Objective: To train the model using Support Vector Machines.

In [0]:

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

```
Drive already mounted at /content/drive; to at
tempt to forcibly remount, call drive.mount("/
content/drive", force_remount=True).
```

In [0]:

```
%cd /content/drive/My Drive
```

```
/content/drive/My Drive
```

In [0]:

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

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

from sklearn.feature_extraction.text import CountVectorizer
```

```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

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

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

from tqdm import tqdm
import os

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

Reading Data

In [0]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [0]:

```
print("Number of data points in train data", project_data.shape)
print(project_data.columns)
```

```
Number of data points in train data (109248, 17)
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category',
       'project_subject_categories', 'project_subject_subcategories',
       'project_title', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved'],
      dtype='object')
```

In [0]:

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


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

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


project_data.head(2)
```

Out[0]:

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

In [0]:

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

```
Number of data points in resource data (154127
2, 4)
Number of data points in resource data Index([
'id', 'description', 'quantity', 'price'], dty
pe='object')
```

Out[0]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-	1	149.00

		Space Mobile Drying Rack		
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [0]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[0]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [0]:

```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [0]:

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

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

In [0]:

```
project_data.shape
```

Out[0]:

```
(109248, 19)
```

1.1 Preprocessing of project_subject_categories

In [0]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The
```

```
" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc
", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the &
 value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inp
lace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv
: kv[1]))
```

1.2 Preprocessing of project_subject_subcategories

In [0]:

```
sub_catogories = list(project_data['project_subject_subcatego
ries'].values)
# remove special characters from list of strings python: http
s://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-pyth
on/
# https://stackoverflow.com/questions/23669024/how-to-strip-a
```

```
-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whit
espace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth
, Care & Hunger"
    for j in i.split(','): # it will split it in three parts
["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            j=j.replace('The','') # if we have the words "The
" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc
", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1,
inplace=True)

# count of all the words in corpus python: https://stackoverf
low.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=l
ambda kv: kv[1]))
```

2.Text Preprocessing

2.1 Preprocessing of essay

In [0]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [0]:

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

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

In [0]:

```
# 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)
```

```
I have been fortunate enough to use the Fairy
Tale STEM kits in my classroom as well as the
STEM journals, which my students really enjoye
d.  I would love to implement more of the Lake
shore STEM kits in my classroom for the next s
chool year as they provide excellent and engag
ing STEM lessons.My students come from a varie
ty of backgrounds, including language and soci
oeconomic status.  Many of them don't have a l
ot of experience in science and engineering an
d these kits give me the materials to provide
these exciting opportunities for my students.E
ach month I try to do several science or STEM/
STEAM projects.  I would use the kits and robo
t to help guide my science instruction in enga
ging and meaningful ways.  I can adapt the kit
s to my current language arts pacing guide whe
re we already teach some of the material in th
e kits like tall tales (Paul Bunyan) or Johnny
 Appleseed.  The following units will be taugh
t in the next school year where I will impleme
nt these kits: magnets, motion, sink vs. float
, robots.  I often get to these units and don'
t know If I am teaching the right way or using
 the right materials.    The kits will give me
```

additional ideas, strategies, and lessons to prepare my students in science.It is challenging to develop high quality science activities. These kits give me the materials I need to provide my students with science activities that will go along with the curriculum in my clasroom. Although I have some things (like magnets) in my classroom, I don't know how to use them effectively. The kits will provide me with the right amount of materials and show me how to use them in an appropriate way.

==========

I teach high school English to students with learning and behavioral disabilities. My students all vary in their ability level. However, the ultimate goal is to increase all students literacy levels. This includes their reading, writing, and communication levels.I teach a really dynamic group of students. However, my students face a lot of challenges. My students all live in poverty and in a dangerous neighborhood. Despite these challenges, I have students who have the the desire to defeat these challenges. My students all have learning disabilities and currently all are performing below grade level. My students are visual learners and will benefit from a classroom that fulfills their preferred learning style.The materials I am requesting will allow my students to be prepared for the classroom with the necessary supplies. Too often I am challenged with students who come to school unprepared for class due to economic challenges. I want my students to be able to focus on learning and not how they will be able to get school supplies. The supplies will last all year. Students will be abl

e to complete written assignments and maintain a classroom journal. The chart paper will be used to make learning more visual in class and to create posters to aid students in their learning. The students have access to a classroom printer. The toner will be used to print student work that is completed on the classroom Chromebooks.I want to try and remove all barriers for the students learning and create opportunities for learning. One of the biggest barriers is the students not having the resources to get pens, paper, and folders. My students will be able to increase their literacy skills because of this project.

==========

\"Life moves pretty fast. If you don't stop and look around once in awhile, you could miss it.\" from the movie, Ferris Bueller's Day Off. Think back...what do you remember about your grandparents? How amazing would it be to be able to flip through a book to see a day in their lives?My second graders are voracious readers! They love to read both fiction and nonfiction books. Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson. They also love to read about insects, space and plants. My students are hungry bookworms! My students are eager to learn and read about the world around them. My kids love to be at school and are like little sponges absorbing everything around them. Their parents work long hours and usually do not see their children. My students are usually cared for by their grandparents or a family friend. Most of my students do not have someone who speaks English at home. Thus it is difficult for

my students to acquire language.Now think forw
ard... wouldn't it mean a lot to your kids, ni
eces or nephews or grandchildren, to be able t
o see a day in your life today 30 years from n
ow? Memories are so precious to us and being a
ble to share these memories with future genera
tions will be a rewarding experience. As part
 of our social studies curriculum, students wi
ll be learning about changes over time. Stude
nts will be studying photos to learn about how
 their community has changed over time. In pa
rticular, we will look at photos to study how
the land, buildings, clothing, and schools hav
e changed over time. As a culminating activit
y, my students will capture a slice of their h
istory and preserve it through scrap booking.
Key important events in their young lives will
 be documented with the date, location, and na
mes. Students will be using photos from home
 and from school to create their second grade
memories. Their scrap books will preserve th
eir unique stories for future generations to e
njoy.Your donation to this project will provid
e my second graders with an opportunity to lea
rn about social studies in a fun and creative
manner. Through their scrapbooks, children wi
ll share their story with others and have a hi
storical document for the rest of their lives.
==========
====
\"A person's a person, no matter how small.\"
(Dr.Seuss) I teach the smallest students with
the biggest enthusiasm for learning. My studen
ts learn in many different ways using all of o
ur senses and multiple intelligences. I use a
wide range of techniques to help all my studen
ts succeed. \r\nStudents in my class come from

a variety of different backgrounds which make s for wonderful sharing of experiences and cul tures, including Native Americans.\r\nOur scho ol is a caring community of successful learner s which can be seen through collaborative stud ent project based learning in and out of the c lassroom. Kindergarteners in my class love to work with hands-on materials and have many dif ferent opportunities to practice a skill befor e it is mastered. Having the social skills to work cooperatively with friends is a crucial a spect of the kindergarten curriculum.Montana i s the perfect place to learn about agriculture and nutrition. My students love to role play in our pretend kitchen in the early childhood classroom. I have had several kids ask me, \"C an we try cooking with REAL food?\" I will tak e their idea and create \"Common Core Cooking Lessons\" where we learn important math and wr iting concepts while cooking delicious healthy food for snack time. My students will have a grounded appreciation for the work that went i nto making the food and knowledge of where the ingredients came from as well as how it's hea lthy for their bodies. This project would expa nd our learning of nutrition and agricultural cooking recipes by having us peel our own appl es to make homemade applesauce, make our own b read, and mix up healthy plants from our class room garden in the spring. We will also create our own cookbooks to be printed and shared wi th families. \r\nStudents will gain math and l iterature skills as well as a life long enjoym ent for healthy cooking.nannan
==========
====

In [0]:

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

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

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

In [0]:

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

```
\"A person is a person, no matter how small.\"
 (Dr.Seuss) I teach the smallest students with
 the biggest enthusiasm for learning. My stude
nts learn in many different ways using all of
our senses and multiple intelligences. I use a
 wide range of techniques to help all my stude
nts succeed. \r\nStudents in my class come fro
m a variety of different backgrounds which mak
es for wonderful sharing of experiences and cu
ltures, including Native Americans.\r\nOur sch
ool is a caring community of successful learne
```

```
rs which can be seen through collaborative stu
dent project based learning in and out of the
classroom. Kindergarteners in my class love to
 work with hands-on materials and have many di
fferent opportunities to practice a skill befo
re it is mastered. Having the social skills to
 work cooperatively with friends is a crucial
aspect of the kindergarten curriculum.Montana
is the perfect place to learn about agricultur
e and nutrition. My students love to role play
 in our pretend kitchen in the early childhood
 classroom. I have had several kids ask me, \"
Can we try cooking with REAL food?\" I will ta
ke their idea and create \"Common Core Cooking
 Lessons\" where we learn important math and w
riting concepts while cooking delicious health
y food for snack time. My students will have a
 grounded appreciation for the work that went
into making the food and knowledge of where th
e ingredients came from as well as how it is h
ealthy for their bodies. This project would ex
pand our learning of nutrition and agricultura
l cooking recipes by having us peel our own ap
ples to make homemade applesauce, make our own
 bread, and mix up healthy plants from our cla
ssroom garden in the spring. We will also crea
te our own cookbooks to be printed and shared
with families. \r\nStudents will gain math and
 literature skills as well as a life long enjo
yment for healthy cooking.nannan
==========================================
====
```

In [0]:

```
# \r \n \t remove from string python: http://texthandler.com/
info/remove-line-breaks-python/
```

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

```
 A person is a person, no matter how small.  (
Dr.Seuss) I teach the smallest students with t
he biggest enthusiasm for learning. My student
s learn in many different ways using all of ou
r senses and multiple intelligences. I use a w
ide range of techniques to help all my student
s succeed.   Students in my class come from a
variety of different backgrounds which makes f
or wonderful sharing of experiences and cultur
es, including Native Americans.  Our school is
 a caring community of successful learners whi
ch can be seen through collaborative student p
roject based learning in and out of the classr
oom. Kindergarteners in my class love to work
with hands-on materials and have many differen
t opportunities to practice a skill before it
is mastered. Having the social skills to work
cooperatively with friends is a crucial aspect
 of the kindergarten curriculum.Montana is the
 perfect place to learn about agriculture and
nutrition. My students love to role play in ou
r pretend kitchen in the early childhood class
room. I have had several kids ask me,  Can we
try cooking with REAL food?  I will take their
 idea and create  Common Core Cooking Lessons
 where we learn important math and writing con
cepts while cooking delicious healthy food for
 snack time. My students will have a grounded
appreciation for the work that went into makin
g the food and knowledge of where the ingredie
nts came from as well as how it is healthy for
 their bodies. This project would expand our l
```

```
earning of nutrition and agricultural cooking
recipes by having us peel our own apples to ma
ke homemade applesauce, make our own bread, an
d mix up healthy plants from our classroom gar
den in the spring. We will also create our own
 cookbooks to be printed and shared with famil
ies.   Students will gain math and literature
skills as well as a life long enjoyment for he
althy cooking.nannan
```

In [0]:

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

```
 A person is a person no matter how small Dr S
euss I teach the smallest students with the bi
ggest enthusiasm for learning My students lear
n in many different ways using all of our sens
es and multiple intelligences I use a wide ran
ge of techniques to help all my students succe
ed Students in my class come from a variety of
 different backgrounds which makes for wonderf
ul sharing of experiences and cultures includi
ng Native Americans Our school is a caring com
munity of successful learners which can be see
n through collaborative student project based
learning in and out of the classroom Kindergar
teners in my class love to work with hands on
materials and have many different opportunitie
s to practice a skill before it is mastered Ha
ving the social skills to work cooperatively w
ith friends is a crucial aspect of the kinderg
arten curriculum Montana is the perfect place
to learn about agriculture and nutrition My st
udents love to role play in our pretend kitche
```

n in the early childhood classroom I have had several kids ask me Can we try cooking with REAL food I will take their idea and create Common Core Cooking Lessons where we learn important math and writing concepts while cooking delicious healthy food for snack time My students will have a grounded appreciation for the work that went into making the food and knowledge of where the ingredients came from as well as how it is healthy for their bodies This project would expand our learning of nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce make our own bread and mix up healthy plants from our classroom garden in the spring We will also create our own cookbooks to be printed and shared with families Students will gain math and literature skills as well as a life long enjoyment for healthy cooking nannan

In [0]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
```

```
            'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in
', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where',
 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', '
same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't",
'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "co
uldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", '
isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't",
'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
 \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:

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

```
100%|██████████| 109248/109248 [01:06<00:00, 1
641.48it/s]
```

In [0]:

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

Out[0]:

```
'person person no matter small dr seuss teach
smallest students biggest enthusiasm learning
students learn many different ways using sense
s multiple intelligences use wide range techni
ques help students succeed students class come
 variety different backgrounds makes wonderful
 sharing experiences cultures including native
 americans school caring community successful
learners seen collaborative student project ba
sed learning classroom kindergarteners class l
ove work hands materials many different opport
unities practice skill mastered social skills
work cooperatively friends crucial aspect kind
ergarten curriculum montana perfect place lear
n agriculture nutrition students love role pla
y pretend kitchen early childhood classroom se
veral kids ask try cooking real food take idea
 create common core cooking lessons learn impo
rtant math writing concepts cooking delicious
healthy food snack time students grounded appr
eciation work went making food knowledge ingre
dients came well healthy bodies project would
expand learning nutrition agricultural cooking
 recipes us peel apples make homemade applesau
ce make bread mix healthy plants classroom gar
den spring also create cookbooks printed share
d families students gain math literature skill
s well life long enjoyment healthy cooking nan
nan'
```

2.2 Preprocessing of project_title

In [0]:

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

```
Engineering STEAM into the Primary Classroom
==================================================
====
Building Blocks for Learning
==================================================
====
Empowering Students Through Art:Learning About
 Then and Now
==================================================
====
Health Nutritional Cooking in Kindergarten
==================================================
====
```

In [0]:

```
import re

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

    # general
```

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

In [0]:

```
sent = decontracted(project_data['project_title'].values[2000
0])
print(sent)
print("="*50)
```

```
Health Nutritional Cooking in Kindergarten
==========================================
====
```

In [0]:

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

```
Health Nutritional Cooking in Kindergarten
```

In [0]:

```
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Health Nutritional Cooking in Kindergarten
```

In [0]:

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

```
100%|██████████| 109248/109248 [00:03<00:00, 3
5178.71it/s]
```

In [0]:

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

Out[0]:

```
'health nutritional cooking kindergarten'
```

In [0]:

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

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	

1

37728 p043609 3f60494c61921b3b43ab61bdde2904df Ms.

In [0]:

```
print(project_data.shape)
print(project_data.columns)
```

```
(109248, 20)
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
her_prefix', 'school_state',
       'Date', 'project_grade_category', 'proj
ect_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', '
project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'price', 'quantity', 'clean_categories'
, 'clean_subcategories',
       'essay'],
      dtype='object')
```

3.Splitting data into Train and cross validation(or test): Stratified Sampling

In [0]:

```
Y = project_data['project_is_approved'].values
X = project_data
```

In [0]:

```
project_data.shape
```

Out[0]:

```
(109248, 20)
```

In [0]:

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

# X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, shuffle=Flase)# this is for time series split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,stratify=Y) # this is random splitting
```

In [0]:

```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(73196, 20) (73196,)
(36052, 20) (36052,)
```

4.Vectorizing Text data

4.1 Essay

In [0]:

```
vectorizer = CountVectorizer(min_df=10,ngram_range=(2,2), max_features=5000)
vectorizer.fit(X_train['essay'].values) # fit has to happen o
```

```
nly on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_essay_bow = vectorizer.transform(X_train['essay'].val
ues)
X_test_essay_bow = vectorizer.transform(X_test['essay'].value
s)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_test_essay_bow.shape, y_test.shape)
```

```
After vectorizations
(73196, 5000) (73196,)
(36052, 5000) (36052,)
```

4.2 Project_title

In [0]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10, max_features=5000)
vectorizer.fit(X_train['project_title'].values) # fit has to
happen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_title_bow = vectorizer.transform(X_train['project_tit
le'].values)
X_test_title_bow = vectorizer.transform(X_test['project_title
'].values)

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_test_title_bow.shape, y_test.shape)
```

```
After vectorizations
(73196, 2636) (73196,)
(36052, 2636) (36052,)
```

4.3 Project_resource_summary

In [0]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10, max_features=5000)
vectorizer.fit(X_train['project_resource_summary'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_res_sum_bow = vectorizer.transform(X_train['project_resource_summary'].values)
X_test_res_sum_bow = vectorizer.transform(X_test['project_resource_summary'].values)

print("After vectorizations")
print(X_train_res_sum_bow.shape, y_train.shape)
print(X_test_res_sum_bow.shape, y_test.shape)
```

```
After vectorizations
(73196, 4861) (73196,)
(36052, 4861) (36052,)
```

5.Catogorical features:one hot encoding

5.1 Clean_categories

In [0]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data


# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_category_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_test_clean_category_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_clean_category_ohe.shape, y_train.shape)
print(X_test_clean_category_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(73196, 9) (73196,)
(36052, 9) (36052,)
['appliedlearning', 'care_hunger', 'health_spo
rts', 'history_civics', 'literacy_language', '
math_science', 'music_arts', 'specialneeds', '
warmth']
```

5.2 Clean_subcategories

In [0]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data
```

```
# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_clean_subcategory_ohe = vectorizer.transform(X_train[
'clean_subcategories'].values)
X_test_clean_subcategory_ohe = vectorizer.transform(X_test['c
lean_subcategories'].values)

print("After vectorizations")
print(X_train_clean_subcategory_ohe.shape, y_train.shape)
print(X_test_clean_subcategory_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(73196, 30) (73196,)
(36052, 30) (36052,)
['appliedsciences', 'care_hunger', 'charactere
ducation', 'civics_government', 'college_caree
rprep', 'communityservice', 'earlydevelopment'
, 'economics', 'environmentalscience', 'esl',
'extracurricular', 'financialliteracy', 'forei
gnlanguages', 'gym_fitness', 'health_lifescien
ce', 'health_wellness', 'history_geography', '
literacy', 'literature_writing', 'mathematics'
, 'music', 'nutritioneducation', 'other', 'par
entinvolvement', 'performingarts', 'socialscie
nces', 'specialneeds', 'teamsports', 'visualar
ts', 'warmth']
```

5.3 Teacher_prefix

In [0]:

```
X_train.teacher_prefix = X_train.teacher_prefix.fillna('')
X_train['teacher_prefix'].value_counts()
```

Out[0]:

```
Mrs.       38368
Ms.        26111
Mr.         7122
Teacher     1583
Dr.            9
                3
Name: teacher_prefix, dtype: int64
```

In [0]:

```
X_test.teacher_prefix = X_test.teacher_prefix.fillna('')
X_test['teacher_prefix'].value_counts()
```

Out[0]:

```
Mrs.       18901
Ms.        12844
Mr.         3526
Teacher      777
Dr.            4
Name: teacher_prefix, dtype: int64
```

In [0]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values) # fit has to
 happen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_p
refix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_pre
fix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
```

```
print(vectorizer.get_feature_names())
```

```
After vectorizations
(73196, 5) (73196,)
(36052, 5) (36052,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
```

5.4 School_state

In [0]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values) # fit has to h
appen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_state_ohe = vectorizer.transform(X_train['school_stat
e'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'
].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(73196, 51) (73196,)
(36052, 51) (36052,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc
', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', '
in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi',
 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh
', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', '
pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va',
```

```
 'vt', 'wa', 'wi', 'wv', 'wy']
```

5.5 Project_grade_category

In [0]:

```
X_train.project_grade_category = X_train.project_grade_categ
ory.str.replace('\s+', '_')
X_train.project_grade_category = X_train.project_grade_categ
ory.str.replace('-', '_')
X_train['project_grade_category'].value_counts()
```

Out[0]:

```
Grades_PreK_2    29661
Grades_3_5       24890
Grades_6_8       11329
Grades_9_12       7316
Name: project_grade_category, dtype: int64
```

In [0]:

```
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values) # fi
t has to happen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_grade_ohe = vectorizer.transform(X_train['project_gra
de_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade
_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(73196, 4) (73196,)
(36052, 4) (36052,)
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'G
rades_PreK_2']
```

6.Numerical features

6.1 Price

In [0]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['price'].values.reshape(-1,1))

X_train_price_std = standard_vec.transform(X_train['price'].v
alues.reshape(-1,1))
X_test_price_std = standard_vec.transform(X_test['price'].val
ues.reshape(-1,1))

print("After vectorizations")
print(X_train_price_std.shape, y_train.shape)
print(X_test_price_std.shape, y_test.shape)
```

```
After vectorizations
(73196, 1) (73196,)
(36052, 1) (36052,)
```

6.2 Teacher_number_of_previously_posted_projects

In [0]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['teacher_number_of_previously_posted
_projects'].values.reshape(-1,1))

X_train_prev_projects_std = standard_vec.transform(X_train['t
eacher_number_of_previously_posted_projects'].values.reshape(-
1,1))
X_test_prev_projects_std = standard_vec.transform(X_test['tea
cher_number_of_previously_posted_projects'].values.reshape(-1,
1))

print("After vectorizations")
print(X_train_prev_projects_std.shape, y_train.shape)
print(X_test_prev_projects_std.shape, y_test.shape)
```

```
After vectorizations
(73196, 1) (73196,)
(36052, 1) (36052,)
```

6.3 Quantity

In [0]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['quantity'].values.reshape(-1,1))

X_train_quantity_std = standard_vec.transform(X_train['quanti
ty'].values.reshape(-1,1))
X_test_quantity_std = standard_vec.transform(X_test['quantity
'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_quantity_std.shape, y_train.shape)
print(X_test_quantity_std.shape, y_test.shape)
```

```
After vectorizations
(73196, 1) (73196,)
(36052, 1) (36052,)
```

7.Apply Support Vector Machines(SGDClassifier with hinge loss: Linear SVM)

7.1 Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

Merging all the above features

In [0]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_train_bow = hstack((X_train_essay_bow,X_train_title_bow,X_train_res_sum_bow,X_train_clean_category_ohe,X_train_clean_subcategory_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_std,X_train_prev_projects_std,X_train_quantity_std)).tocsr()
X_test_bow = hstack((X_test_essay_bow,X_test_title_bow,X_test_res_sum_bow,X_test_clean_category_ohe,X_test_clean_subcategory_ohe, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_price_std,X_test_prev_projects_std,X_test_quantity_std)).tocsr()

print("Final Data matrix")
print(X_train_bow.shape, y_train.shape)
print(X_test_bow.shape, y_test.shape)
```

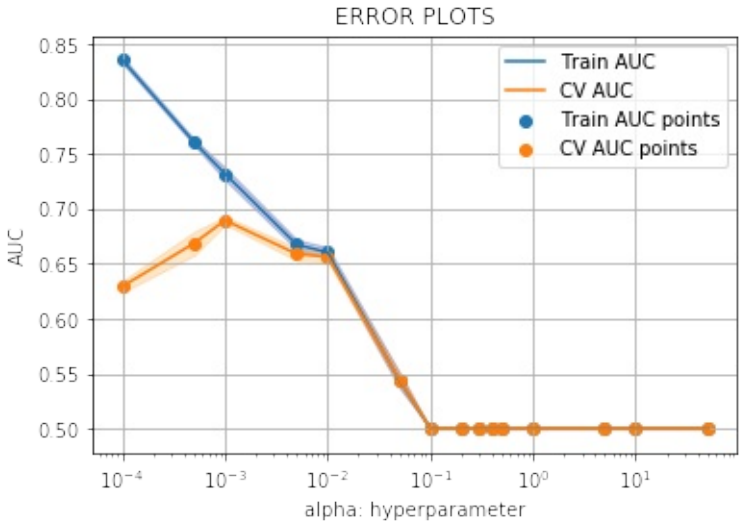
```
Final Data matrix
```

```
(73196, 12621) (73196,)
(36052, 12621) (36052,)
```

In [0]:

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

SVM_BOW = SGDClassifier(class_weight='balanced',loss='hinge',
 penalty='l1')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0
.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_BOW, parameters, cv=3, scoring='roc_au
c',return_train_score=True)
clf.fit(X_train_bow, 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(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC
points')
```

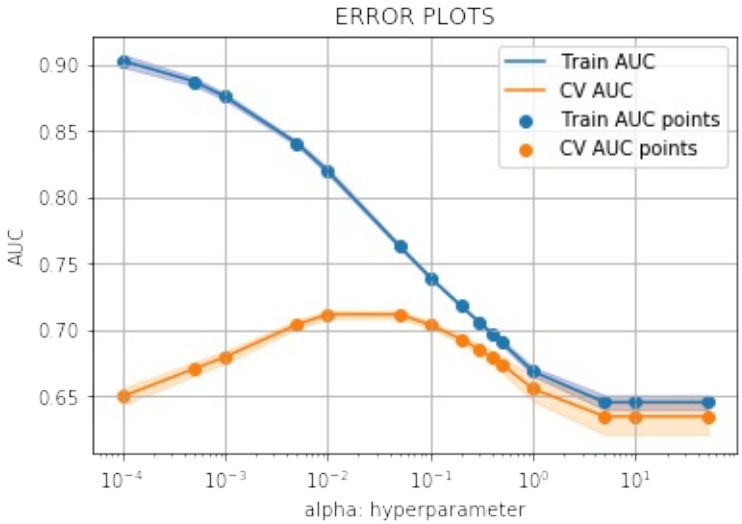
```
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points
')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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

SVM_BOW = SGDClassifier(class_weight='balanced',loss='hinge',
 penalty='l2')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0
.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_BOW, parameters, cv=3, scoring='roc_au
c',return_train_score=True)
```

```
clf.fit(X_train_bow, 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(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC
points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points
')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

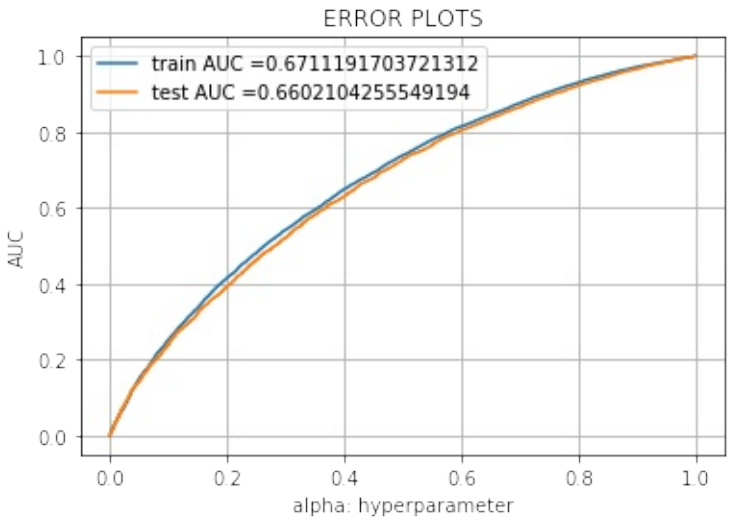


when penalty is L2 regularization,the plot is smooth to choose the best alpha as compared to L1

In [0]:

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


SVM_BOW = SGDClassifier(class_weight='balanced',loss='hinge',
 penalty='l2',alpha=0.01)
SVM_BOW.fit(X_train_bow, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


#https://datascience.stackexchange.com/questions/18374/predicting-probability-from-scikit-learn-svc-decision-function-with-
#decision-fun/18375#18375
```

```
y_train_pred = SVM_BOW.decision_function(X_train_bow)
y_test_pred = SVM_BOW.decision_function(X_test_bow)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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

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

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

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

In [0]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999796
47145 for threshold -0.282
[[ 5542  5541]
 [11785 50328]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161
092995 for threshold 0.15
[[ 4085  1374]
 [13364 17229]]
```

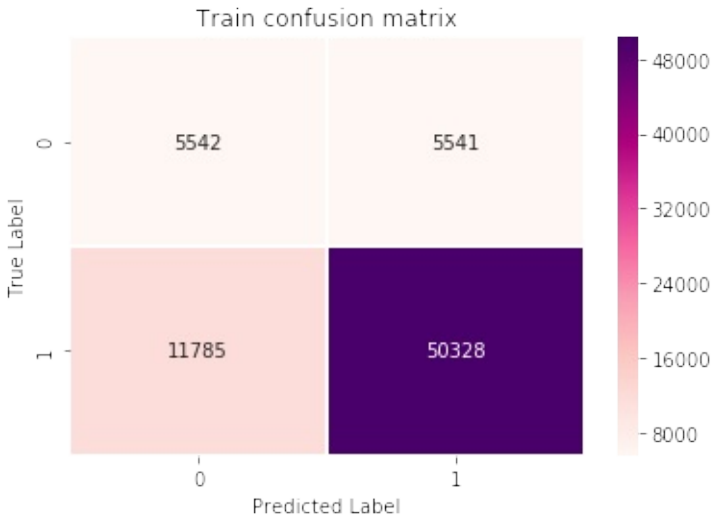
In [0]:

```
frame_confusion_train = pd.DataFrame(confusion_matrix(y_train
,predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)),
range(2),range(2))
frame_confusion_test = pd.DataFrame(confusion_matrix(y_test,p
redict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), rang
e(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999796
47145 for threshold -0.282
the maximum value of tpr*(1-fpr) 0.24999999161
092995 for threshold 0.15
```

In [0]:

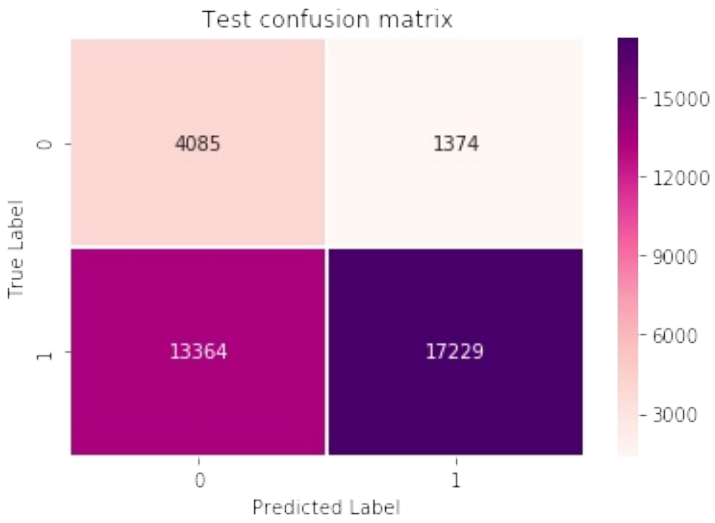
```
sns.heatmap(frame_confusion_train, annot = True, fmt="d", cma
p="RdPu", linewidths=.5)
plt.title("Train confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



In [0]:

```
sns.heatmap(frame_confusion_test, annot = True, fmt="d", cmap
="RdPu", linewidths=.5)
```

```
plt.title("Test confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



SET 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)

In [0]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['essay'].values) # fit has to happen o
nly on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].v
alues)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].val
ues)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
```

```
print(X_test_essay_tfidf.shape, y_test.shape)
```

```
After vectorizations
(16750, 8058) (16750,)
(8250, 8058) (8250,)
```

In [0]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['project_title'].values) # fit has to
happen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_title_tfidf = vectorizer.transform(X_train['project_t
itle'].values)
X_test_title_tfidf = vectorizer.transform(X_test['project_tit
le'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_test_title_tfidf.shape, y_test.shape)
```

```
After vectorizations
(73196, 2634) (73196,)
(36052, 2634) (36052,)
```

In [0]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['project_resource_summary'].values) #
fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to ve
ctor
X_train_res_sum_tfidf = vectorizer.transform(X_train['project
_resource_summary'].values)
X_test_res_sum_tfidf = vectorizer.transform(X_test['project_r
```

```
esource_summary'].values)

print("After vectorizations")
print(X_train_res_sum_tfidf.shape, y_train.shape)
print(X_test_res_sum_tfidf.shape, y_test.shape)
```

```
After vectorizations
(73196, 4885) (73196,)
(36052, 4885) (36052,)
```

In [0]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_train_tfidf = hstack((X_train_essay_tfidf,X_train_title_tfidf,X_train_res_sum_tfidf,X_train_clean_category_ohe,X_train_clean_subcategory_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_std,X_train_prev_projects_std,X_train_quantity_std)).tocsr()
X_test_tfidf = hstack((X_test_essay_tfidf,X_test_title_tfidf, X_test_res_sum_tfidf,X_test_clean_category_ohe,X_test_clean_subcategory_ohe, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_price_std,X_test_prev_projects_std,X_test_quantity_std)).tocsr()

print("Final Data matrix")
print(X_train_tfidf.shape, y_train.shape)
print(X_test_tfidf.shape, y_test.shape)
```

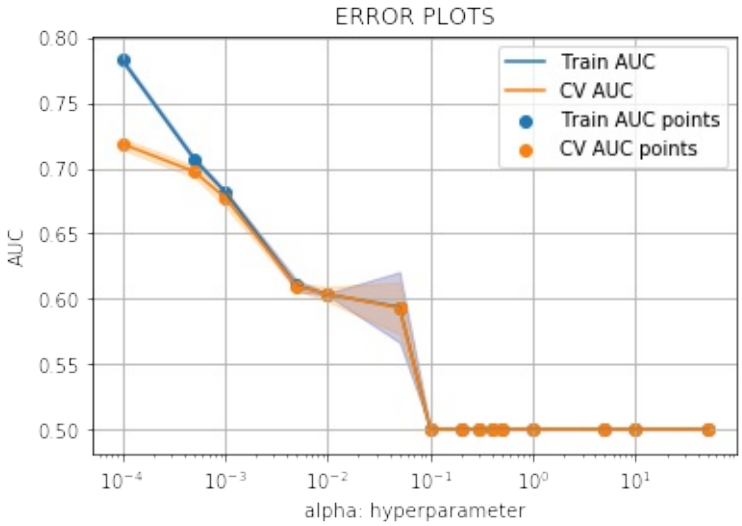
```
Final Data matrix
(73196, 22365) (73196,)
(36052, 22365) (36052,)
```

In [0]:

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

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier

SVM_TFIDF = SGDClassifier(class_weight='balanced',loss='hinge
', penalty='l1')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0
.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_TFIDF, parameters, cv=3, scoring='roc_
auc',return_train_score=True)
clf.fit(X_train_tfidf, 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(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC
points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points
')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
```

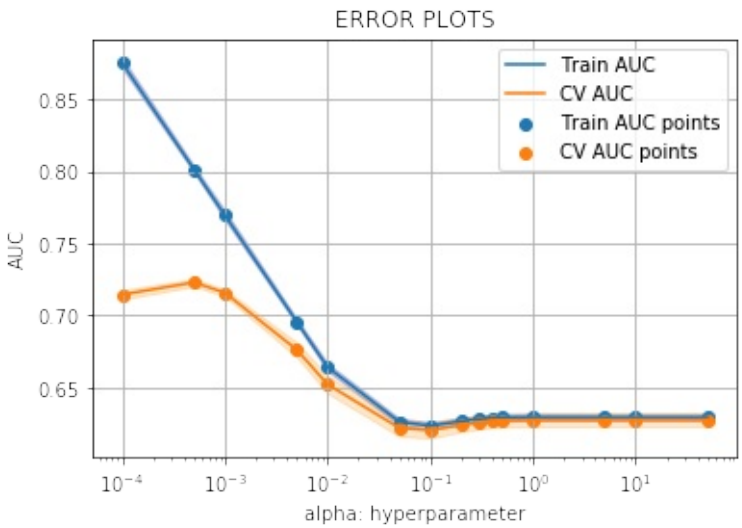
```
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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

SVM_TFIDF = SGDClassifier(class_weight='balanced',loss='hinge', penalty='l2')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_TFIDF, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_tfidf, 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']
```

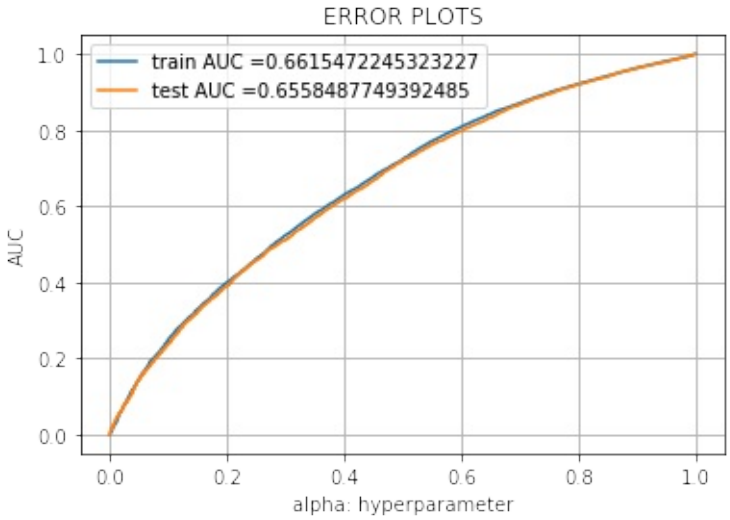
```
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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


SVM_TFIDF = SGDClassifier(class_weight='balanced',loss='hinge', penalty='l2',alpha=0.01)
SVM_TFIDF.fit(X_train_tfidf, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


#https://datascience.stackexchange.com/questions/18374/predicting-probability-from-scikit-learn-svc-decision-function-with-
#decision-fun/18375#18375
y_train_pred = SVM_TFIDF.decision_function(X_train_tfidf)
y_test_pred = SVM_TFIDF.decision_function(X_test_tfidf)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
```

```
plt.show()
```



In [0]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999796
47145 for threshold -0.149
[[ 5542  5541]
 [17202 44911]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161
092998 for threshold 0.309
[[ 4249  1210]
 [17647 12946]]
```

In [0]:

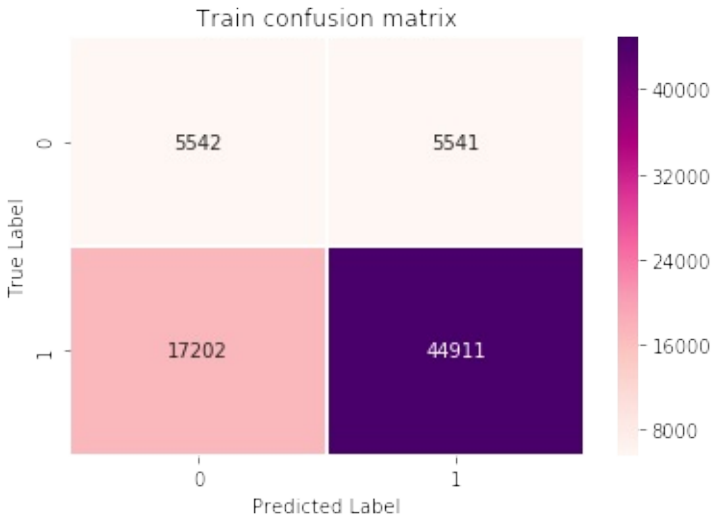
```
frame_confusion_train = pd.DataFrame(confusion_matrix(y_train
```

```
,predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)),
range(2),range(2))
frame_confusion_test = pd.DataFrame(confusion_matrix(y_test,p
redict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), rang
e(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999796
47145 for threshold -0.149
the maximum value of tpr*(1-fpr) 0.24999999161
092998 for threshold 0.309
```

In [0]:

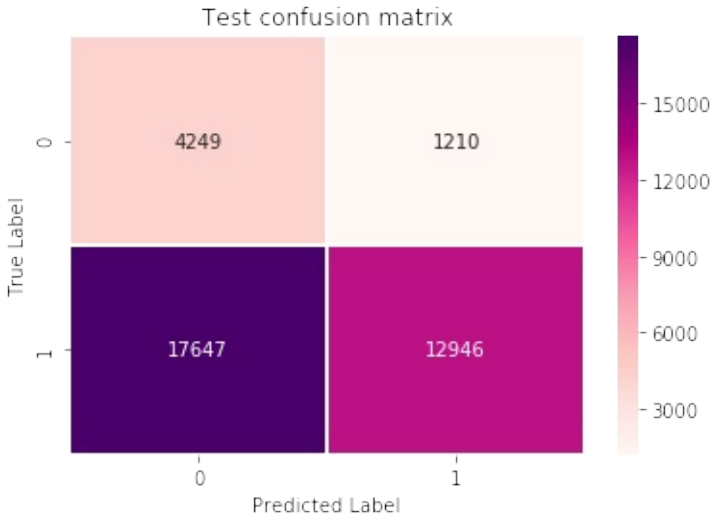
```
sns.heatmap(frame_confusion_train, annot = True, fmt="d", cma
p="RdPu", linewidths=.5)
plt.title("Train confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



In [0]:

```
sns.heatmap(frame_confusion_test, annot = True, fmt="d", cmap
="RdPu", linewidths=.5)
plt.title("Test confusion matrix")
plt.xlabel("Predicted Label")
```

```
plt.ylabel("True Label")
plt.show()
```



Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)

In [0]:

```
!unzip glove*.zip
```

```
Archive:  glove.42B.300d.zip
  inflating: glove.42B.300d.txt
```

In [0]:

```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
import numpy as np
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open('glove.42B.300d.txt',encoding="utf8")
    model = {}
    for line in tqdm(f):
```

```
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine
[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')
```

```
Loading Glove Model
```

```
1917495it [04:05, 7818.98it/s]
```

```
Done. 1917495  words loaded!
```

In [0]:

```
words = []
for i in project_data['essay']:
    words.extend(i.split(' '))

for i in project_data['project_title']:
    words.extend(i.split(' '))
print("all the words in the corpus", len(words))
words = set(words)
print("the unique words in the corpus", len(words))

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

```
all the words in the corpus 28930375
the unique words in the corpus 360322
The number of words that are present in both g
love vectors and our corpus 44145 ( 12.252 %)
```

In [0]:

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

```
word 2 vec length 44145
```

In [0]:

```
# stronging variables into pickle files python: http://www.je
ssicayung.com/how-to-use-pickle-to-save-and-load-variables-in
-python/
import pickle

with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [0]:

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

In [0]:

```
# average Word2V# compute average word2vec for each review.
X_train_essay_avg_w2v_vectors = []; # the avg-w2v for each se
ntence/review is stored in this list
for sentence in tqdm(X_train['essay']): # for each review/sen
tence
```

```
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    X_train_essay_avg_w2v_vectors.append(vector)
print("X_train")
print(len(X_train_essay_avg_w2v_vectors))
print(len(X_train_essay_avg_w2v_vectors[0]))


# average Word2V# compute average word2vec for each review.
X_test_essay_avg_w2v_vectors = []; # the avg-w2v for each sen
tence/review is stored in this list
for sentence in tqdm(X_test['essay']): # for each review/sent
ence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    X_test_essay_avg_w2v_vectors.append(vector)
print("X_test")
print(len(X_test_essay_avg_w2v_vectors))
print(len(X_test_essay_avg_w2v_vectors[0]))
```

```
100%|██████████| 73196/73196 [00:34<00:00, 214
1.57it/s]
  1%|          | 205/36052 [00:00<00:17, 2049.
05it/s]
```

```
X_train
73196
300
```

```
100%|██████████| 36052/36052 [00:16<00:00, 212
7.70it/s]
```

```
X_test
36052
300
```

In [0]:

```
X_train_essay_avg_w2v_vectors = np.array(X_train_essay_avg_w2
v_vectors)
X_test_essay_avg_w2v_vectors = np.array(X_test_essay_avg_w2v_
vectors)
```

In [0]:

```
# average Word2Vec
# compute average word2vec for each review.
X_train_avg_w2v_vectors = []; # the avg-w2v for each sentence
/review is stored in this list
for sentence in tqdm(X_train['project_title']): # for each re
view/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    X_train_avg_w2v_vectors.append(vector)
print("X_train")
print(len(X_train_avg_w2v_vectors))
print(len(X_train_avg_w2v_vectors[0]))

# average Word2Vec
# compute average word2vec for each review.
X_test_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['project_title']): # 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
    X_test_avg_w2v_vectors.append(vector)
print("X_test")
print(len(X_test_avg_w2v_vectors))
print(len(X_test_avg_w2v_vectors[0]))
```

```
100%|██████████| 73196/73196 [00:00<00:00, 112920.98it/s]
 32%|███       | 11372/36052 [00:00<00:00, 113693.00it/s]
```

```
X_train
73196
```

```
300
```

```
100%|██████████| 36052/36052 [00:00<00:00, 108
077.56it/s]
```

```
X_test
36052
300
```

In [0]:

```
X_train_avg_w2v_vectors = np.array(X_train_avg_w2v_vectors)
X_test_avg_w2v_vectors = np.array(X_test_avg_w2v_vectors)
```

In [0]:

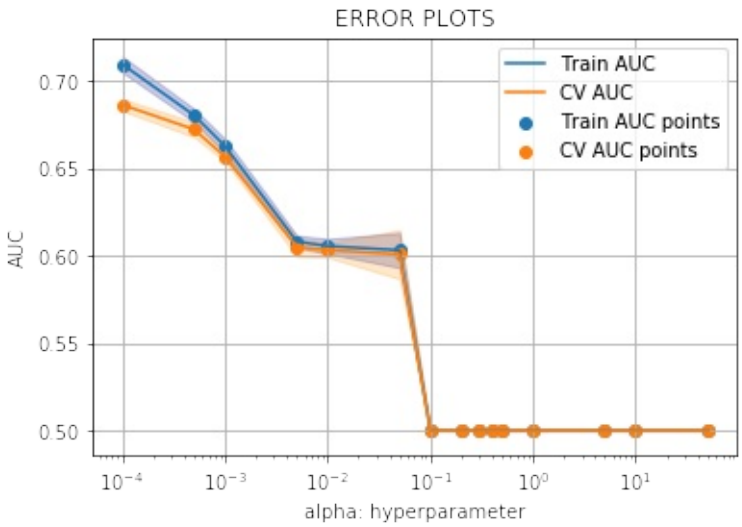
```
# merge two sparse matrices: https://stackoverflow.com/a/1971
0648/4084039
from scipy.sparse import hstack
X_train_w2v = hstack((X_train_essay_avg_w2v_vectors,X_train_a
vg_w2v_vectors,X_train_clean_category_ohe,X_train_clean_subca
tegory_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_g
rade_ohe, X_train_price_std,X_train_prev_projects_std,X_train
_quantity_std)).tocsr()
X_test_w2v = hstack((X_test_essay_avg_w2v_vectors,X_test_avg_
w2v_vectors,X_test_clean_category_ohe,X_test_clean_subcategor
y_ohe, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe
, X_test_price_std,X_test_prev_projects_std,X_test_quantity_s
td)).tocsr()

print("Final Data matrix")
print(X_train_w2v.shape, y_train.shape)
print(X_test_w2v.shape, y_test.shape)
```

```
Final Data matrix
(73196, 702) (73196,)
(36052, 702) (36052,)
```

In [0]:

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

SVM_W2V = SGDClassifier(class_weight='balanced',loss='hinge',
 penalty='l1')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0
.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_W2V, parameters, cv=3, scoring='roc_au
c',return_train_score=True)
clf.fit(X_train_w2v, 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(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC
points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points
')
```

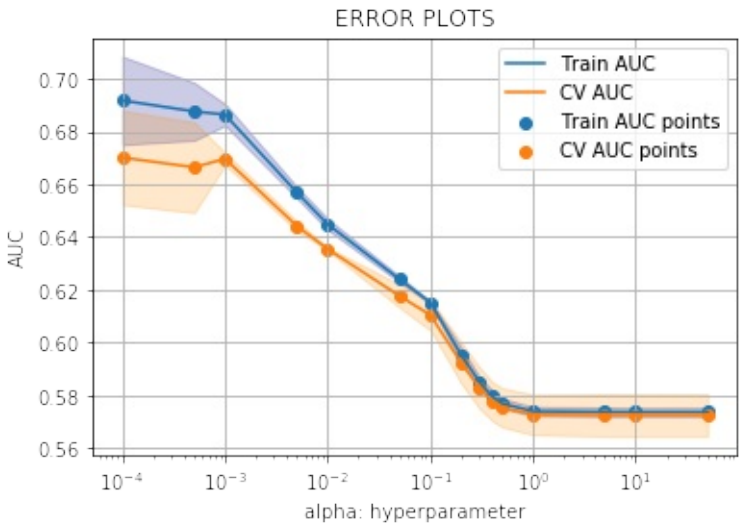
```
plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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

SVM_W2V = SGDClassifier(class_weight='balanced',loss='hinge',
 penalty='l2')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0
.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_W2V, parameters, cv=3, scoring='roc_au
c',return_train_score=True)
clf.fit(X_train_w2v, y_train)

train_auc= clf.cv_results_['mean_train_score']
```

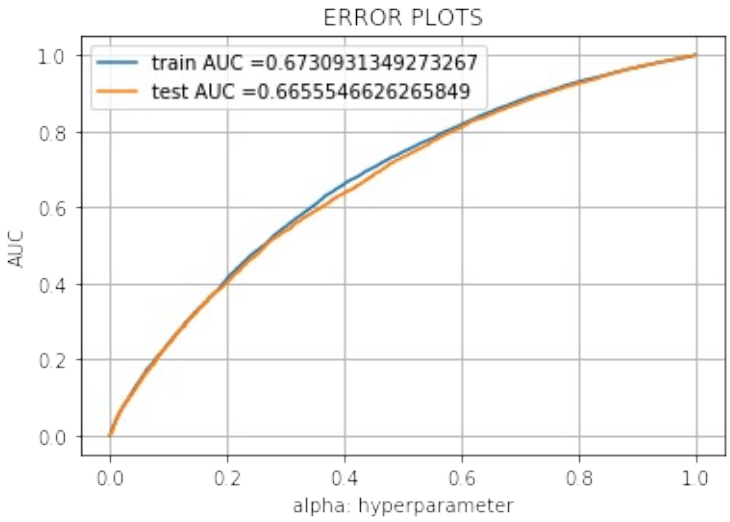
```
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC
points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points
')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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


SVM_W2V = SGDClassifier(class_weight='balanced',loss='hinge',
 penalty='l2',alpha=0.002)
SVM_W2V.fit(X_train_w2v, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


#https://datascience.stackexchange.com/questions/18374/predicting-probability-from-scikit-learn-svc-decision-function-with-
#decision-fun/18375#18375
y_train_pred = SVM_W2V.decision_function(X_train_w2v)
y_test_pred = SVM_W2V.decision_function(X_test_w2v)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
```

```
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Train confusion matrix
```

```
the maximum value of tpr*(1-fpr) 0.24999999796
47145 for threshold -0.097
[[ 5542  5541]
 [18532 43581]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161
092998 for threshold 0.349
[[ 4154  1305]
 [17556 13037]]
```

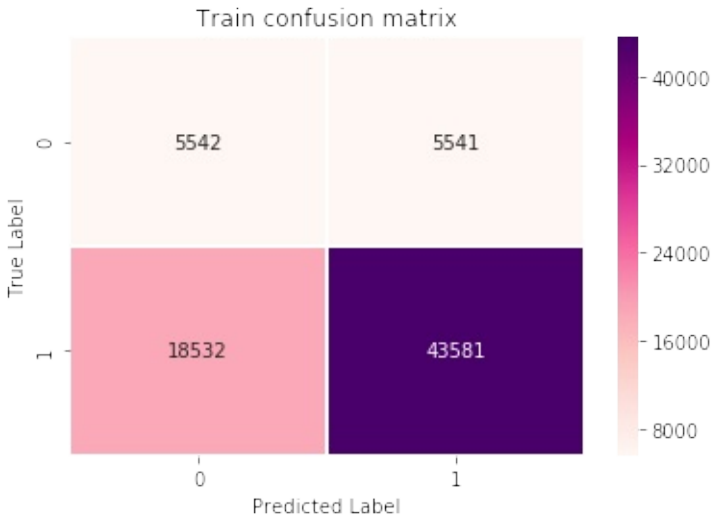
In [0]:

```
frame_confusion_train = pd.DataFrame(confusion_matrix(y_train
,predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)),
range(2),range(2))
frame_confusion_test = pd.DataFrame(confusion_matrix(y_test,p
redict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), rang
e(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999796
47145 for threshold -0.097
the maximum value of tpr*(1-fpr) 0.24999999161
092998 for threshold 0.349
```

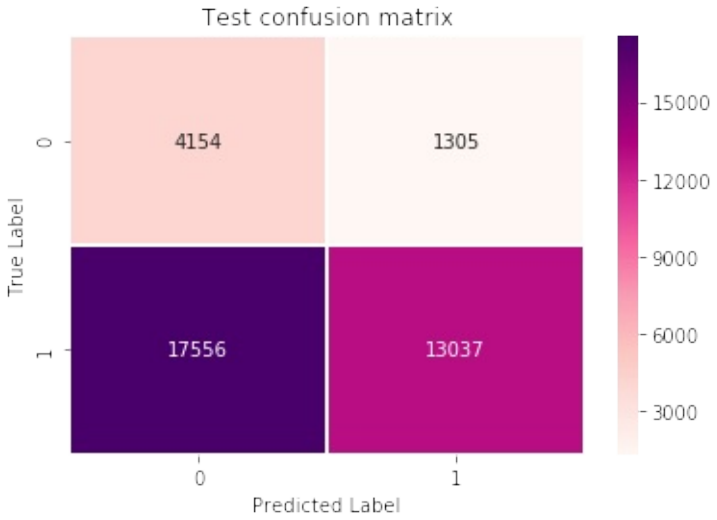
In [0]:

```
sns.heatmap(frame_confusion_train, annot = True, fmt="d", cma
p="RdPu", linewidths=.5)
plt.title("Train confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



In [0]:

```
sns.heatmap(frame_confusion_test, annot = True, fmt="d", cmap
="RdPu", linewidths=.5)
plt.title("Test confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



Set 4: categorical, numerical features +

project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

In [0]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the
idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [0]:

```
# average Word2Vec
# compute average word2vec for each review.
X_train_essay_tfidf_w2v_vectors = []; # the avg-w2v for each
sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # for each review/sen
tence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
```

```
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_essay_tfidf_w2v_vectors.append(vector)

print("X_train_essay_tfidf_w2v")
print(len(X_train_essay_tfidf_w2v_vectors))
print(len(X_train_essay_tfidf_w2v_vectors[0]))

# average Word2Vec
# compute average word2vec for each review.
X_test_essay_tfidf_w2v_vectors = []; # the avg-w2v for each s
entence/review is stored in this list
for sentence in tqdm(X_test['essay']): # for each review/sent
ence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_essay_tfidf_w2v_vectors.append(vector)
```

```
print("X_test_essay_tfidf_w2v")
print(len(X_test_essay_tfidf_w2v_vectors))
print(len(X_test_essay_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 73196/73196 [05:12<00:00, 233
.95it/s]
  0%|          | 26/36052 [00:00<02:19, 257.55
it/s]
```

```
X_train_essay_tfidf_w2v
73196
300
```

```
100%|██████████| 36052/36052 [02:31<00:00, 237
.51it/s]
```

```
X_test_essay_tfidf_w2v
36052
300
```

In [0]:

```
X_train_essay_tfidf_w2v_vectors = np.array(list(X_train_essay
_tfidf_w2v_vectors))
X_test_essay_tfidf_w2v_vectors = np.array(list(X_test_essay_t
fidf_w2v_vectors))
```

In [0]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the
idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [0]:

```
# average Word2Vec
# compute average word2vec for each review.
X_train_title_tfidf_w2v_vectors = []; # the avg-w2v for each
sentence/review is stored in this list
for sentence in tqdm(X_train['project_title']): # for each re
view/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_title_tfidf_w2v_vectors.append(vector)

print("X_train_title_tfidf_w2v")
print(len(X_train_title_tfidf_w2v_vectors))
print(len(X_train_title_tfidf_w2v_vectors[0]))

# average Word2Vec
# compute average word2vec for each review.
X_test_title_tfidf_w2v_vectors = []; # the avg-w2v for each s
entence/review is stored in this list
```

```
for sentence in tqdm(X_test['project_title']): # for each rev
iew/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_title_tfidf_w2v_vectors.append(vector)

print("X_test_title_tfidf_w2v")
print(len(X_test_title_tfidf_w2v_vectors))
print(len(X_test_title_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 73196/73196 [00:00<00:00, 787
47.73it/s]
 22%|██        | 7948/36052 [00:00<00:00, 7947
0.22it/s]
```

```
X_train_title_tfidf_w2v
73196
300
```

```
100%|██████████| 36052/36052 [00:00<00:00, 768
76.41it/s]
```

```
X_test_title_tfidf_w2v
36052
300
```

In [0]:

```
X_train_title_tfidf_w2v_vectors = np.array(list(X_train_title
_tfidf_w2v_vectors))
X_test_title_tfidf_w2v_vectors = np.array(list(X_test_title_t
fidf_w2v_vectors))
```

In [0]:

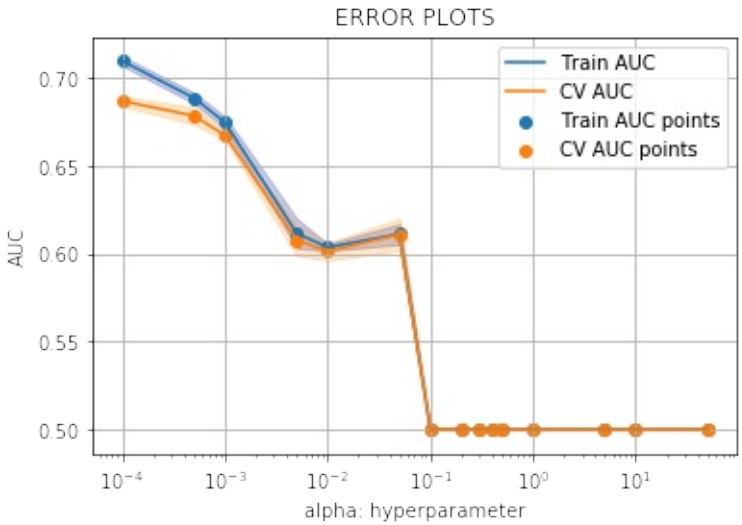
```
# merge two sparse matrices: https://stackoverflow.com/a/1971
0648/4084039
from scipy.sparse import hstack
X_train_tfidf_w2v = hstack((X_train_essay_tfidf_w2v_vectors,X
_train_title_tfidf_w2v_vectors,X_train_clean_category_ohe,X_t
rain_clean_subcategory_ohe, X_train_state_ohe, X_train_teache
r_ohe, X_train_grade_ohe, X_train_price_std,X_train_prev_proj
ects_std,X_train_quantity_std)).tocsr()
X_test_tfidf_w2v = hstack((X_test_essay_tfidf_w2v_vectors,X_t
est_title_tfidf_w2v_vectors,X_test_clean_category_ohe,X_test_
clean_subcategory_ohe, X_test_state_ohe, X_test_teacher_ohe,
X_test_grade_ohe, X_test_price_std,X_test_prev_projects_std,X
_test_quantity_std)).tocsr()

print("Final Data matrix")
print(X_train_tfidf_w2v.shape, y_train.shape)
print(X_test_tfidf_w2v.shape, y_test.shape)
```

```
Final Data matrix
(73196, 702) (73196,)
(36052, 702) (36052,)
```

In [0]:

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

SVM_TFIDF_W2V = SGDClassifier(class_weight='balanced',loss='hinge', penalty='l1')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_TFIDF_W2V, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_tfidf_w2v, 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(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')


plt.legend()
```

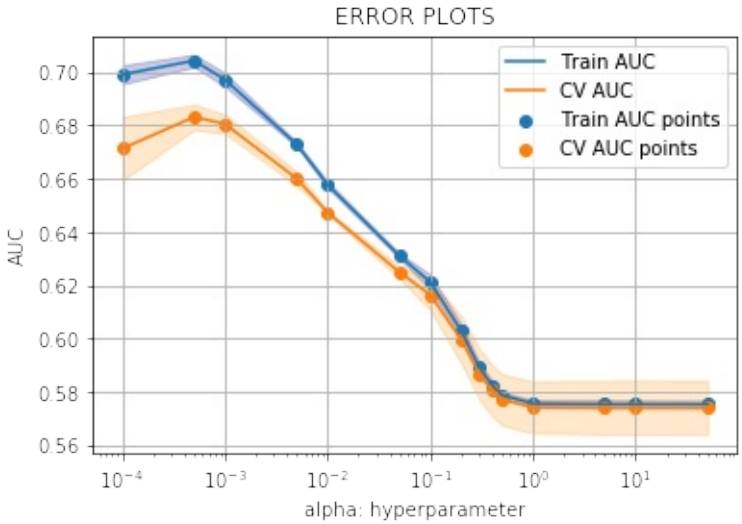
```
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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

SVM_TFIDF_W2V = SGDClassifier(class_weight='balanced',loss='hinge', penalty='l2')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_TFIDF_W2V, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_tfidf_w2v, 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']
```

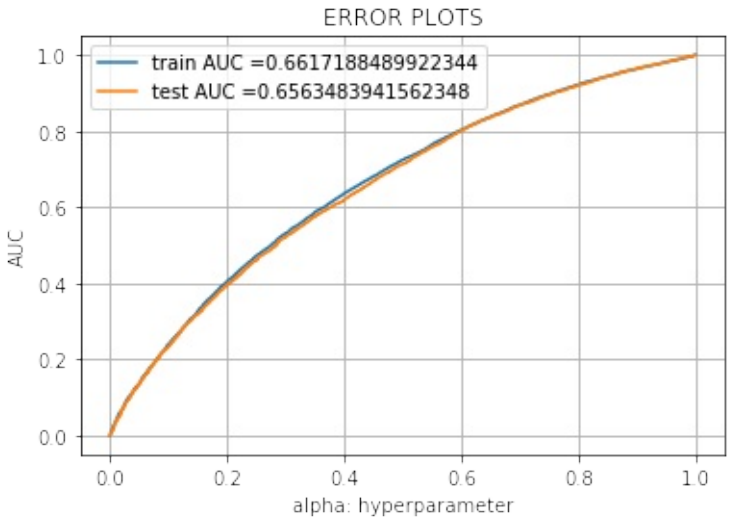
```
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC
points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points
')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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


SVM_TFIDF_W2V = SGDClassifier(class_weight='balanced',loss='hinge', penalty='l2',alpha=0.008)
SVM_TFIDF_W2V.fit(X_train_tfidf_w2v, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


#https://datascience.stackexchange.com/questions/18374/predicting-probability-from-scikit-learn-svc-decision-function-with-
#decision-fun/18375#18375
y_train_pred = SVM_TFIDF_W2V.decision_function(X_train_tfidf_w2v)
y_test_pred = SVM_TFIDF_W2V.decision_function(X_test_tfidf_w2v)
```

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999796
5 for threshold 1.127
[[ 5542  5541]
 [16437 45676]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161
1 for threshold 1.213
[[ 5155   304]
 [26188  4405]]
```

In [0]:

```
frame_confusion_train = pd.DataFrame(confusion_matrix(y_train
, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
,range(2),range(2))
frame_confusion_test = pd.DataFrame(confusion_matrix(y_test,p
redict(y_test_pred, tr_thresholds, test_fpr, test_fpr)),range
(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999796
5 for threshold 1.127
the maximum value of tpr*(1-fpr) 0.24999999161
1 for threshold 1.213
```

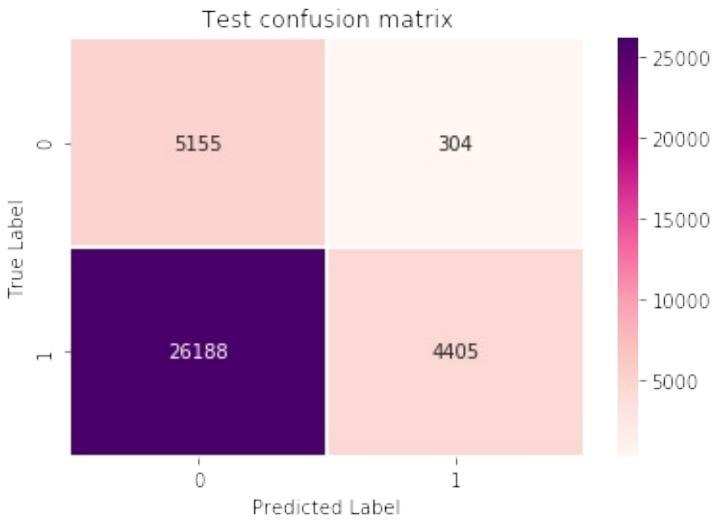
In [0]:

```
sns.heatmap(frame_confusion_train, annot = True, fmt="d", cma
p="RdPu", linewidths=.5)
plt.title("Train confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

In [0]:

```
sns.heatmap(frame_confusion_test, annot = True, fmt="d", cmap
="RdPu", linewidths=.5)
```

```
plt.title("Test confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



Set 5: categorical + numerical features

In [0]:

```
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
S=X_train['essay']

compound_tr = []
positive_tr = []
neutral_tr = []
negative_tr = []
for x in S:
    x_comp = sid.polarity_scores(x)["compound"]
    x_pos = sid.polarity_scores(x)["pos"]
```

```
    x_neu = sid.polarity_scores(x)["neu"]
    x_neg = sid.polarity_scores(x)["neg"]
    compound_tr.append(x_comp)
    positive_tr.append(x_pos)
    neutral_tr.append(x_neu)
    negative_tr.append(x_neg)

# we can use these 4 things as features/attributes (neg, neu,
 pos, compound)
```

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

In [0]:

```
X_train["compound"] = compound_tr
X_train["positive"] = positive_tr
X_train["neutral"] = neutral_tr
X_train["negative"] = negative_tr
```

In [0]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
X=X_test['essay']

compound = []
positive = []
neutral = []
negative = []
for x in X:
    x_comp = sid.polarity_scores(x)["compound"]
    x_pos = sid.polarity_scores(x)["pos"]
    x_neu = sid.polarity_scores(x)["neu"]
    x_neg = sid.polarity_scores(x)["neg"]
    compound.append(x_comp)
```

```
    positive.append(x_pos)
    neutral.append(x_neu)
    negative.append(x_neg)
```

In [0]:

```
X_test["compound"] = compound
X_test["positive"] = positive
X_test["neutral"] = neutral
X_test["negative"] = negative
```

In [0]:

```
X_train["essay_words"] = X_train["essay"].str.split().apply(len)
X_test["essay_words"] = X_test["essay"].str.split().apply(len)
```

In [0]:

```
X_train["title_words"] = X_train["project_title"].str.split().apply(len)
X_test["title_words"] = X_test["project_title"].str.split().apply(len)
```

In [0]:

```
X_train.head(1)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix
41938	139806	p057496	916a3177a9ca035b3d9b5d9d4a7e70f8	Ms.

In [0]:

```
X_test.head(1)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	sch
38231	22544	p159937	9bae21a19ca2fdf33a815d5ec8344427	Ms.	

Apply Truncated SVD

Considering train_data with 25k points,as my laptop is 4GB RAM its crashing.

In [0]:

```
from sklearn.decomposition import TruncatedSVD

X = [100,500,1000,2000,3000,4000,5000,6000]
variance = []

for i in tqdm(X):
    svd = TruncatedSVD(n_components=i, n_iter=7, random_state
=42)
    svd.fit(X_train_essay_tfidf)
    variance.append(svd.explained_variance_ratio_.sum())
```

```
100%|██████████| 8/8 [43:31<00:00, 541.33s/it]
```

In [0]:

```
X = [100,500,1000,2000,3000,4000,5000,6000]
```

In [0]:

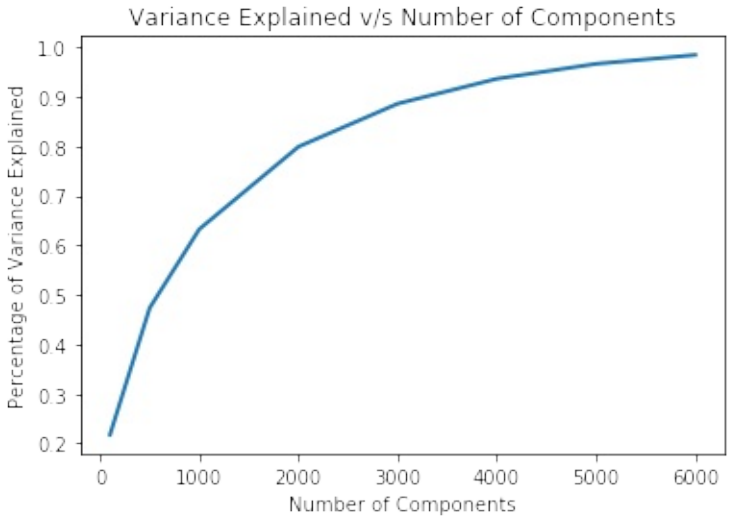
```
variance
```

Out[0]:

```
[0.2177337152645168,
 0.4736627052301479,
 0.6328486600923353,
 0.7994908485633121,
 0.8864460171653418,
 0.9367683256321565,
 0.9669391496797436,
 0.9849280731809974]
```

In [0]:

```
plt.xlabel("Number of Components")
plt.ylabel("Percentage of Variance Explained")
plt.title("Variance Explained v/s Number of Components")
plt.plot(X,variance,lw=2)
plt.show()
```



In [0]:

```
svd = TruncatedSVD(n_components=4000, n_iter=7, random_state=
42)
svd.fit(X_train_essay_tfidf)
svd_train = svd.transform(X_train_essay_tfidf)
```

In [0]:

```
svd_train.shape
```

Out[0]:

```
(16750, 4000)
```

In [0]:

```
svd_test = svd.transform(X_test_essay_tfidf)
```

In [0]:

```
svd_test.shape
```

Out[0]:

```
(8250, 4000)
```

In [0]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['essay_words'].values.reshape(-1,1))

X_train_essay_words = standard_vec.transform(X_train['essay_w
```

```
ords'].values.reshape(-1,1))
X_test_essay_words = standard_vec.transform(X_test['essay_wor
ds'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_essay_words.shape, y_train.shape)
print(X_test_essay_words.shape, y_test.shape)
```

```
After vectorizations
(16750, 1) (16750,)
(8250, 1) (8250,)
```

In [0]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['title_words'].values.reshape(-1,1))

X_train_title_words = standard_vec.transform(X_train['title_w
ords'].values.reshape(-1,1))
X_test_title_words = standard_vec.transform(X_test['title_wor
ds'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_title_words.shape, y_train.shape)
print(X_test_title_words.shape, y_test.shape)
```

```
After vectorizations
(16750, 1) (16750,)
(8250, 1) (8250,)
```

In [0]:

```

```

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['negative'].values.reshape(-1,1))

X_train_negative = standard_vec.transform(X_train['negative']
.values.reshape(-1,1))
X_test_negative = standard_vec.transform(X_test['negative'].v
alues.reshape(-1,1))

print("After vectorizations")
print(X_train_negative.shape, y_train.shape)
print(X_test_negative.shape, y_test.shape)
```

```
After vectorizations
(16750, 1) (16750,)
(8250, 1) (8250,)
```

In [0]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['positive'].values.reshape(-1,1))

X_train_positive = standard_vec.transform(X_train['positive']
.values.reshape(-1,1))
X_test_positive = standard_vec.transform(X_test['positive'].v
```

```
alues.reshape(-1,1))

print("After vectorizations")
print(X_train_positive.shape, y_train.shape)
print(X_test_positive.shape, y_test.shape)
```

```
After vectorizations
(16750, 1) (16750,)
(8250, 1) (8250,)
```

In [0]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['neutral'].values.reshape(-1,1))

X_train_neutral = standard_vec.transform(X_train['neutral'].v
alues.reshape(-1,1))
X_test_neutral = standard_vec.transform(X_test['neutral'].val
ues.reshape(-1,1))

print("After vectorizations")
print(X_train_neutral.shape, y_train.shape)
print(X_test_neutral.shape, y_test.shape)
```

```
After vectorizations
(16750, 1) (16750,)
(8250, 1) (8250,)
```

In [0]:

```
from sklearn.preprocessing import StandardScaler
```

```
standard_vec = StandardScaler(with_mean = False)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['compound'].values.reshape(-1,1))

X_train_compound = standard_vec.transform(X_train['compound']
.values.reshape(-1,1))
X_test_compound = standard_vec.transform(X_test['compound'].v
alues.reshape(-1,1))

print("After vectorizations")
print(X_train_compound.shape, y_train.shape)
print(X_test_compound.shape, y_test.shape)
```

```
After vectorizations
(16750, 1) (16750,)
(8250, 1) (8250,)
```

In [0]:

```
# merge two sparse matrices: https://stackoverflow.com/a/1971
0648/4084039
from scipy.sparse import hstack
X_train_set5 = hstack((svd_train,X_train_essay_words,X_train_
title_words,X_train_negative,X_train_positive,X_train_neutral,
X_train_compound,X_train_clean_category_ohe,X_train_clean_sub
category_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train
_grade_ohe, X_train_price_std,X_train_prev_projects_std,X_tra
in_quantity_std)).tocsr()

X_test_set5 = hstack((svd_test,X_test_essay_words,X_test_titl
e_words,X_test_negative,X_test_positive,X_test_neutral,X_test
_compound,X_test_clean_category_ohe,X_test_clean_subcategory_
ohe, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe,
```

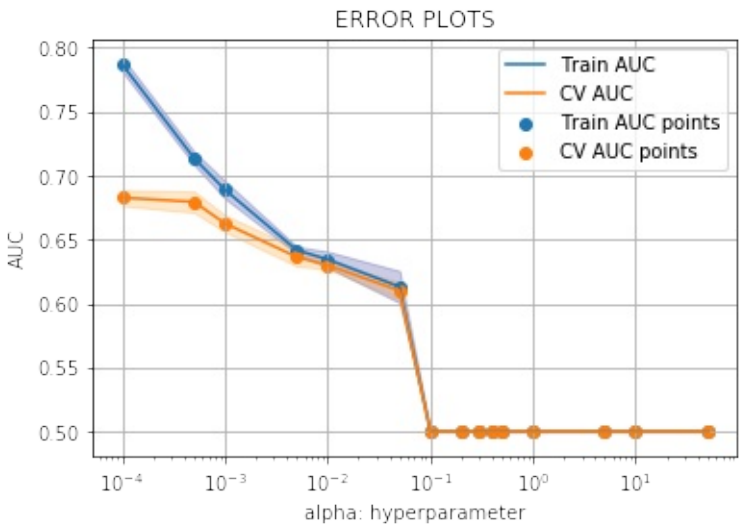
```
X_test_price_std,X_test_prev_projects_std,X_test_quantity_std
)).tocsr()

print("Final Data matrix")
print(X_train_set5.shape, y_train.shape)

print(X_test_set5.shape, y_test.shape)
```

```
Final Data matrix
(16750, 4108) (16750,)
(8250, 4108) (8250,)
```

In [0]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
odel_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier

SVM_SET5 = SGDClassifier(class_weight='balanced',loss='hinge'
, penalty='l1')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0
.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_SET5, parameters, cv=3, scoring='roc_a
uc',return_train_score=True)
clf.fit(X_train_set5, 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(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')
```

```
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

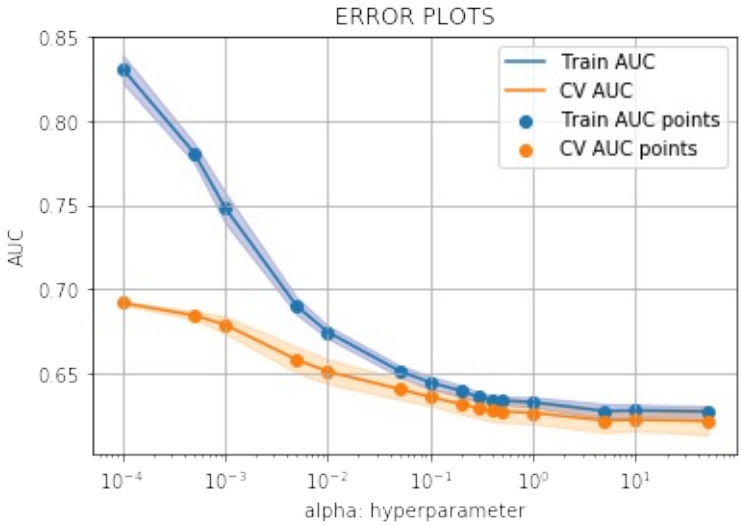


In [0]:

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

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier

SVM_SET5 = SGDClassifier(class_weight='balanced',loss='hinge'
, penalty='l2')
parameters = {'alpha':[0.0001,0.0005,0.001,0.005,0.01,0.05, 0
.1,0.2,0.3,0.4,0.5,1,5,10,50]}
clf = GridSearchCV(SVM_SET5, parameters, cv=3, scoring='roc_a
uc',return_train_score=True)
clf.fit(X_train_set5, 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(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_
auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_st
d,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC
points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points
')


plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
```

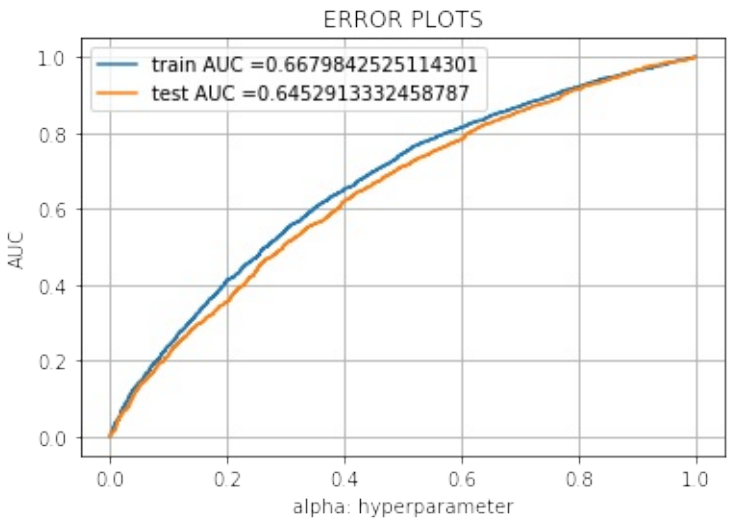
```
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

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


SVM_SET5 = SGDClassifier(class_weight='balanced',loss='hinge', penalty='l2',alpha=0.01)
SVM_SET5.fit(X_train_set5, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


#https://datascience.stackexchange.com/questions/18374/predicting-probability-from-scikit-learn-svc-decision-function-with-
#decision-fun/18375#18375
y_train_pred = SVM_SET5.decision_function(X_train_set5)
```

```
y_test_pred = SVM_SET5.decision_function(X_test_set5)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [0]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
print("Test confusion matrix")
```

```
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999986077
077296 for threshold -0.465
[[ 1341  1339]
 [ 3599 10471]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.027
[[ 973  347]
 [3733 3197]]
```

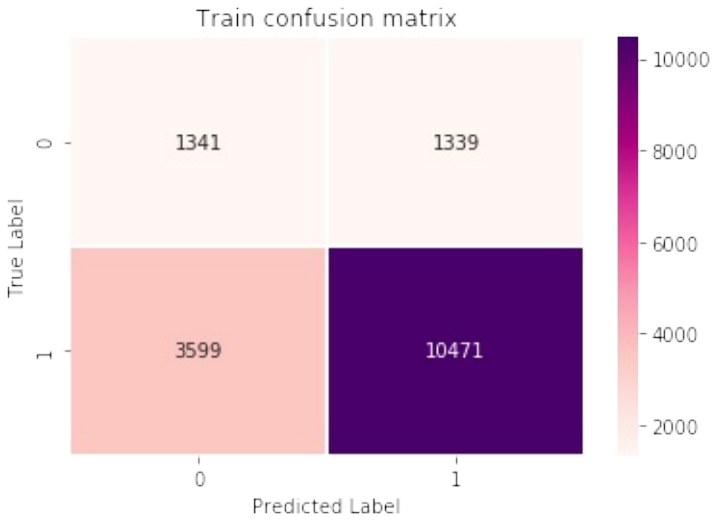
In [0]:

```
frame_confusion_train = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
frame_confusion_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999986077
077296 for threshold -0.465
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.027
```

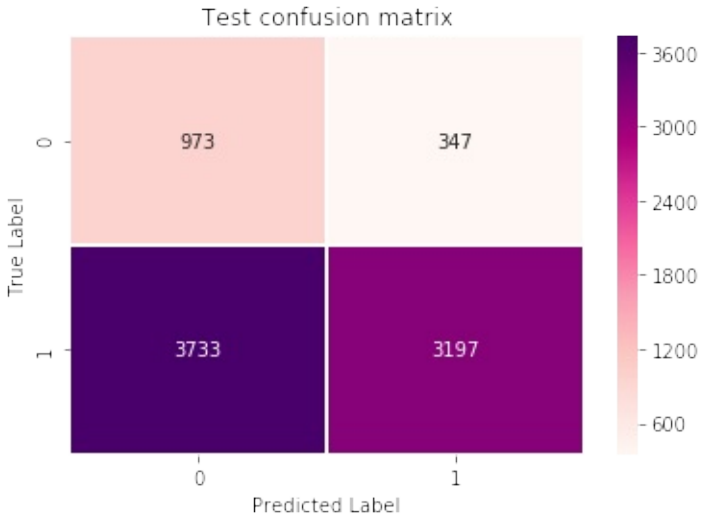
In [0]:

```
sns.heatmap(frame_confusion_train, annot = True, fmt="d", cmap="RdPu", linewidths=.5)
plt.title("Train confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



In [0]:

```
sns.heatmap(frame_confusion_test, annot = True, fmt="d", cmap
="RdPu", linewidths=.5)
plt.title("Test confusion matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



Summary

Pretty Table

In [1]:

```
#http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer","alpha:Hyperparameter","Penalty
","AUC"]
x.add_row(["BOW",0.01,"l2",0.6602])
x.add_row(["TFIDF",0.01,"l2",0.6558])
x.add_row(["W2V", 0.002,"l2", 0.6655])
x.add_row(["TFIDF_w2v", 0.008,"l2", 0.6563])
x.add_row(["NUM+CAT",0.01,"l2", 0.6452])
print(x)
```

```
+------------+----------------------+---------
+--------+
| Vectorizer | alpha:Hyperparameter | Penalty
|  AUC   |
+------------+----------------------+---------
+--------+
|    BOW     |         0.01         |    l2
| 0.6602 |
|   TFIDF    |         0.01         |    l2
| 0.6558 |
|    W2V     |        0.002         |    l2
| 0.6655 |
| TFIDF_w2v  |        0.008         |    l2
| 0.6563 |
|  NUM+CAT   |         0.01         |    l2
| 0.6452 |
+------------+----------------------+---------
+--------+
```

In [ ]: