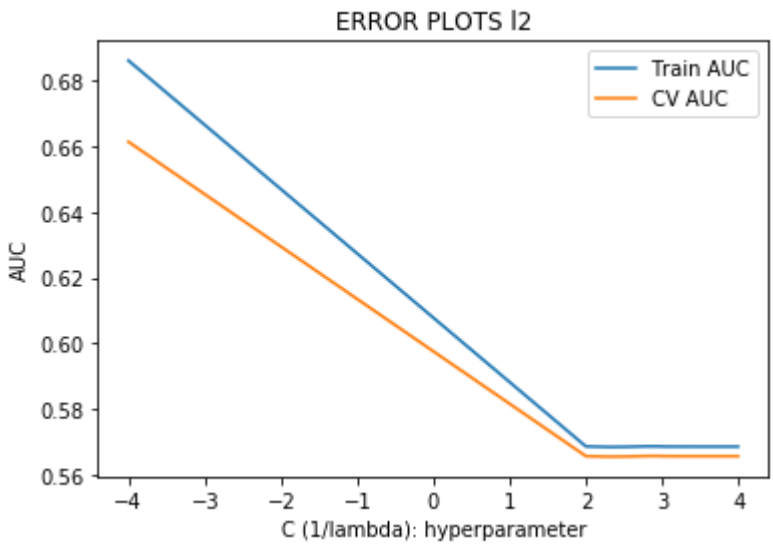
```
numpy.ndarray
```

```
# hstacking all the newly formed features to x matrices
final_x_train = hstack((x_train_vec,xtrain_sent,xtrain_title_count,xtrain_essay_
final_x_cv = hstack((x_cv_vec,xcv_sent,xcv_title_count,xcv_essay_count,svd_cv),f
final_x_test = hstack((x_test_vec,xtest_sent_df,xtest_title_count,xtest_essay_co
```

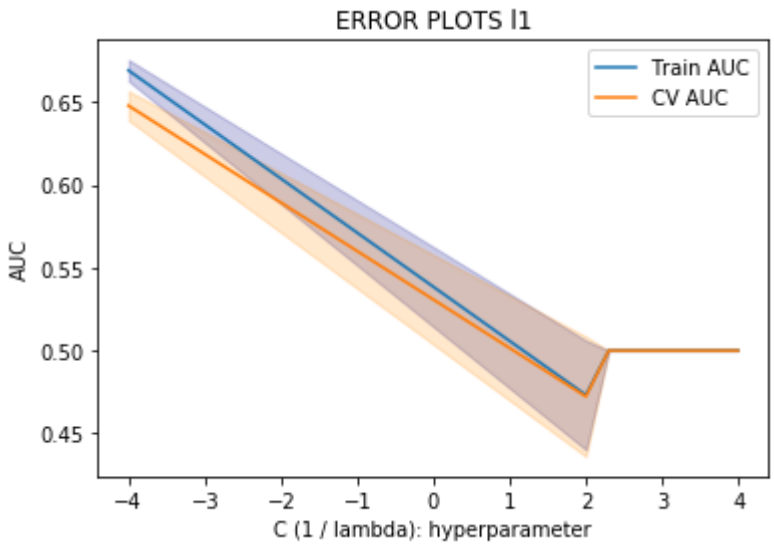
```
print_dimension_info(_obj=[final_x_train,final_x_cv,final_x_test],_name='set 5 f
```

```
The Values for :  set 5 final features dimensions

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : '
```

```
# hyper tuning through cross validation with custom model as gridsearch is throw

# With L2 Regularizer
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv
```



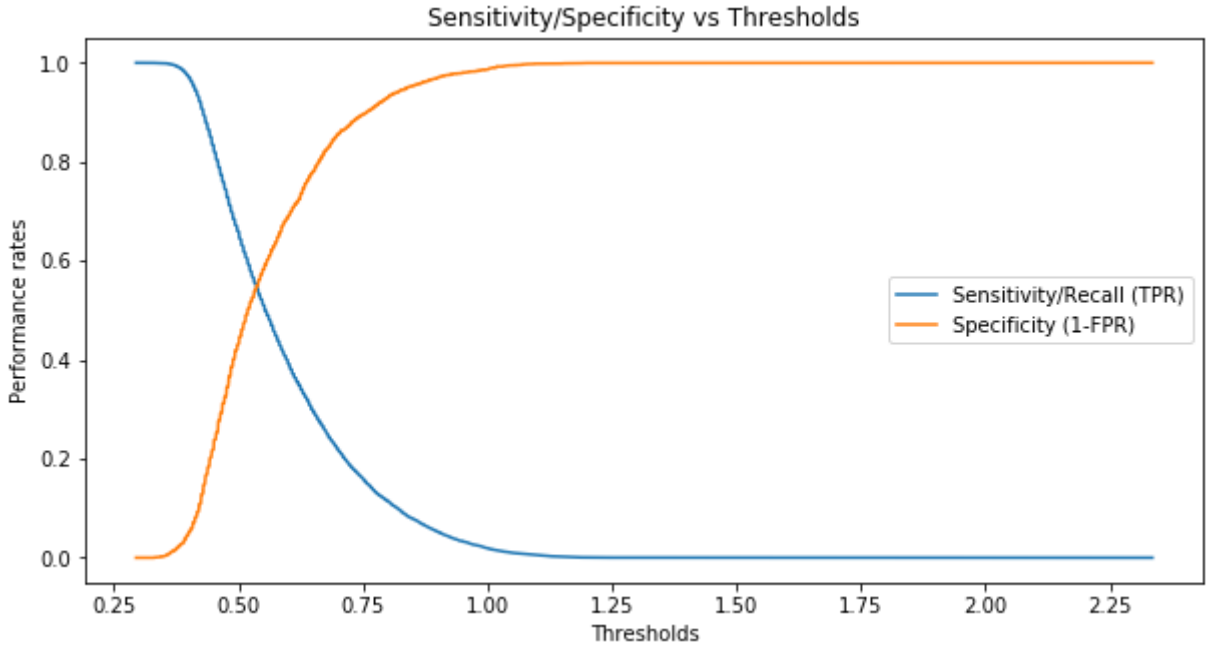

```
# With L1 Regularizer with 5000 iterations only as it is taking forever, The coc
_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv
```



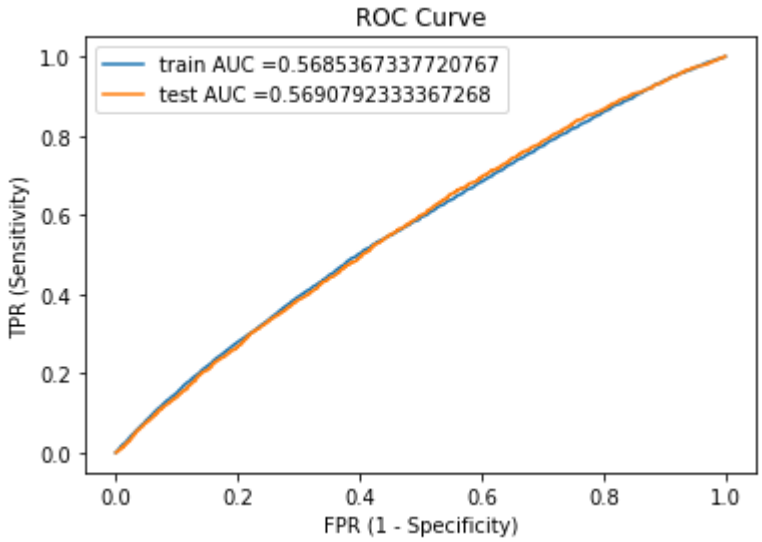

```
# fitting the Support vector machine with the best c

from sklearn.linear_model import SGDClassifier
set5_c = 10**2.5
svc_clf = SGDClassifier(loss='hinge',penalty='l2',alpha=set5_c,max_iter=10000,le
svc_clf.fit(final_x_train,y_train)
# Deciding appropriate threshold with cross validation dataset
# evaluate threshold( c,x train,y train,x cv,y cv)
```

```
evaluate_threshold(clf=svc_clf,x_cv=final_x_cv,y_cv=y_cv.values)
```

Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value
0.9997977482640059	0.0	0.29455807021738917
0.9989887413200297	0.003022289384208565	0.34931209728247115
0.9656846221263399	0.0570457121269361	0.4040661243475531
0.8013887952538259	0.258783528522856	0.4588201514126351
0.6167329602912425	0.4771439365319229	0.513574178477717
0.468819524034248	0.6225916131469589	0.5683282055427991
0.3485471583631093	0.7249716660370231	0.6230822326078811
0.2512640733499629	0.8254627880619569	0.677836259672963
0.17717252073080295	0.8832640725349452	0.7325902867380449
0.12303647272972426	0.9210426898375519	0.787344313803127
0.08150744960560911	0.9516433698526634	0.842098340868209
0.05346187554776512	0.9686437476388364	0.8968523679332909
0.031753522551068565	0.9803551190026445	0.9516063949983729
0.01712398031416436	0.9897997733282962	1.0063604220634548
0.007887817703768623	0.9958443520967133	1.061114449128537
0.002898941549248298	0.9981110691348697	1.1158684761936188
0.0006741724533135576	0.9992444276539478	1.1706225032587008
0.00026966898132542303	0.9996222138269739	1.2253765303237827
6.741724533135576e-05	1.0	1.2801305573888648
0.0	1.0	1.3348845844539468

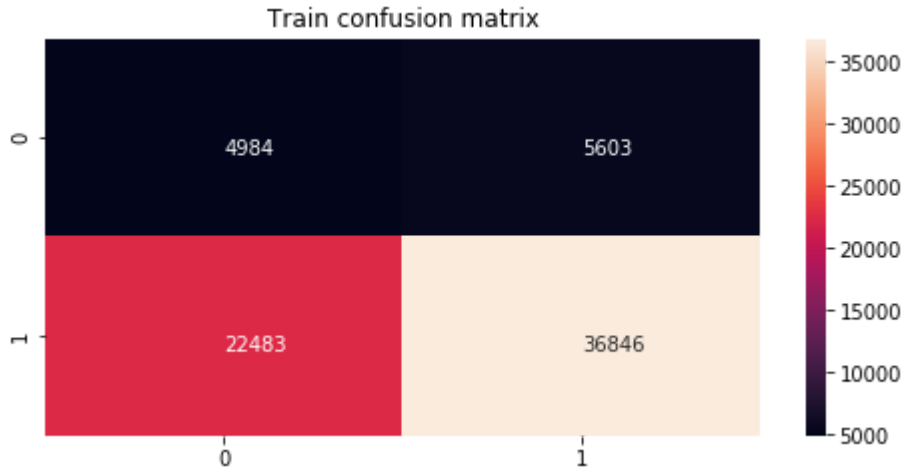


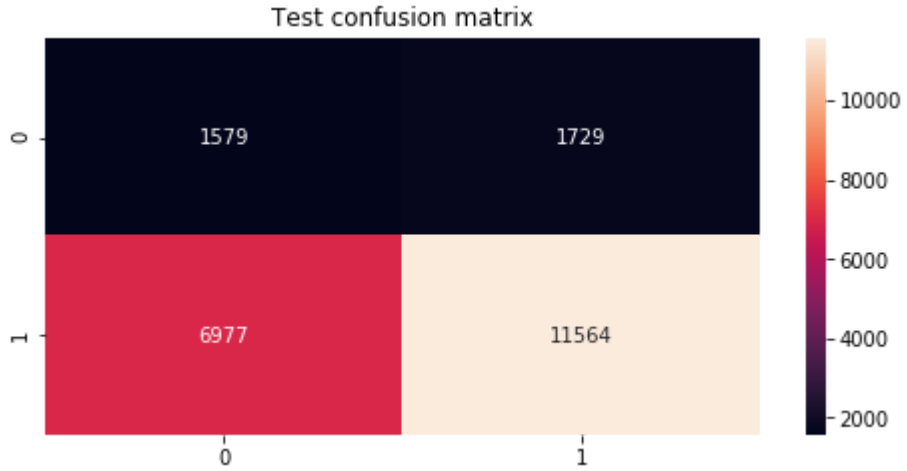

```
ı,x_test,y_test,best_c,cutoff_val,clf,features_names)
ɔdel_gen_score(svc = svc_clf,x_train = final_x_train,y_train=y_train.values,x_te

```



==========

Confusion matrix





4. Conclusions

```
1 # Please compare all your models using Prettytable library
```

```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ['Vectorizer','Model',' Hyperparameters','TEST AUC']

x.add_row(['BOW','BRUTE',bow_c,bow_test_auc])
x.add_row(['TFIDF','BRUTE',tfidf_c,tfidf_test_auc])
x.add_row(['AVGW2V','BRUTE',avgw2v_c,avgw2v_test_auc])
x.add_row(['TFIDFW2V','BRUTE',tfidfw2v_c,tfidfw2v_test_auc])
x.add_row(['set5 features','BRUTE',set5_c,set5_test_auc])

print(x)

```

```
+---------------+-------+--------------------+--------------------+
|   Vectorizer  | Model |   Hyperparameters  |      TEST AUC      |
+---------------+-------+--------------------+--------------------+
|      BOW      | BRUTE |        100         | 0.6571615655933479 |
|     TFIDF     | BRUTE |        100         | 0.6297032355561945 |
|     AVGW2V    | BRUTE | 316.22776601683796 | 0.6537431146254711 |
|    TFIDFW2V   | BRUTE | 316.22776601683796 | 0.6289286686253095 |
| set5 features | BRUTE | 316.22776601683796 | 0.5690792333367268 |
+---------------+-------+--------------------+--------------------+
```

Conclusion

* The BOW and AVGW2V model has similar performance.
* AVGW2V models are performing slighly better on test set, According to this paper https://www.researchgate.net/publication/310463971_An_Analysis_on_Better_Testing_than_Training_Performa perform slighly better on test set because it is able to correctly identify the data points near the boundary i.e. Su
* L2 Regularizer performs better than L1 regularizer
* Model with Truncated SVD and L2 Regularizer is underfitting with almost 80% variance preserved.