Copy_of_Copy_of_4_DonorsChoose_NB

March 16, 2019

1 DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main p

How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly a
<li>How to increase the consistency of project vetting across different volunteers to improve the experience for teach
<li>How to focus volunteer time on the applications that need the most assistance</li>
</ul>

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

1.1 About the DonorsChoose Data Set

The train.csv data set provided by DonorsChoose contains the following features:

Feature	Description
project_id	A unique identifier for the proposed project. **Example:** p036502

project_title | Title of the project. **Examples:**
Art Will Make You Happy!
First Grade Fun
project_grade_category | Grade level of students for which the project is targeted. One of the following enumerated values:
Grades PreK-2
Grades 3-5
Grades 6-8
Grades 9-12
project_subject_categories | One or more (comma-separated) subject categories for the project from the following enumerated list of values:

Applied Learning
Care & Hunger
Health & Sports
History & Civics
Literacy & Language
Math & Science
Music & The Arts
Special Needs
Warmth
Examples:
Music & The Arts
Literacy & Language, Math & Science
school_state | State where school is located (Two-letter U.S. postal code). **Example:** WY
project_subject_subcategories | One or more (comma-separated) subject subcategories for the project. **Examples:**
Literacy
Literature & Writing, Social Sciences
project_resource_summary | An explanation of the resources needed for the project. **Example:** My students need hands on literacy materials to manage sensory needs!
project_essay_1 | First application essay
project_essay_2 | *Second application essay* project_essay_3 | Third application essay project_essay_4 | *Fourth application essay* project_submitted_datetime | Datetime when project application was submitted. **Example:** 2016-04-28 12:43:56.245
teacher_id | A unique identifier for the teacher of the proposed project. **Example:** bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix | Teacher's title. One of the following enumerated values:
nan
Dr.
Mr.
Mrs.
Ms.
Teacher.
teacher_number_of_previously_posted_projects | Number of project applications previously submitted by the same teacher. **Example:** 2
* See the section Notes on the Essay Data for more details about these features.
Additionally, the resources.csv data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
id	A project_id value from the train.csv file. **Example:** p036502
description	Desciption of the resource. **Example:** Tenor Saxophone Reeds, Box of 25

Feature	Description
quantity	Quantity of the resource required. **Example:** 3
price	Price of the resource required. **Example:** 9.95

Note: Many projects require multiple resources. The id value corresponds to a project_id in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
project_is_approved	A binary flag indicating whether DonorsChoose approved the project. A value of 0 indicates the project was not approved, and a value of 1 indicates the project was approved.

1.1.1 Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

project_essay_1: "Introduce us to your classroom"

project_essay_2: "Tell us more about your students"

project_essay_3: "Describe how your students will use the materials you're requesting"

project_essay_3: "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

project_essay_1: "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

project_essay_2: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

```
In [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
```

1.2 1.1 Reading Data

In [0]:
```
#since im using google_colab, i have to mount the gdrive folder for accessing the files

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/gdrive
```

In [0]:
```
#reading the datasets, i have taken only 5000 datapoints into consideration for avoiding mermory issues
```

```
        project_data = pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/Assignments_DonorsChoose_2
        resource_data = pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/Assignments_DonorsChoose_
```

```
In [0]: print("Number of data points in train data", project_data.shape)
        print('-'*50)
        print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

```
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  \
        473        100660   p234804   cbc0e38f522143b86d372f8b43d4cff3           Mrs.
        41558       33679   p137682   06f6e62e17de34fcf81020c77549e1d5           Mrs.

              school_state                 Date project_grade_category  \
        473             GA 2016-04-27 00:53:00          Grades PreK-2
        41558           WA 2016-04-27 01:05:25             Grades 3-5

              project_subject_categories project_subject_subcategories  \
        473             Applied Learning             Early Development
        41558        Literacy & Language                      Literacy

                                          project_title  \
        473    Flexible Seating for Flexible Learning
        41558  Going Deep: The Art of Inner Thinking!
```

```
                                         project_essay_1  \
473    I recently read an article about giving studen...
41558  My students crave challenge, they eat obstacle...

                                         project_essay_2  \
473    I teach at a low-income (Title 1) school. Ever...
41558  We are an urban, public k-5 elementary school...

                                         project_essay_3  \
473    We need a classroom rug that we can use as a c...
41558  With the new common core standards that have b...

                                         project_essay_4  \
473    Benjamin Franklin once said, \"Tell me and I f...
41558  These remarkable gifts will provide students w...

                                project_resource_summary  \
473    My students need flexible seating in the class...
41558  My students need copies of the New York Times ...

       teacher_number_of_previously_posted_projects  project_is_approved
473                                               2                    1
41558                                             2                    1
```

In [0]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[0]:
```
        id                                        description  quantity  \
0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
1  p069063        Bouncy Bands for Desks (Blue support pipes)         3

    price
0  149.00
1   14.95
```

1.3 1.2 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
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing '
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.4 1.3 preprocessing of project_subject_subcategories

```
In [0]: sub_catogories = list(project_data['project_subject_subcategories'].values)
        # remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

        # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
```

```
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.5 1.3 Text preprocessing

```
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  \
        473        100660  p234804  cbc0e38f522143b86d372f8b43d4cff3           Mrs.
        41558       33679  p137682  06f6e62e17de34fcf81020c77549e1d5           Mrs.

              school_state                 Date project_grade_category  \
        473             GA 2016-04-27 00:53:00          Grades PreK-2
        41558           WA 2016-04-27 01:05:25             Grades 3-5

                                            project_title  \
        473    Flexible Seating for Flexible Learning
        41558  Going Deep: The Art of Inner Thinking!

                                                  project_essay_1  \
        473    I recently read an article about giving studen...
        41558  My students crave challenge, they eat obstacle...

                                                  project_essay_2  \
        473    I teach at a low-income (Title 1) school. Ever...
        41558  We are an urban, public k-5 elementary school...

                                                  project_essay_3  \
        473    We need a classroom rug that we can use as a c...
        41558  With the new common core standards that have b...

                                                  project_essay_4  \
        473    Benjamin Franklin once said, \"Tell me and I f...
        41558  These remarkable gifts will provide students w...

                                         project_resource_summary  \
        473    My students need flexible seating in the class...
        41558  My students need copies of the New York Times ...
```

```
       teacher_number_of_previously_posted_projects  project_is_approved  \
473                                               2                    1
41558                                             2                    1

        clean_categories clean_subcategories  \
473      AppliedLearning    EarlyDevelopment
41558  Literacy_Language            Literacy

                                                   essay
473    I recently read an article about giving studen...
41558  My students crave challenge, they eat obstacle...
```

In [0]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

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)
#print(project_data['essay'].values[99999])
#print("="*50)
```

```
I recently read an article about giving students a choice about how they learn. We already set goals; why not let t
==================================================
At the beginning of every class we start out with a Math Application problem to help students see the relevance o
==================================================
My students love coming to school and they love learning. I strive daily to make our classroom a relaxed, comfort
==================================================
I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school provides free brea
==================================================
```

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)
```

```
I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school provides free brea
==================================================
```

```
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)
```

```
I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school provides free brea
```

```
In [0]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        print(sent)
```

```
I teach at a Title 1 school with 73 of my students who receive free reduced lunch Our school provides free breakfa
```

```
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', 'between', 'into', 'through', 'during', 'before', 'after',\
                    'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
                    'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', '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', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
                    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
                    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "
                    '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%|| 50000/50000 [00:32<00:00, 1552.22it/s]
```

```
In [0]: # after preprocesing
        preprocessed_essays[20000]
```

```
Out[0]: 'teach title 1 school 73 students receive free reduced lunch school provides free breakfast students special
```

1.4 Preprocessing of project_title

```
In [0]: # similarly you can preprocess the titles also

        from tqdm import tqdm
        preprocessed_project_title = []
        # 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 not in stopwords)
            preprocessed_project_title.append(sent.lower().strip())
```

```
100%|| 50000/50000 [00:01<00:00, 32611.76it/s]
```

1.6 1.5 Preparing data for models

```
In [0]: project_data.columns
```

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

we are going to consider

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

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

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

1.6.1 Modifying DataSet (essay & project_title)

```
In [0]: project_data['clean_essay'] = preprocessed_essays
        project_data['clean_project_title'] = preprocessed_project_title
        project_data.drop(['essay'], axis=1, inplace=True)
        project_data.drop(['project_title'], axis=1, inplace=True)
```

```
In [0]: project_data.head(1)
```

```
Out[0]:     Unnamed: 0        id                        teacher_id teacher_prefix  \
        473      100660  p234804  cbc0e38f522143b86d372f8b43d4cff3            Mrs.

            school_state                 Date project_grade_category  \
        473           GA 2016-04-27 00:53:00          Grades PreK-2

                                              project_essay_1  \
        473  I recently read an article about giving studen...

                                              project_essay_2  \
        473  I teach at a low-income (Title 1) school. Ever...

                                              project_essay_3  \
        473  We need a classroom rug that we can use as a c...

                                              project_essay_4  \
        473  Benjamin Franklin once said, \"Tell me and I f...

                                     project_resource_summary  \
473  My students need flexible seating in the class...

     teacher_number_of_previously_posted_projects  project_is_approved  \
473                                             2                    1

    clean_categories clean_subcategories  \
473  AppliedLearning    EarlyDevelopment

                                           clean_essay  \
473  recently read article giving students choice l...

                      clean_project_title
473  flexible seating flexible learning
```

1.7 Spliiting DataSet

```
In [0]: y = project_data['project_is_approved'].values
        project_data.drop(['project_is_approved'], axis=1, inplace=True)
        project_data.head(1)
```

```
Out[0]:     Unnamed: 0       id                        teacher_id teacher_prefix  \
473      100660  p234804  cbc0e38f522143b86d372f8b43d4cff3           Mrs.

    school_state                 Date project_grade_category  \
473           GA 2016-04-27 00:53:00          Grades PreK-2

                                       project_essay_1  \
473  I recently read an article about giving studen...

                                       project_essay_2  \
473  I teach at a low-income (Title 1) school. Ever...

                                       project_essay_3  \
473  We need a classroom rug that we can use as a c...

                                       project_essay_4  \
473  Benjamin Franklin once said, \"Tell me and I f...

                              project_resource_summary  \
473  My students need flexible seating in the class...

     teacher_number_of_previously_posted_projects clean_categories  \
473                                             2  AppliedLearning

    clean_subcategories                                        clean_essay  \
473    EarlyDevelopment  recently read article giving students choice l...

                      clean_project_title
```

```
       473  flexible seating flexible learning
```

```
In [0]: X = project_data
```

```
In [0]: # please write all the code with proper documentation, and proper titles for each subsection
        # go through documentations and blogs before you start coding
        # first figure out what to do, and then think about how to do.
        # reading and understanding error messages will be very much helpfull in debugging your code
        # when you plot any graph make sure you use
            # a. Title, that describes your plot, this will be very helpful to the reader
            # b. Legends if needed
            # c. X-axis label
            # d. Y-axis label


        # train test split
        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, stratify=y)
        X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

1.7.1 1.5.1 Vectorizing Categorical data

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

1.7.2 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_cc_ohe = vectorizer.transform(X_train['clean_categories'].values)
        X_cv_cc_ohe = vectorizer.transform(X_cv['clean_categories'].values)
        X_test_cc_ohe = vectorizer.transform(X_test['clean_categories'].values)

        feat_cc = vectorizer.get_feature_names()

        print("After vectorizations")
        print(X_train_cc_ohe.shape, y_train.shape)
        print(X_cv_cc_ohe.shape, y_cv.shape)
        print(X_test_cc_ohe.shape, y_test.shape)
        print(vectorizer.get_feature_names())
        print("="*100)
```

```
After vectorizations
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_science', 'music_a
```

```
==========================================================================================
```

1.7.3 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 vector
        X_train_csc_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
        X_cv_csc_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
        X_test_csc_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

        feat_csc = vectorizer.get_feature_names()


        print("After vectorizations")
        print(X_train_csc_ohe.shape, y_train.shape)
        print(X_cv_csc_ohe.shape, y_cv.shape)
        print(X_test_csc_ohe.shape, y_test.shape)
        print(vectorizer.get_feature_names())
        print("="*100)
```

```
After vectorizations
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_careerprep', 'communityserv
==========================================================================================
```

1.7.4 school_state

```
In [0]: #vectorizer = CountVectorizer()
        vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

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

        feat_ss = vectorizer.get_feature_names()


        print("After vectorizations")
        print(X_train_state_ohe.shape, y_train.shape)
        print(X_cv_state_ohe.shape, y_cv.shape)
        print(X_test_state_ohe.shape, y_test.shape)
        print(vectorizer.get_feature_names())
        print("="*100)
```

```
After vectorizations
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'm
==========================================================================
```

1.7.5 teacher_prefix

```
In [0]: #vectorizer = CountVectorizer()
        vectorizer.fit(X_train['teacher_prefix'].values.astype('U')) # fit has to happen only on train data

        # we use the fitted CountVectorizer to convert the text to vector
        X_train_teacher_ohe = vectorizer.transform(X_train['teacher_prefix'].values.astype('U'))
        X_cv_teacher_ohe = vectorizer.transform(X_cv['teacher_prefix'].values.astype('U'))
        X_test_teacher_ohe = vectorizer.transform(X_test['teacher_prefix'].values.astype('U'))

        feat_tp = vectorizer.get_feature_names()


        print("After vectorizations")
        print(X_train_teacher_ohe.shape, y_train.shape)
        print(X_cv_teacher_ohe.shape, y_cv.shape)
        print(X_test_teacher_ohe.shape, y_test.shape)
        print(vectorizer.get_feature_names())
        print("="*100)
```

```
After vectorizations
(22445, 5) (22445,)
(11055, 5) (11055,)
(16500, 5) (16500,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
==========================================================================
```

1.7.6 project_grade_category

```
In [0]: #vectorizer = CountVectorizer()
        vectorizer.fit(X_train['project_grade_category'].values) # fit has to happen only on train data

        # we use the fitted CountVectorizer to convert the text to vector
        X_train_grade_ohe = vectorizer.transform(X_train['project_grade_category'].values)
        X_cv_grade_ohe = vectorizer.transform(X_cv['project_grade_category'].values)
        X_test_grade_ohe = vectorizer.transform(X_test['project_grade_category'].values)
```

```
feat_pgc = vectorizer.get_feature_names()


print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 3) (22445,)
(11055, 3) (11055,)
(16500, 3) (16500,)
['12', 'grades', 'prek']
==========================================================================
```

1.7.7 1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

1.7.8 essays

```
In [0]: from sklearn.feature_extraction.text import CountVectorizer
        vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
        vectorizer.fit(X_train['clean_essay'].values) # fit has to happen only on train data
        feat_bow_e = vectorizer.get_feature_names()


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


        print("After vectorizations")
        print(X_train_essay_bow.shape, y_train.shape)
        print(X_cv_essay_bow.shape, y_cv.shape)
        print(X_test_essay_bow.shape, y_test.shape)
        print("="*100)
```

```
After vectorizations
(22445, 50000) (22445,)
(11055, 50000) (11055,)
(16500, 50000) (16500,)
==========================================================================
```

1.7.9 project_title

```
In [0]: from sklearn.feature_extraction.text import CountVectorizer
        #vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
        vectorizer.fit(X_train['clean_project_title'].values) # fit has to happen only on train data
        feat_bow_pt = vectorizer.get_feature_names()

        # we use the fitted CountVectorizer to convert the text to vector
        X_train_pt_bow = vectorizer.transform(X_train['clean_project_title'].values)
        X_cv_pt_bow = vectorizer.transform(X_cv['clean_project_title'].values)
        X_test_pt_bow = vectorizer.transform(X_test['clean_project_title'].values)

        print("After vectorizations")
        print(X_train_pt_bow.shape, y_train.shape)
        print(X_cv_pt_bow.shape, y_cv.shape)
        print(X_test_pt_bow.shape, y_test.shape)
        print("="*100)
```

```
After vectorizations
(22445, 2010) (22445,)
(11055, 2010) (11055,)
(16500, 2010) (16500,)
=================================================================================
```

1.5.2.2 TFIDF vectorizer

1.7.10 essays

```
In [0]: from sklearn.feature_extraction.text import TfidfVectorizer
        vectorizert = TfidfVectorizer(min_df=10)
        X_train_essay_tfidf = vectorizert.fit_transform(X_train['clean_essay'].values)
        X_cv_essay_tfidf = vectorizert.transform(X_cv['clean_essay'].values)
        X_test_essay_tfidf = vectorizert.transform(X_test['clean_essay'].values)

        feat_tfidf_e = vectorizert.get_feature_names()


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

```
Shape of matrix after one hot encodig  (22445, 8786)
Shape of matrix after one hot encodig  (11055, 8786)
Shape of matrix after one hot encodig  (16500, 8786)
```

1.7.11 project_title

```
In [0]: from sklearn.feature_extraction.text import TfidfVectorizer
        #vectorizer = TfidfVectorizer(min_df=10)
        X_train_pt_tfidf = vectorizert.fit_transform(X_train['clean_project_title'].values)
        X_cv_pt_tfidf = vectorizert.transform(X_cv['clean_project_title'].values)
        X_test_pt_tfidf = vectorizert.transform(X_test['clean_project_title'].values)

        feat_tfidf_pt = vectorizert.get_feature_names()


        print("Shape of matrix after one hot encodig ",X_train_pt_tfidf.shape)
        print("Shape of matrix after one hot encodig ",X_cv_pt_tfidf.shape)
        print("Shape of matrix after one hot encodig ",X_test_pt_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (22445, 1227)
Shape of matrix after one hot encodig  (11055, 1227)
Shape of matrix after one hot encodig  (16500, 1227)
```

1.7.12 1.5.3 Vectorizing Numerical features

1.7.13 price

```
In [0]: price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
        X_train = pd.merge(X_train, price_data, on='id', how='left')
        X_cv = pd.merge(X_cv, price_data, on='id', how='left')
        X_test = pd.merge(X_test, price_data, on='id', how='left')
```

```
In [0]: from sklearn.preprocessing import Normalizer
        normalizer = Normalizer()
        # normalizer.fit(X_train['price'].values)
        # this will rise an error Expected 2D array, got 1D array instead:
        # 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.
        normalizer.fit(X_train['price'].values.reshape(-1,1))

        X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
        X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))
        X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))

        print("After vectorizations")
        print(X_train_price_norm.shape, y_train.shape)
        print(X_cv_price_norm.shape, y_cv.shape)
        print(X_test_price_norm.shape, y_test.shape)
        print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
========================================================================
```

1.7.14 tnppp

```
In [0]: from sklearn.preprocessing import Normalizer
        normalizerT = Normalizer()
        # normalizer.fit(X_train['price'].values)
        # this will rise an error Expected 2D array, got 1D array instead:
        # 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.
        normalizerT.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

        X_train_tnppp_norm = normalizerT.transform(X_train['teacher_number_of_previously_posted_projec
        X_cv_tnppp_norm = normalizerT.transform(X_cv['teacher_number_of_previously_posted_projects'].va
        X_test_tnppp_norm = normalizerT.transform(X_test['teacher_number_of_previously_posted_projects'

        print("After vectorizations")
        print(X_train_tnppp_norm.shape, y_train.shape)
        print(X_cv_tnppp_norm.shape, y_cv.shape)
        print(X_test_tnppp_norm.shape, y_test.shape)
        print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
========================================================================
```

Feature_Aggregation

```
In [0]: feat_tnppp = list(X_train['teacher_number_of_previously_posted_projects'])
        feat_price = list(X_train['price'])
```

```
In [0]: feature_agg_bow = feat_cc + feat_csc + feat_ss + feat_tp + feat_pgc  + feat_bow + feat_bow_pt +
        feature_agg_tfidf = feat_cc + feat_csc + feat_ss + feat_tp + feat_pgc  + feat_tfidf_e + feat_tfidf_pt
```

1.7.15 1.5.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

SET1

```
In [0]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
        from scipy.sparse import hstack
        X_tr1 = hstack((X_train_cc_ohe, X_train_csc_ohe, X_train_grade_ohe, X_train_price_norm, X_tra
        X_cr1 = hstack((X_cv_cc_ohe, X_cv_csc_ohe, X_cv_grade_ohe, X_cv_price_norm, X_cv_tnppp_n
        X_te1 = hstack((X_test_cc_ohe, X_test_csc_ohe, X_test_grade_ohe, X_test_price_norm, X_test_tr

        print("Final Data matrix")
        print(X_tr1.shape, y_train.shape)
        print(X_cr1.shape, y_cv.shape)
        print(X_te1.shape, y_test.shape)
        print("="*100)
```

```
Final Data matrix
(22445, 52054) (22445,)
(11055, 52054) (11055,)
(16500, 52054) (16500,)
==================================================================
```

SET2

```
In [0]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
        from scipy.sparse import hstack
        X_tr2 = hstack((X_train_cc_ohe, X_train_csc_ohe, X_train_grade_ohe, X_train_price_norm, X_tra
        X_cr2 = hstack((X_cv_cc_ohe, X_cv_csc_ohe, X_cv_grade_ohe, X_cv_price_norm, X_cv_tnppp_n
        X_te2 = hstack((X_test_cc_ohe, X_test_csc_ohe, X_test_grade_ohe, X_test_price_norm, X_test_tr

        print("Final Data matrix")
        print(X_tr2.shape, y_train.shape)
        print(X_cr2.shape, y_cv.shape)
        print(X_te2.shape, y_test.shape)
        print("="*100)
```

```
Final Data matrix
(22445, 10057) (22445,)
(11055, 10057) (11055,)
(16500, 10057) (16500,)
==================================================================
```

2 Assignment 4: Naive Bayes

```
<li><strong>Apply Multinomial NaiveBayes on these feature sets</strong>
    <ul>
     <li><font color='red'>Set 1</font>: categorical, numerical features + project_title(BOW) + preprocessed_e
     <li><font color='red'>Set 2</font>: categorical, numerical features + project_title(TFIDF)+  preprocessed_
    </ul>
</li>
<br>
<li><strong>The hyper paramter tuning(find best Alpha)</strong>
    <ul>
<li>Find the best hyper parameter which will give the maximum <a href='https://www.appliedaicourse.com/cours
ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-
1/'>AUC</a> value</li>
<li>Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001</li>
<li>Find the best hyper paramter using k-fold cross validation or simple cross validation data</li>
<li>Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter
    </ul>
</li>
<br>
<li><strong>Feature importance</strong>
    <ul>
<li>Find the top 10 features of positive class and top 10 features of negative class for both feature sets <font color='r
learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html'>MultinomialNB</a> and print t
    </ul>
</li>
<br>
<li><strong>Representation of results</strong>
    <ul>
<li>You need to plot the performance of model both on train data and cross validation data for each hyper paramete
axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply
<img src='train_cv_auc.JPG' width=300px></li>
<li>Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test d
<img src='train_test_auc.JPG' width=300px></li>
<li>Along with plotting ROC curve, you need to print the <a href='https://www.appliedaicourse.com/course/appli
ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/'>confusion matrix</a> with predicted and original la
<img src='confusion_matrix.png' width=300px></li>
    </ul>
</li>
<br>
<li><strong>Conclusion</strong>
    <ul>
<li>You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a t
    <img src='summary.JPG' width=400px>
</li>
    </ul>
```

Note: Data Leakage

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

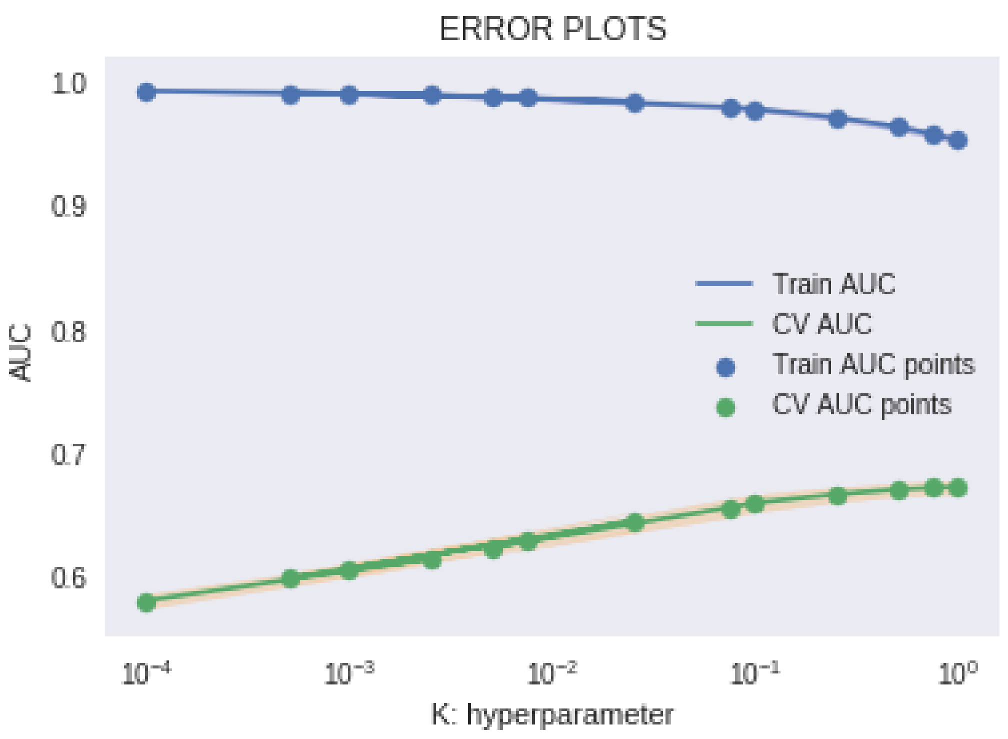
4. For more details please go through this link.

2. Naive Bayes

2.1 HyperParameter Tuning

2.1.1 SET1

```
In [0]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
        from sklearn.model_selection import GridSearchCV
        from sklearn.naive_bayes import MultinomialNB

        nb1 = MultinomialNB()

        parameters = {'alpha':[0.0001, 0.0025, 0.0005, 0.0075, 0.001, 0.025, .005, 0.075, 0.1, 0.25, 0.5, 0.75, 1]}
        clf = GridSearchCV(nb1, parameters, cv=3, scoring='roc_auc')
        mnb1 = clf.fit(X_tr1, y_train)

        train_auc1= clf.cv_results_['mean_train_score']
        train_auc_std1= clf.cv_results_['std_train_score']
        cv_auc1 = clf.cv_results_['mean_test_score']
        cv_auc_std1 = clf.cv_results_['std_test_score']

        plt.plot(parameters['alpha'], train_auc1, label='Train AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'],train_auc1 - train_auc_std1,train_auc1 + train_auc_std1,alph

        plt.plot(parameters['alpha'], cv_auc1, label='CV AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'],cv_auc1 - cv_auc_std1, cv_auc1 + cv_auc_std1,alpha=0.2,colo

        plt.scatter(parameters['alpha'], train_auc1, label='Train AUC points')
        plt.scatter(parameters['alpha'], cv_auc1, label='CV AUC points')

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

        print(train_auc1)
        print(cv_auc1)       #k_best = 1.0
```

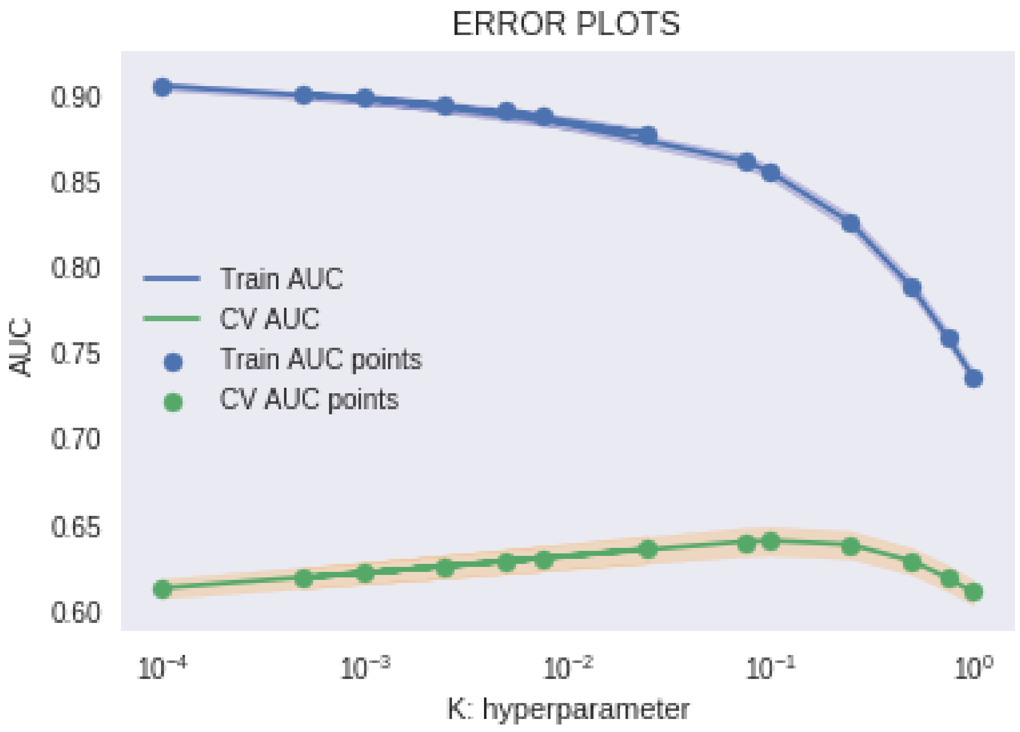


```
[0.9923867  0.98895263 0.9909895  0.9869453  0.99021096 0.98373462
 0.9877678  0.97920372 0.97762801 0.9709549  0.96344119 0.95754415
 0.95244305]
[0.58100178 0.61471396 0.59906309 0.63055899 0.6067468  0.64522506
 0.62382264 0.65615951 0.65996501 0.66686024 0.6708156  0.67218491
 0.67268854]
```

2.1.2 SET2

```
In [0]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
        from sklearn.model_selection import GridSearchCV
        from sklearn.naive_bayes import MultinomialNB

        nb2 = MultinomialNB()
        parameters = {'alpha':[0.0001, 0.0025, 0.0005, 0.0075, 0.001, 0.025, .005, 0.075, 0.1, 0.25, 0.5, 0.75, 1]}
        clf = GridSearchCV(nb2, parameters, cv=3, scoring='roc_auc')
        mnb2 = clf.fit(X_tr2, y_train)

        train_auc2= clf.cv_results_['mean_train_score']
        train_auc_std2= clf.cv_results_['std_train_score']
        cv_auc2 = clf.cv_results_['mean_test_score']
        cv_auc_std2 = clf.cv_results_['std_test_score']
```

```
plt.plot(parameters['alpha'], train_auc2, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc2 - train_auc_std2,train_auc2 + train_auc_std2,alph

plt.plot(parameters['alpha'], cv_auc2, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc2 - cv_auc_std2, cv_auc2 + cv_auc_std2,alpha=0.2,colo

plt.scatter(parameters['alpha'], train_auc2, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc2, label='CV AUC points')

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

print(train_auc2)
print(cv_auc2)          #k_best = 0.1
```



```
[0.90473977 0.89372376 0.90025686 0.88732272 0.89776442 0.87695656
```

```
 0.8899252  0.86094173 0.85482369 0.8253908  0.78801296 0.75906122
 0.7361055 ]
[0.61292205 0.62544956 0.61889038 0.63022194 0.62166762 0.63567136
 0.62845323 0.63994621 0.64053802 0.63823286 0.62874746 0.61904502
 0.61071504]
```

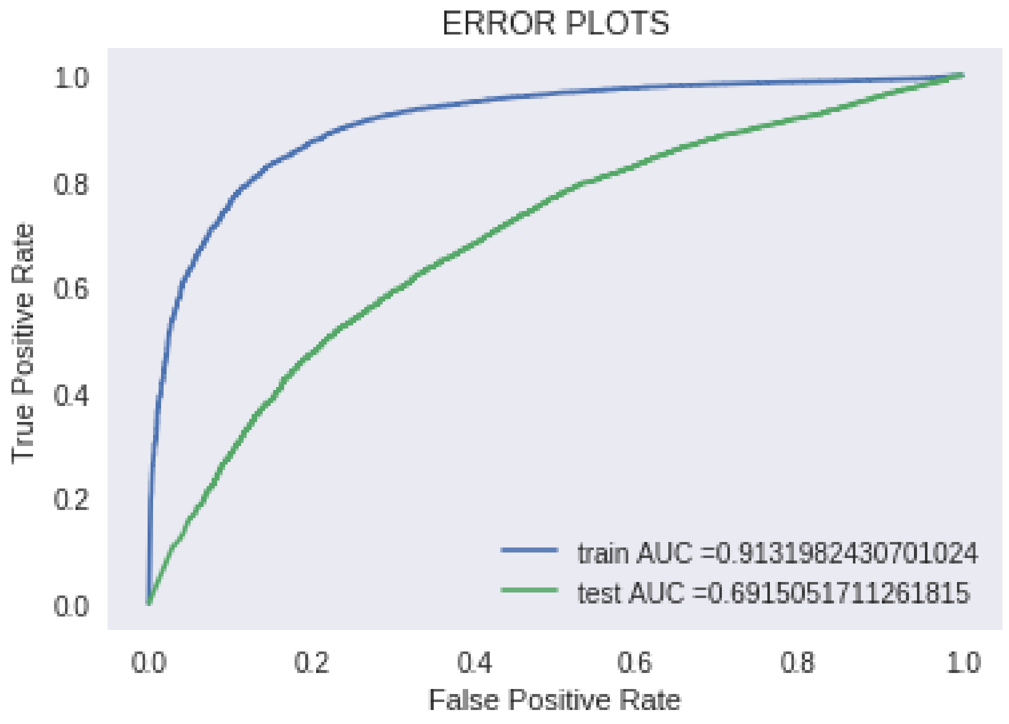
2.1.3 ROC Curve

SET1

```
In [0]: def batch_predict(clf, data):
            # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive cl
            # not the predicted outputs

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

            return y_data_pred
```

```
In [0]: # https://scikit-leahttps://stackoverflow.com/questions/29867367/sklearn-multinomial-nb-most-informativ

        best_k1 = 1

        from sklearn.metrics import roc_curve, auc
        from sklearn.naive_bayes import MultinomialNB



        nb1 = MultinomialNB(alpha = best_k1)
        nb1.fit(X_tr1, y_train)
        # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
        # not the predicted outputs

        y_train_pred1 = batch_predict(nb1, X_tr1)
        y_test_pred1 = batch_predict(nb1, X_te1)

        train_fpr1, train_tpr1, tr_thresholds1 = roc_curve(y_train, y_train_pred1)
        test_fpr1, test_tpr1, te_thresholds1 = roc_curve(y_test, y_test_pred1)

        plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr1, train_tpr1)))
        plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr1, test_tpr1)))
        plt.legend()
```

```
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ERROR PLOTS")
    plt.grid()
    plt.show()
```

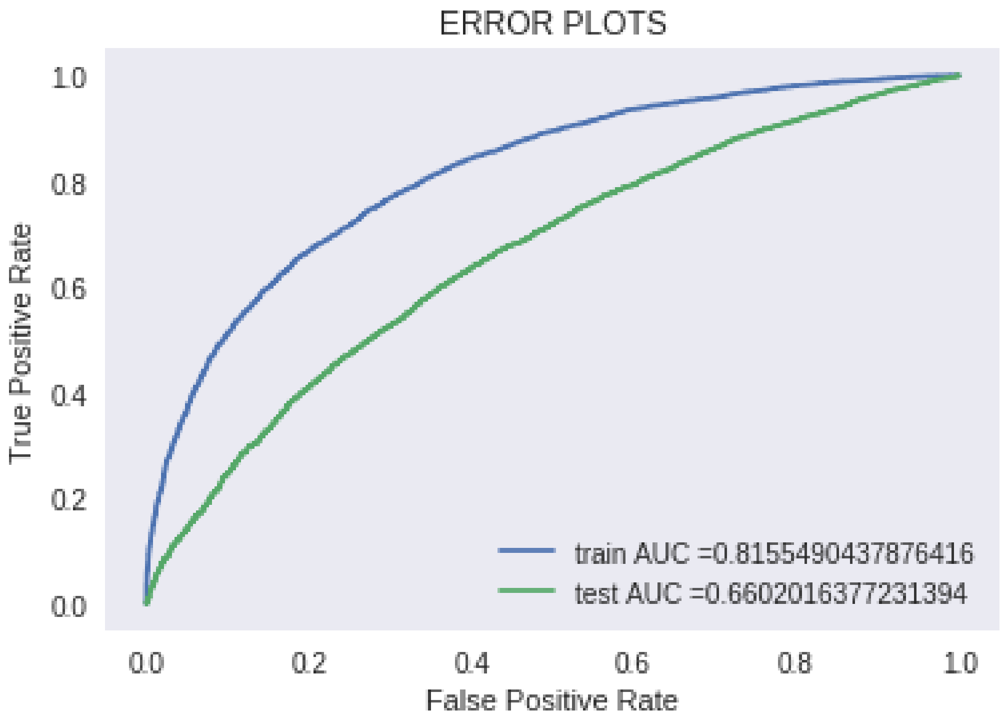


2.1.4 SET2

```
In [0]: def batch_predict(clf, data):
            # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive cl
            # not the predicted outputs

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

            return y_data_pred
```

```
In [0]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_
```

```
best_k2 = 0.1

from sklearn.metrics import roc_curve, auc
from sklearn.naive_bayes import MultinomialNB



nb2 = MultinomialNB(alpha = best_k2)
nb2.fit(X_tr2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred2 = batch_predict(nb2, X_tr2)
y_test_pred2 = batch_predict(nb2, X_te2)

train_fpr2, train_tpr2, tr_thresholds2 = roc_curve(y_train, y_train_pred2)
test_fpr2, test_tpr2, te_thresholds2 = roc_curve(y_test, y_test_pred2)

plt.plot(train_fpr2, train_tpr2, label="train AUC ="+str(auc(train_fpr2, train_tpr2)))
plt.plot(test_fpr2, test_tpr2, label="test AUC ="+str(auc(test_fpr2, test_tpr2)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



2.1.5 Confusion Matrix

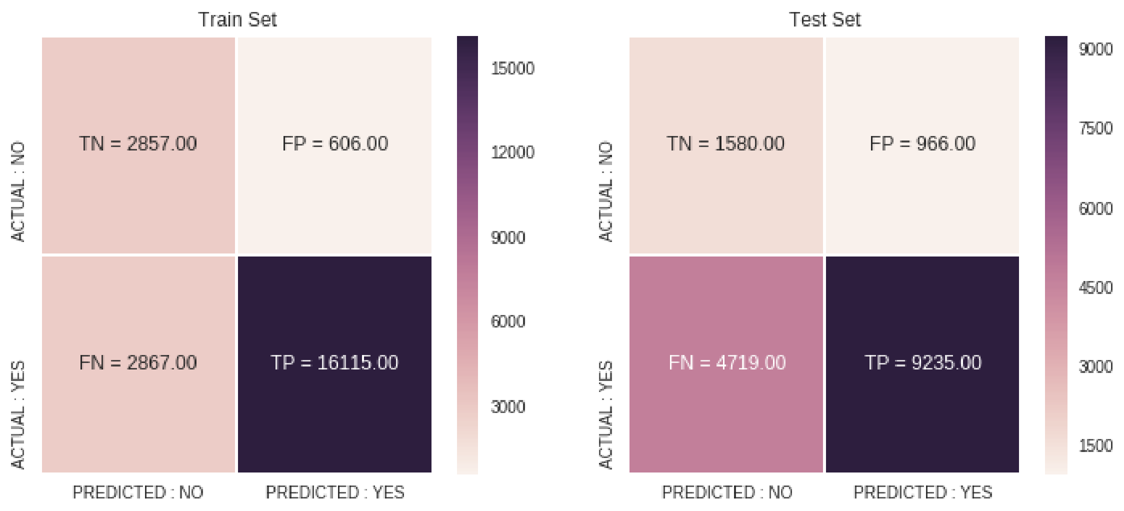
SET1

```
In [0]: def predict(proba, threshould, fpr, tpr):

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

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

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

```
In [0]: #https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
        import seaborn as sns; sns.set()

        con_m_train = confusion_matrix(y_train, predict(y_train_pred1, tr_thresholds1, train_fpr1, train_tpr1
        con_m_test = confusion_matrix(y_test, predict(y_test_pred1, te_thresholds1, test_fpr1, test_tpr1))

        key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

        fig, ax = plt.subplots(1,2, figsize=(12,5))

        labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(), con_m_
        labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(), con_m_te

        sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], ytick
        sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklab

        ax[0].set_title('Train Set')
        ax[1].set_title('Test Set')

        plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.7073058528965523 for threshold

 0.701
the maximum value of tpr*(1-fpr) 0.41694824685441023 for threshold
```

```
0.998
```



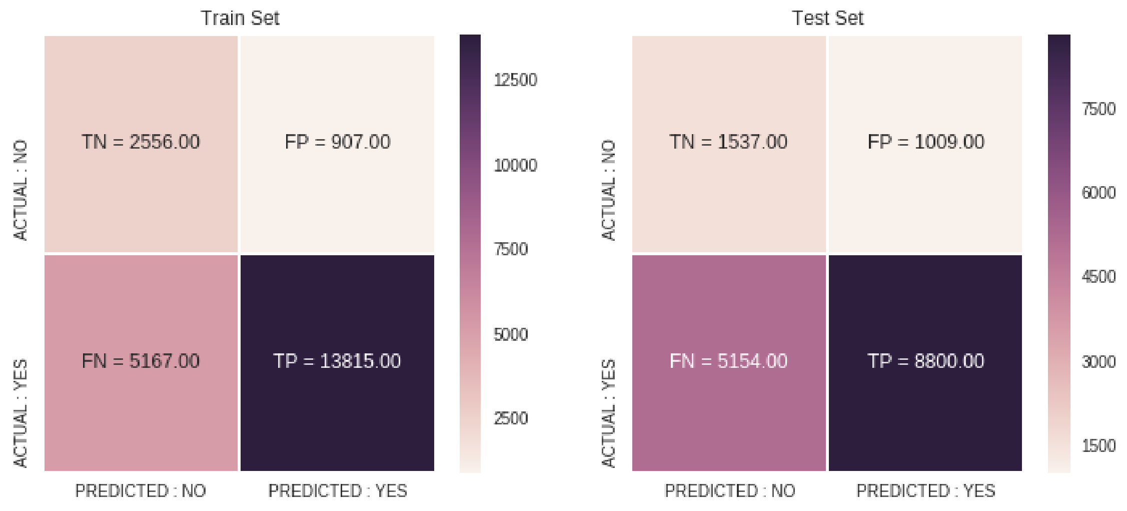
2.1.6 SET2

In [0]:
```
def predict(proba, threshould, fpr, tpr):

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

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

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

In [0]:
```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()

con_m_train = confusion_matrix(y_train, predict(y_train_pred2, tr_thresholds2, train_fpr2, train_tpr
con_m_test = confusion_matrix(y_test, predict(y_test_pred2, te_thresholds2, test_fpr2, test_tpr2))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(12,5))
```

```
labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(), con_m_
labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(), con_m_te

sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], ytickl
sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklab

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')

plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.541171442173297 for threshold

 0.846
the maximum value of tpr*(1-fpr) 0.38342887037320805 for threshold

 0.868
```



2.1.7 Feature Importance

2.1.8 SET1

```
In [0]: pos_cls_sort_prob_1 = np.argsort((nb1.feature_log_prob_)[1])[: :-1][0:10]
        neg_cls_sort_prob_1 = np.argsort((nb1.feature_log_prob_)[0])[: :-1][0:10]

        pcsp = feature_agg_bow

        print("Top 10 +ve class features : \n\n{0}\n\n" .format(np.take(pcsp, pos_cls_sort_prob_1)))
        print("Top 10 -ve class features : \n\n{0}" .format(np.take(pcsp, neg_cls_sort_prob_1)))
```

```
Top 10 +ve class features :

['student population school' 'saying not' 'learners nannan' 'class would'
 'non english' 'laugh' 'hear daily' 'al' 'az' 'making students']


Top 10 -ve class features :

['student population school' 'saying not' 'learners nannan' 'class would'
 'non english' 'laugh' 'hear daily' 'az' 'al' 'music room']
```

2.1.9 SET2

```
In [0]: pos_cls_sort_prob_2 = np.argsort(abs(nb2.feature_log_prob_)[1])[: :-1][0:10]
        neg_cls_sort_prob_2 = np.argsort(abs(nb2.feature_log_prob_)[0])[: :-1][0:10]

        pcspt = feature_agg_tfidf

        print("Top 10 +ve class features : \n\n{0}\n\n" .format(np.take(pcspt, pos_cls_sort_prob_2)))
        print("Top 10 -ve class features : \n\n{0}" .format(np.take(pcspt, neg_cls_sort_prob_2)))
```

```
Top 10 +ve class features :

['wave' 'completes' 'chooses' 'speakers' 'minds' 'reported' 'myriad'
 'speaking' 'product' 'socratic']


Top 10 -ve class features :

['redesigning' 'coloring' 'afford' 'reflect' 'coming' 'comic' 'register'
 'rehearsal' 'reliability' 'repeat']
```

3. Conclusions

```
In [0]: # Please compare all your models using Prettytable library

        from prettytable import PrettyTable

        x = PrettyTable()
        x.field_names = ["Vectorizer", "Model", "HyperParameter", "AUC"]

        x.add_row(["BOW", "Multinomial Naive Bayes", 1.0, 0.691])
        x.add_row(["TFIDF", "Multinomial Naive Bayes", 0.1, 0.660])

        print(x)
```

```
+------------+-------------------------+----------------+-------+
| Vectorizer |          Model          | HyperParameter |  AUC  |
+------------+-------------------------+----------------+-------+
|    BOW     | Multinomial Naive Bayes |      1.0       | 0.691 |
|   TFIDF    | Multinomial Naive Bayes |      0.1       |  0.66 |
+------------+-------------------------+----------------+-------+
```

- ** By seeing AUC score, we can come to conclusion that my NB model is performing average, any way better than Random model.**
- ** So when we take any new datapoint, there is 69% chance of getting correctly classified for SET1 and 66% chance of getting correctly classified for SET2 **

2.1.10 Steps Followed

I took **50000 datapoints** for my analysis and building my model

- I splitted the dataset into train, cv and test dataset
- Preprocessed all the text fetaures
- Vectorized all the text, categorical and numerical features, for text i used BOW & TFIDF
- Merged all features using hstack as instructed
- Using train dataset, i plotted my AUC curve using GridSearchCV using 3Fold Cross Validation for both categories
- from AUC curve, i picked best alpha. using best alpha, i plotted ROC curve on train and test data.
- Then i plotted my confusion matrix for both the sets.
- I chose top 20 features for both +ve and -ve class for both the sets.
- Atlast you can see my result in tabular format.