Install and Import the libraries

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore",category=UserWarning)

# Data Manipulation libraries
import pandas as pd
import numpy as np

#Plotting libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Estimators and metrics

from sklearn.preprocessing import Normalizer
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer,TfidfVectorizer
from sklearn.model_selection import train_test_split,GridSearchCV,RandomizedSearchCV,cross_validate
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.metrics import confusion_matrix,roc_curve,auc,roc_auc_score,accuracy_score

# NLP libraries
import nltk
from nltk.corpus import stopwords
from gensim.models import Word2Vec,KeyedVectors

import re
import pickle

from tqdm import tqdm
from collections import Counter
from scipy.sparse import hstack
#Code Reference: https://ptable.readthedocs.io/en/latest/tutorial.html
from prettytable import PrettyTable
```

In [2]:

```
# Read the data into Pandas Dataframe

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

In [3]:

```
print('Number of data points in the  Train dataset :',project_data.shape[0])
print("-"*53)
print('Number of features in the Train dataset :',project_data.shape[1])
print("-"*53)
print("List of Features in the Train dataset:\n",project_data.columns.values.tolist())
```

```
Number of data points in the  Train dataset : 109248
-----------------------------------------------------
Number of features in the Train dataset : 17
-----------------------------------------------------
List of Features in the Train dataset:
 ['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state', 'project_submitted_datetime', 'project_grade_category', 'proj
ect_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 [4]:

```
print('Number of data points in the  Resourse dataset :',resource_data.shape[0])
print("-"*55)
print('Number of features in the Resourse dataset :',resource_data.shape[1])
print("-"*55)
print("List of Features in the Resourse dataset:",resource_data.columns.values.tolist())
```

```
Number of data points in the  Resourse dataset : 1541272
-------------------------------------------------------
Number of features in the Resourse dataset : 4
-------------------------------------------------------
List of Features in the Resourse dataset: ['id', 'description', 'quantity', 'price']
```

In [5]:

```
cols=['Date' if each_col=='project_submitted_datetime' else each_col for each_col in project_data.columns.values.tolist()]

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)
```

In [6]:

```
project_data=project_data[cols]

print("Sample records from  Training data ")
project_data.head()
```

```
Sample records from  Training data
```

Out[6]:

	Unnamed:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_subject_categories	project_subject_subcategories
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2	Math & Science	Applied Sciences, Health & Life Science
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	Special Needs	Special Needs
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Grades PreK-2	Literacy & Language	Literacy
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Grades PreK-2	Applied Learning	Early Development
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5	Literacy & Language	Literacy

In [7]:

```
print("Sample records from  Resourse data ")
resource_data.head()
```

```
Sample records from  Resourse data
```

Out[7]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

Data Analysis

Data Analysis

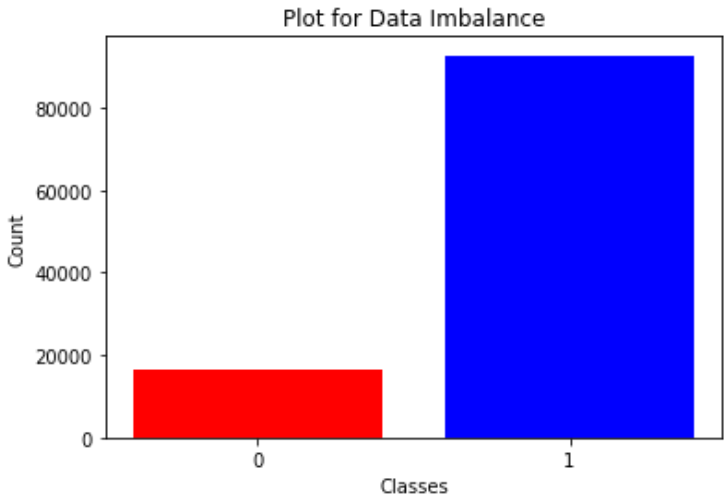
In [8]:

```
def check_class_bal(dataset,target_class):
    count_per_class=list(dataset[target_class].value_counts())
    classes=list(dataset[target_class].value_counts().index)

    print("Ratio of the classes :")

    for each_cls,cls_count in zip(classes,count_per_class):
        print("Class {} has {} records with a ratio of {}%".
              format(each_cls,cls_count,np.round((cls_count/dataset.shape[0]*100),2)))

    plt.bar(classes,count_per_class,color=['b','r'])
    plt.xticks(classes)
    plt.ylabel("Count")
    plt.xlabel("Classes")
    plt.title("Plot for Data Imbalance")
    plt.show()

    del classes
    del count_per_class
```

In [9]:

```
check_class_bal(project_data,'project_is_approved')
```

```
Ratio of the classes :
Class 1 has 92706 records with a ratio of 84.86%
Class 0 has 16542 records with a ratio of 15.14%
```



Data Preprocessing

chek for null values

In [10]:

```
print("Null values from Train data :\n")
print(project_data.isnull().sum())
```

```
Null values from Train data :

Unnamed: 0                                          0
id                                                  0
teacher_id                                          0
teacher_prefix                                      3
school_state                                        0
Date                                                0
project_grade_category                              0
project_subject_categories                          0
project_subject_subcategories                       0
project_title                                       0
project_essay_1                                     0
project_essay_2                                     0
project_essay_3                                105490
project_essay_4                                105490
project_resource_summary                            0
teacher_number_of_previously_posted_projects        0
project_is_approved                                 0
dtype: int64
```

In [11]:

```
project_data['teacher_prefix'].fillna(method='ffill',inplace=True)
```

In [12]:

```
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 [13]:

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

In [14]:

```
project_data.isnull().sum()
```

Out[14]:

```
Unnamed: 0                                      0
id                                              0
teacher_id                                      0
teacher_prefix                                  0
school_state                                    0
Date                                            0
project_grade_category                          0
project_subject_categories                      0
project_subject_subcategories                   0
project_title                                   0
project_resource_summary                        0
teacher_number_of_previously_posted_projects    0
project_is_approved                             0
essay                                           0
dtype: int64
```

In [15]:

```
print("Null values from Train data :\n")
print(resource_data.isnull().sum())
```

```
Null values from Train data :

id              0
description    292
quantity         0
price            0
dtype: int64
```

In [16]:

```
resource_data['description'].fillna(method='ffill',inplace=True)
```

In [17]:

```
resource_data.isnull().sum()
```

Out[17]:

```
id             0
description    0
quantity       0
price          0
dtype: int64
```

Text Pre-processing

Text Pre-processing

In [18]:

```
def processed_list(list_elements):
    processed_list=[]
    for i in list_elements:
        temp=''
        for j in i.split(','):
            if 'The' in j.split():
                j=j.replace('The','')
            j=j.replace(' ','')
            temp+=j.strip()+' '
            temp=temp.replace('&','_')
        processed_list.append(temp.strip())
    return processed_list
```

In [19]:

```
def get_sorted_dic(col):
    my_Counter=Counter()
    for word in list(project_data[col]):
        my_Counter.update(word.split())
    count_dict=dict(my_Counter)
    return dict(sorted(count_dict.items(),key=lambda x: x[1]))
```

project_subject_categories

In [20]:

```
clean_categories=processed_list(list(project_data['project_subject_categories']))
project_data['clean_categories']=clean_categories
project_data.drop(['project_subject_categories'],axis=1,inplace=True)
sorted_cat_dict=get_sorted_dic('clean_categories')
```

project_subject_subcategories

In [21]:

```
clean_sub_categories=processed_list(list(project_data['project_subject_subcategories']))
project_data['clean_sub_categories']=clean_sub_categories
project_data.drop(['project_subject_subcategories'],axis=1,inplace=True)
sorted_subcat_dict=get_sorted_dic('clean_sub_categories')
```

essay

In [22]:

```
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 [23]:

```
# 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', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [24]:

```
def text_processing(dataset,feature_name):
    processed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(dataset[feature_name].values):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\"', ' ')
        sent = sent.replace('\\n', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
```

```
        processed_text.append(sent.lower().strip())
    return processed_text
```

In [25]:

```
project_data['essay']=text_processing(project_data,'essay')
```

```
100%|██████████| 109248/109248 [01:09<00:00, 1575.30it/s]
```

project_title

In [26]:

```
project_data['project_title']=text_processing(project_data,'project_title')
```

```
100%|██████████| 109248/109248 [00:03<00:00, 33802.43it/s]
```

project_resource_summary

In [27]:

```
project_data['project_resource_summary']=text_processing(project_data,'project_resource_summary')
```

```
100%|██████████| 109248/109248 [00:07<00:00, 14527.70it/s]
```

project_grade_category

In [28]:

```
processed_grade=[]

for each_grade in tqdm(project_data['project_grade_category'].values):
    temp=""
    temp=each_grade.lower()
    temp=temp.replace(' ','_')
    temp=temp.replace('-','_')
    processed_grade.append(temp)

project_data['project_grade_category']=processed_grade
```

```
100%|██████████| 109248/109248 [00:00<00:00, 923818.11it/s]
```

In [29]:

```
# Merge the projectdata and pricedata by using id feature
```

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

In [30]:

```
print("Final Feature Names:\n\n", list(project_data.columns))
print("\nSample Data set")
project_data.head()
```

```
Final Feature Names:

 ['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state', 'Date', 'project_grade_category', 'project_title', 'project_r
esource_summary', 'teacher_number_of_previously_posted_projects', 'project_is_approved', 'essay', 'clean_categories', 'clean_sub_ca
tegories', 'quantity', 'price']

Sample Data set
```

Out[30]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_title	project_resource_summary	teacher_number_of_pr
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	grades_prek_2	engineering steam primary classroom	students need stem kits learn critical science...	
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	grades_3_5	sensory tools focus	students need boogie boards quiet sensory brea...	
2	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	grades_prek_2	mobile learning mobile listening center	students need mobile listening center able enh...	
3	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	grades_prek_2	flexible seating flexible learning	students need flexible seating classroom choos...	
4	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	grades_3_5	going deep art inner thinking	students need copies new york times best selle...	

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_title	project_resource_summary	teacher_number_of_pr

In [31]:

```
y = project_data['project_is_approved'].values
X=project_data.drop(['project_is_approved'], axis=1)
project_data.head(3)
```

Out[31]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_title	project_resource_summary	teacher_number_of_pr
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	grades_prek_2	engineering steam primary classroom	students need stem kits learn critical science...	
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	grades_3_5	sensory tools focus	students need boogie boards quiet sensory brea...	
2	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	grades_prek_2	mobile learning mobile listening center	students need mobile listening center able enh...	

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, stratify=y)
```

In [33]:

```
print("Training data set shape:",X_train.shape)
print("Test data set shape:",X_test.shape)
```

```
Training data set shape: (76473, 15)
Test data set shape: (32775, 15)
```

Feature Vectorization

teacher_prefix

In [34]:

```
vectorizer=CountVectorizer()
vectorizer.fit(X_train.teacher_prefix.values)
X_tr_teacher_onehot=vectorizer.transform(X_train.teacher_prefix.values)
X_te_teacher_onehot=vectorizer.transform(X_test.teacher_prefix.values)
```

school_state

In [35]:

```
vectorizer=CountVectorizer()
vectorizer.fit(X_train.school_state.values)
X_tr_school_onehot=vectorizer.transform(X_train.school_state.values)
X_te_school_onehot=vectorizer.transform(X_test.school_state.values)
```

project_grade_category

In [36]:

```
vectorizer=CountVectorizer()
vectorizer.fit(X_train.project_grade_category.values)
X_tr_grade_onehot=vectorizer.transform(X_train.project_grade_category.values)
X_te_grade_onehot=vectorizer.transform(X_test.project_grade_category.values)
```

clean_categories

In [37]:

```
vectorizer=CountVectorizer()
vectorizer.fit(X_train.clean_categories.values)
X_tr_cat_onehot=vectorizer.transform(X_train.clean_categories.values)
X_te_cat_onehot=vectorizer.transform(X_test.clean_categories.values)
```

clean_sub_categories

In [38]:

```
vectorizer=CountVectorizer()
vectorizer.fit(X_train.clean_sub_categories.values)
X_tr_sub_cat_onehot=vectorizer.transform(X_train.clean_sub_categories.values)
```

```
X_te_sub_cat_onehot=vectorizer.transform(X_test.clean_sub_categories.values)
```

Normalization

price

In [39]:

```
nrml= Normalizer()
nrml.fit(X_train['price'].values.reshape(1,-1))

X_tr_price_nrml = nrml.transform(X_train.price.values.reshape(1,-1)).reshape(-1,1)
X_te_price_nrml = nrml.transform(X_test.price.values.reshape(1,-1)).reshape(-1,1)
```

teacher_number_of_previously_posted_projects

In [40]:

```
nrml = Normalizer()
nrml.fit(X_train.teacher_number_of_previously_posted_projects.values.reshape(1,-1))
X_tr_teacher_number_nrml = nrml.transform(X_train.teacher_number_of_previously_posted_projects.values.reshape(1,-1)).reshape(-1,1)
X_te_teacher_number_nrml = nrml.transform(X_test.teacher_number_of_previously_posted_projects.values.reshape(1,-1)).reshape(-1,1)
```

In [41]:

```
X_tr_vec=hstack((X_tr_teacher_onehot,X_tr_school_onehot,X_tr_grade_onehot,X_tr_cat_onehot,
                 X_tr_sub_cat_onehot,X_tr_price_nrml,X_tr_teacher_number_nrml)).tocsr()
X_te_vec=hstack((X_te_teacher_onehot,X_te_school_onehot,X_te_grade_onehot,X_te_cat_onehot, X_te_sub_cat_onehot,
                 X_te_price_nrml,X_te_teacher_number_nrml)).tocsr()
```

In [42]:

```
print("After stacking :")
print("Training data set shape :",X_tr_vec.shape)
print("Test data set shape :",X_te_vec.shape)
```

```
After stacking :
Training data set shape : (76473, 101)
Test data set shape : (32775, 101)
```

Model Training

Hyperparameter Values

Hypertuning Values

In [43]:

```
c_values=[10**i  for i in range(-4,4)]
model=LogisticRegression()
```

In [44]:

```
print(" Default Model:\n",model)
```

```
 Default Model:
 LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [45]:

```
# Avoid the bais towards the class which has more number of observations
model.class_weight='balanced'
```

In [46]:

```
def cross_validate_plot(hyper_vals,X_train,y_train):
    auc_scores={}
    train_auc=[]
    cv_auc=[]

    for each_c in tqdm(hyper_vals):
        clf=LogisticRegression(C=each_c)
        clf.class_weight='balanced'
        auc_scores[each_c]=cross_validate(clf, X_train, y_train, cv=5, scoring='roc_auc',return_train_score=True,n_jobs=-1)

    for each_c in hyper_vals:
        train_auc.append(auc_scores[each_c]['train_score'].mean())
        cv_auc.append(auc_scores[each_c]['test_score'].mean())

    hyper_vals=np.log10(hyper_vals)

    plt.plot(hyper_vals, train_auc, label='Train AUC')
    plt.scatter(hyper_vals,train_auc)

    plt.plot(hyper_vals, cv_auc, label='CV AUC')
    plt.scatter(hyper_vals,cv_auc)

    plt.title("AUC PLOT for Train and CV datasets")
    plt.legend()
    plt.xlabel("C: Hyper parameter to the 10th power")
    plt.ylabel("AUC Score")
```

```
    plt.show()

    del auc_scores
    del train_auc
    del cv_auc
```

In [47]:

```
def build_best_model_plot_roc(model,X_train_data,y_train_data,X_test_data,y_test_data):


    model.fit(X_train_data,y_train_data)

    y_tr_pred_prob=model.predict_proba(X_train_data)
    y_te_pred_prob=model.predict_proba(X_test_data)

    plot_roc([y_train_data,y_tr_pred_prob[:,1]],[y_test_data,y_te_pred_prob[:,1]])

    del y_tr_pred_prob
    del y_te_pred_prob
```

In [48]:

```
def plot_roc(y_train,y_test):

    fpr_tr,tpr_tr,thr_tr=roc_curve(y_train[0],y_train[1])
    fpr_te,tpr_te,thr_te=roc_curve(y_test[0],y_test[1])

    plt.plot(fpr_tr,tpr_tr,label="AUC score for Train data is : {}".format(np.round(auc(fpr_tr,tpr_tr),4)))
    plt.plot(fpr_te,tpr_te,label="AUC score is Test data is  : {}".format(np.round(auc(fpr_te,tpr_te),4)))

    plt.plot([0,1],[0,1],'k--',label="Random Curve AUC score is :{}".format(0.5))

    plt.title("ROC Curve for Train and Test data")
    plt.legend()
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.show()

    print('-'*90)
    cutoof_thr=thr_tr[np.argmax(tpr_tr*(1-fpr_tr))]

    print("The Maximum value of 'TPR*(1-FPR)' is {} for 'THRESHOLD VALUE'of {}"
          .format(max(tpr_tr*(1-fpr_tr)),np.round(cutoof_thr,3)))

    print('-'*90)

    y_train_pred=predict_with_best_t(y_train[1],cutoof_thr)
    y_test_pred=predict_with_best_t(y_test[1],cutoof_thr)
```

```
    plot_confusion_matrix(y_train[0],y_train_pred,"TRAIN DATA")
    plot_confusion_matrix(y_test[0],y_test_pred,"TEST DATA")

    display_accuracy([y_train[0],y_train_pred],[y_test[0],y_test_pred])

    del y_train_pred
    del y_test_pred
```

In [49]:

```
def predict_with_best_t(pred_proba, cut_off):
    pred= []
    for i in pred_proba:
        if i>=cut_off:
            pred.append(1)
        else:
            pred.append(0)
    return pred
```

In [50]:

```
def plot_confusion_matrix(y_true,y_pred,set_name):
    sns.heatmap(confusion_matrix(y_true,y_pred), annot=True, fmt="d",cmap="YlGnBu")
    plt.title("Confusion Matrix for {}".format(set_name))
    plt.xlabel("Predicted labels")
    plt.ylabel("Actual labels")
    plt.show()
```

In [51]:

```
def display_accuracy(train_res,test_res):

    acc_table=PrettyTable()
    acc_table.field_names = ["Training Accuracy","Test Accuracy"]
    acc_table.add_row([np.round(accuracy_score(train_res[0],train_res[1]),3),
                       np.round(accuracy_score(test_res[0],test_res[1]),3)])
    print(acc_table)
```

TASK-1

Bag Of Words

project_title

In [52]:

```
vectorizer=CountVectorizer(ngram_range=(1,2),max_features=5000,min_df=10)
vectorizer.fit(X_train.project_title.values)
X_tr_title=vectorizer.transform(X_train.project_title.values)
X_te_title=vectorizer.transform(X_test.project_title.values)
```

essay

In [53]:

```
vectorizer=CountVectorizer(ngram_range=(1,2),max_features=5000,min_df=10)
vectorizer.fit(X_train.essay.values)
X_tr_essay=vectorizer.transform(X_train.essay.values)
X_te_essay=vectorizer.transform(X_test.essay.values)
```

project_resource_summary

In [54]:

```
vectorizer=CountVectorizer(ngram_range=(1,2),max_features=5000,min_df=10)
vectorizer.fit(X_train.project_resource_summary.values)
X_tr_resource=vectorizer.transform(X_train.project_resource_summary.values)
X_te_resource=vectorizer.transform(X_test.project_resource_summary.values)
```

In [55]:

```
X_train_bow=hstack((X_tr_vec,X_tr_title,X_tr_resource,X_tr_essay)).tocsr()
X_test_bow=hstack((X_te_vec,X_te_title,X_te_resource,X_te_essay)).tocsr()
```

In [56]:

```
print("Bag of words:")
print("Training data set shape :",X_train_bow.shape)
print("Test data set shape :",X_test_bow.shape)
```

```
Bag of words:
Training data set shape : (76473, 14853)
Test data set shape : (32775, 14853)
```

In [57]:

```
# Release the memory
del X_tr_title
del X_te_title

del X_tr_resource
```

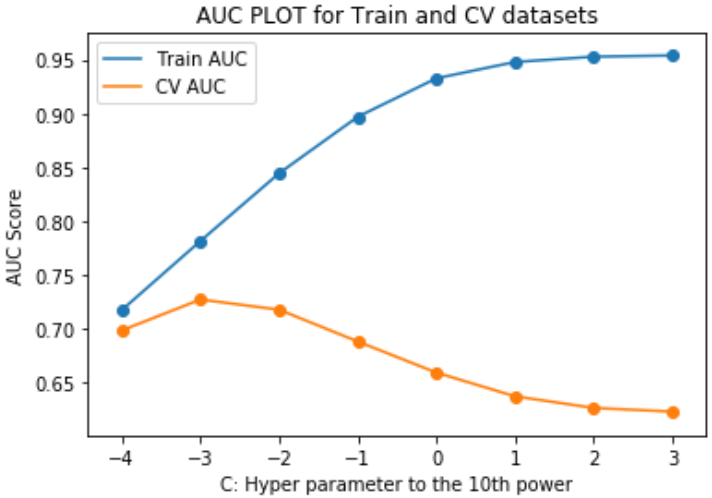
```
del X_te_resource

del X_tr_essay
del X_te_essay
```

Finding best 'C' and build the Model

In [58]:

```
cross_validate_plot(c_values,X_train_bow,y_train)
```

```
100%|██████████| 8/8 [55:45<00:00, 418.23s/it]
```



In [58]:

```
print("'C' Values to the power  of 10:",np.log10(c_values))
```

```
'C' Values to the power  of 10: [-4. -3. -2. -1.  0.  1.  2.  3.]
```

In [59]:

```
best_model=LogisticRegression()
best_model.class_weight='balanced'
print(" Best Model Default Params:\n",best_model)
```

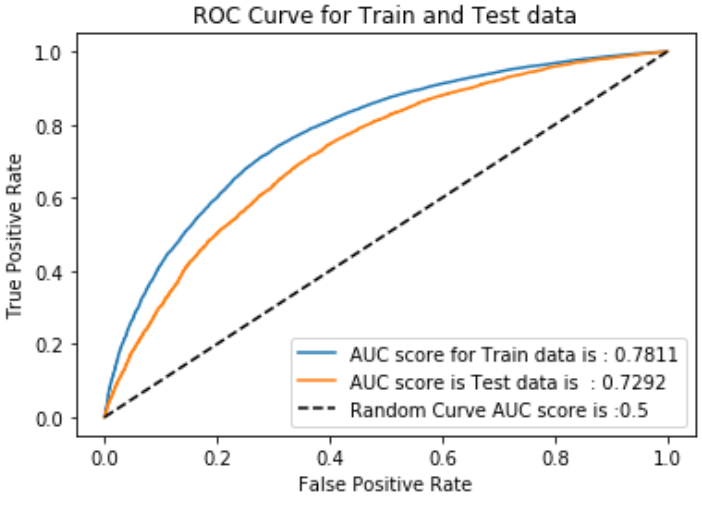
```
 Best Model Default Params:
 LogisticRegression(C=1.0, class_weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

In [78]:

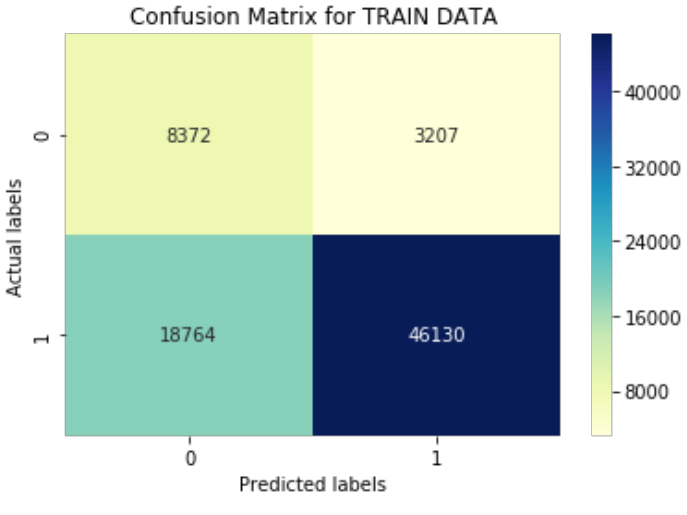
```
best_model.C=10**-3
```

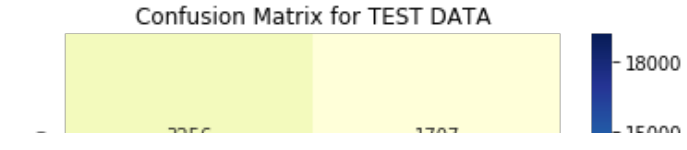
In [79]:

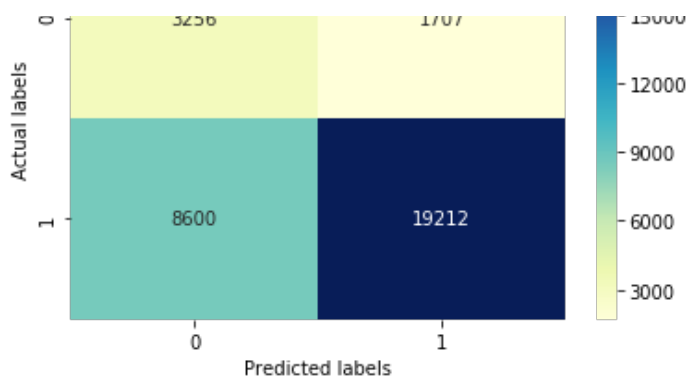
```
build_best_model_plot_roc(best_model,X_train_bow,y_train,X_test_bow,y_test)
```



The Maximum value of 'TPR*(1-FPR)' is 0.5139691781621604 for 'THRESHOLD VALUE'of 0.496








```
+--------------------+---------------+
| Training Accuracy | Test Accuracy |
+--------------------+---------------+
|       0.713        |     0.686     |
+--------------------+---------------+
```

In [80]:

```
del X_train_bow
del X_test_bow
```

TF-IDF

project_title

In [60]:

```
tf_idf_vectorizer=TfidfVectorizer(ngram_range=(1,2),max_features=5000,min_df=10)
tf_idf_vectorizer.fit(X_train.project_title.values)
X_tr_title=tf_idf_vectorizer.transform(X_train.project_title.values)
X_te_title=tf_idf_vectorizer.transform(X_test.project_title.values)
```

essay

In [61]:

```
tf_idf_vectorizer=TfidfVectorizer(ngram_range=(1,2),max_features=5000,min_df=10)
tf_idf_vectorizer.fit(X_train.essay.values)
X_tr_essay=tf_idf_vectorizer.transform(X_train.essay.values)
X_te_essay=tf_idf_vectorizer.transform(X_test.essay.values)
```

project_resource_summary

project_resource_summary

In [62]:

```
tf_idf_vectorizer=TfidfVectorizer(ngram_range=(1,2),max_features=5000,min_df=10)
tf_idf_vectorizer.fit(X_train.project_resource_summary.values)
X_tr_resource=tf_idf_vectorizer.transform(X_train.project_resource_summary.values)
X_te_resource=tf_idf_vectorizer.transform(X_test.project_resource_summary.values)
```

In [63]:

```
X_train_tfidf=hstack((X_tr_vec,X_tr_title,X_tr_resource,X_tr_essay)).tocsr()
X_test_tfidf=hstack((X_te_vec,X_te_title,X_te_resource,X_te_essay)).tocsr()
```

In [64]:

```
print("TF-IDF:")
print("Training data set shape :",X_train_tfidf.shape)
print("Test data set shape :",X_test_tfidf.shape)
```

```
TF-IDF:
Training data set shape : (76473, 14853)
Test data set shape : (32775, 14853)
```

In [65]:

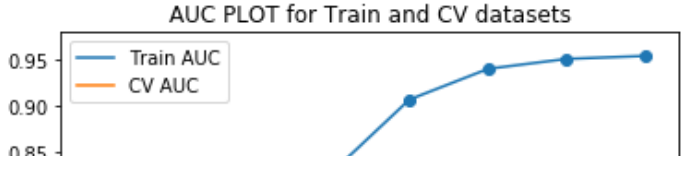
```
# Release the memory
del X_tr_title
del X_te_title

del X_tr_resource
del X_te_resource

del X_tr_essay
del X_te_essay
```

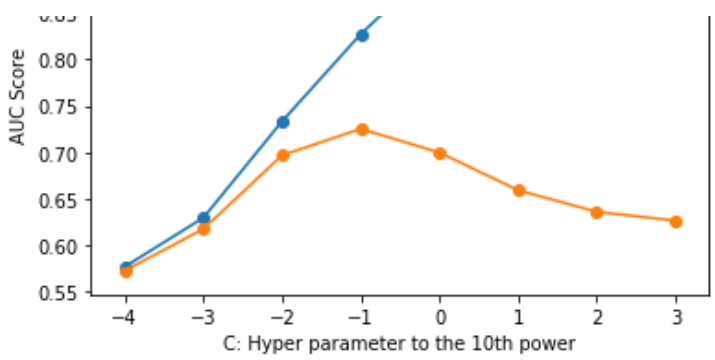
Finding right "C" and build model

In [67]:

```
cross_validate_plot(c_values,X_train_tfidf,y_train)
```

```
100%|██████████| 8/8 [18:35<00:00, 139.45s/it]
```





In [66]:

```
print("Hyper parameters to the power 10  :",np.log10(c_values))
```

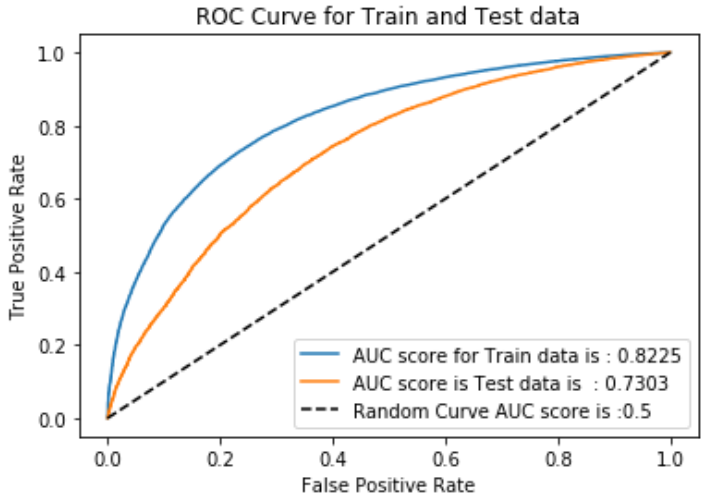
```
Hyper parameters to the power 10  : [-4. -3. -2. -1.  0.  1.  2.  3.]
```

In [81]:

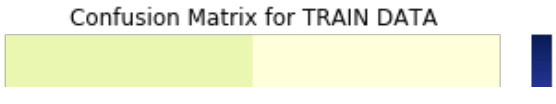
```
best_model.C=10**-1
```

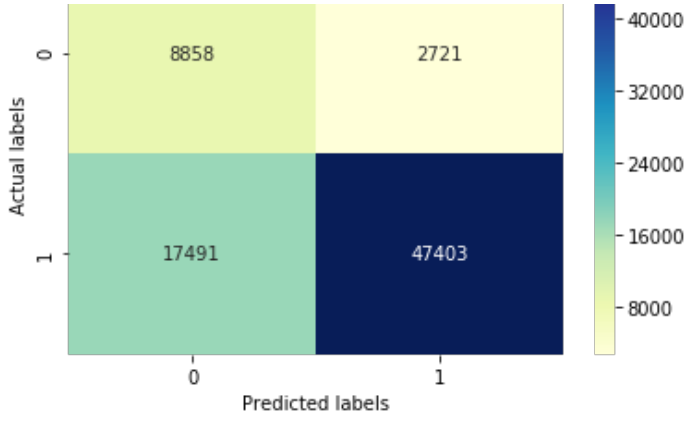
In [82]:

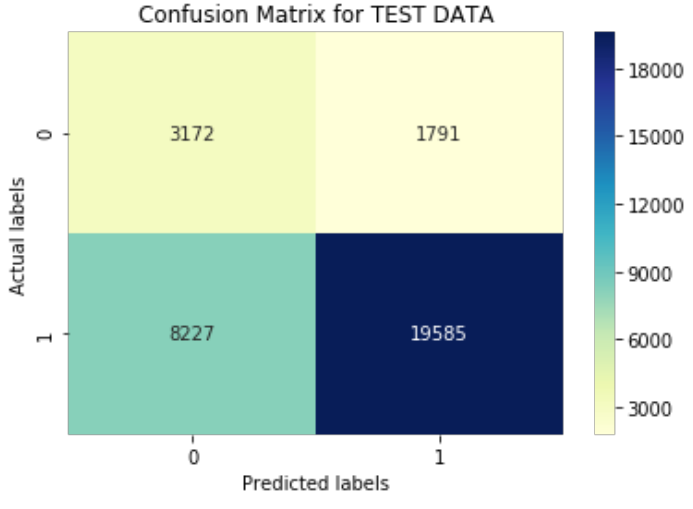
```
build_best_model_plot_roc(best_model,X_train_tfidf,y_train,X_test_tfidf,y_test)
```




```
----------------------------------------------------------------------------------------------------
The Maximum value of 'TPR*(1-FPR)' is 0.5588122338273953 for 'THRESHOLD VALUE'of 0.511
----------------------------------------------------------------------------------------------------
```








```
+--------------------+---------------+
| Training Accuracy | Test Accuracy |
+--------------------+---------------+
|       0.736        |     0.694     |
+--------------------+---------------+
```

In [83]:

```
del X_train_tfidf
del X_test_tfidf
```

Avg W2V

In [84]:

```
# stronging variables into pickle files python:
```

```
#http://www.jessicayung.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 [85]:

```
# average Word2Vec
def avg_w2vec(glove_words,feature_values):
    # compute average word2vec for each review.
    avg_w2v_vec = []; # the avg-w2v for each sentence/review is stored in this list

    for sent in tqdm(feature_values): # for each review/sentence
        sent_vec = np.zeros(300) # as word vectors are of zero length 300, you might need to
        #change this to 300 if you use google's w2v
        cnt_words =0; # num of words with a valid vector in the sentence/review
        for word in sent.split(): # for each word in a review/sentence
            if word in glove_words:
                sent_vec += model[word]
                cnt_words += 1
        if cnt_words != 0:
            sent_vec /= cnt_words
        avg_w2v_vec.append(sent_vec)

    print(len(avg_w2v_vec))
    print(len(avg_w2v_vec[0]))

    return avg_w2v_vec
```

project_title

In [86]:

```
X_tr_title=avg_w2vec(glove_words,X_train.project_title.values)
```

```
100%|██████████| 76473/76473 [00:01<00:00, 66232.80it/s]

76473
300
```

In [87]:

```
X_te_title=avg_w2vec(glove_words,X_test.project_title.values)
```

```
100%|██████████| 32775/32775 [00:00<00:00, 66271.54it/s]

32775
300
```

300

essay

In [88]:

```
X_tr_essay=avg_w2vec(glove_words,X_train.essay.values)
```

```
100%|██████████| 76473/76473 [00:22<00:00, 3458.36it/s]
```

```
76473
300
```

In [89]:

```
X_te_essay=avg_w2vec(glove_words,X_test.essay.values)
```

```
100%|██████████| 32775/32775 [00:09<00:00, 3415.11it/s]
```

```
32775
300
```

project_resource_summary

In [90]:

```
X_tr_resource=avg_w2vec(glove_words,X_train.project_resource_summary.values)
```

```
100%|██████████| 76473/76473 [00:03<00:00, 24765.11it/s]
```

```
76473
300
```

In [91]:

```
X_te_resource=avg_w2vec(glove_words,X_test.project_resource_summary.values)
```

```
100%|██████████| 32775/32775 [00:01<00:00, 25233.30it/s]
```

```
32775
300
```

In [92]:

```
X_train_awv=hstack((X_tr_vec,X_tr_title,X_tr_essay,X_tr_resource)).tocsr()
```

```
X_test_awv=hstack((X_te_vec,X_te_title,X_te_essay,X_te_resource)).tocsr()
```

In [93]:

```
print("Average Word 2 vector:")
print("Training data set shape :",X_train_awv.shape)
print("Test data set shape :",X_test_awv.shape)
```

```
Average Word 2 vector:
Training data set shape : (76473, 1001)
Test data set shape : (32775, 1001)
```

In [94]:

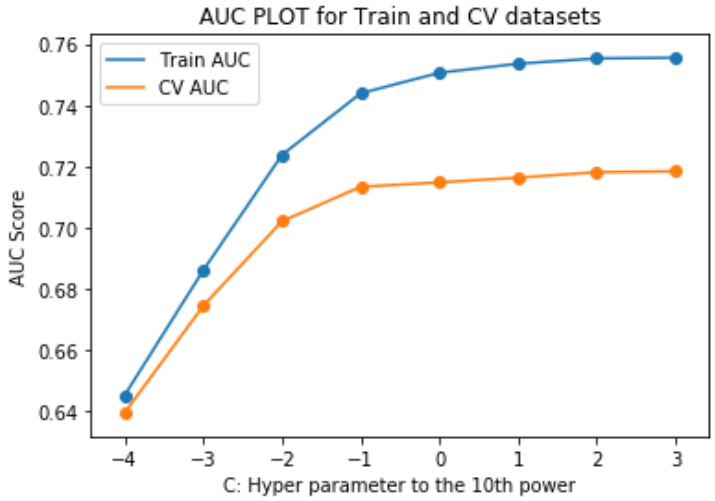
```
# Release the memory
del X_tr_title
del X_te_title

del X_tr_resource
del X_te_resource

del X_tr_essay
del X_te_essay
```

Find the right 'C' and build the Classifier

In [95]:

```
cross_validate_plot(c_values,X_train_awv,y_train)
```

```
100%|██████████| 8/8 [32:48<00:00, 246.09s/it]
```



In [96]:

```
print("'C' Values to the power  of 10:",np.log10(c_values))
```

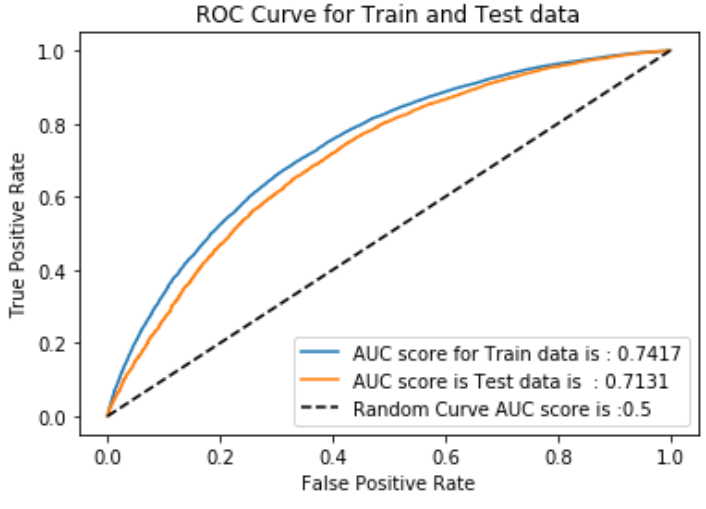
```
'C' Values to the power  of 10: [-4. -3. -2. -1.  0.  1.  2.  3.]
```

In [115]:

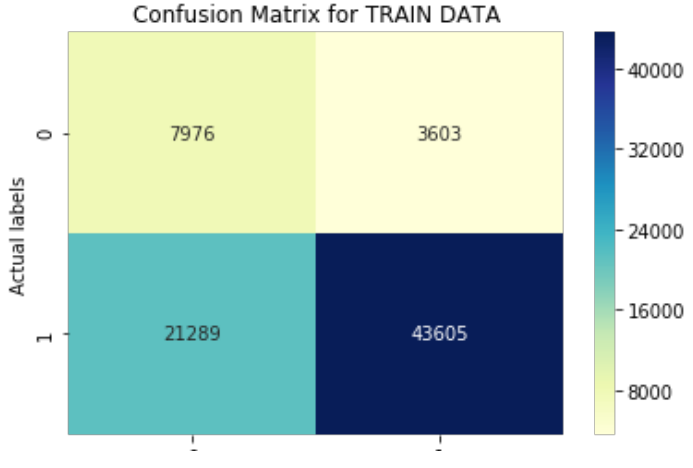
```
best_model.C=10**-1
```

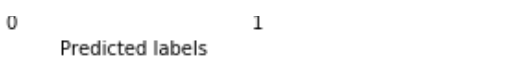
In [116]:

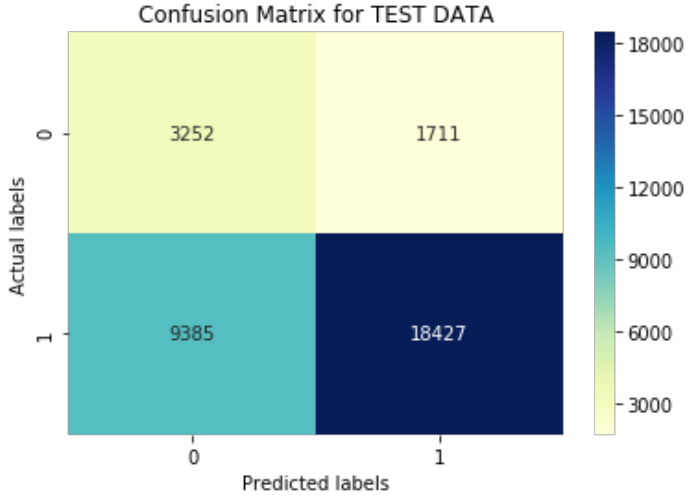
```
build_best_model_plot_roc(best_model,X_train_awv,y_train,X_test_awv,y_test)
```




```
---------------------------------------------------------------------------------------------------
The Maximum value of 'TPR*(1-FPR)' is 0.46285593593376706 for 'THRESHOLD VALUE'of 0.505
---------------------------------------------------------------------------------------------------
```








```
+--------------------+---------------+
| Training Accuracy | Test Accuracy |
+--------------------+---------------+
|       0.674        |     0.661     |
+--------------------+---------------+
```

In [117]:

```
del X_train_awv
del X_test_awv
```

TF-IDF AW2V

In [97]:

```
def tfidf_avgw2v(glove_words,tfidf_words,feature_values):
    processed_tfidf_w2v= []; # the avg-w2v for TITLE is stored in this list
    for sentence in tqdm(feature_values): # for each TITLE
        vector = np.zeros(300) # as word vectors are of zero length
        tf_idf_weight =0; # num of words with a valid vector in the TITLE
        for word in sentence.split(): # for each word in a review/sentence
            if (word in glove_words) and (word in tfidf_words):
                vec = model[word] # getting the vector for each word
                # here we are multiplying idf value(dictionary[word]) and
                #the tf value((sentence.count(word)/len(sentence.split())))
                tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
                # getting the tfidf value for each word
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
```

```
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        processed_tfidf_w2v.append(vector)

    print(len(processed_tfidf_w2v))
    print(len(processed_tfidf_w2v[0]))

    return processed_tfidf_w2v
```

essay

In [98]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train.essay.values)
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [99]:

```
X_tr_essay= tfidf_avgw2v(glove_words,tfidf_words,X_train.essay.values)
```

```
100%|██████████| 76473/76473 [02:30<00:00, 509.60it/s]

76473
300
```

In [100]:

```
X_te_essay= tfidf_avgw2v(glove_words,tfidf_words,X_test.essay.values)
```

```
100%|██████████| 32775/32775 [01:00<00:00, 537.75it/s]

32775
300
```

project_title

In [101]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train.project_title.values)
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [102]:

In [102]:

```
X_tr_title=tfidf_avgw2v(glove_words,tfidf_words,X_train.project_title.values)
```

```
100%|██████████| 76473/76473 [00:02<00:00, 28644.64it/s]
```

```
76473
300
```

In [103]:

```
X_te_title=tfidf_avgw2v(glove_words,tfidf_words,X_test.project_title.values)
```

```
100%|██████████| 32775/32775 [00:01<00:00, 29168.83it/s]
```

```
32775
300
```

project_resource_summary

In [104]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train.project_resource_summary.values)
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [105]:

```
X_tr_resource=tfidf_avgw2v(glove_words,tfidf_words,X_train.project_resource_summary.values)
```

```
100%|██████████| 76473/76473 [00:08<00:00, 9481.38it/s]
```

```
76473
300
```

In [106]:

```
X_te_resource=tfidf_avgw2v(glove_words,tfidf_words,X_test.project_resource_summary.values)
```

```
100%|██████████| 32775/32775 [00:03<00:00, 9585.32it/s]
```

```
32775
300
```

In [107]:

```
X_train_tfidfavgw=hstack((X_tr_ess,X_tr_title,X_tr_essay,X_tr_resource)).tocsr()
```

```
X_test_tfidfawv=hstack((X_te_vec,X_te_title,X_te_essay,X_te_resource)).tocsr()
```

In [108]:

```
print("Average Word 2 vector:")
print("Training data set shape :",X_train_tfidfawv.shape)
print("Test data set shape :",X_test_tfidfawv.shape)
```

```
Average Word 2 vector:
Training data set shape : (76473, 1001)
Test data set shape : (32775, 1001)
```

In [109]:

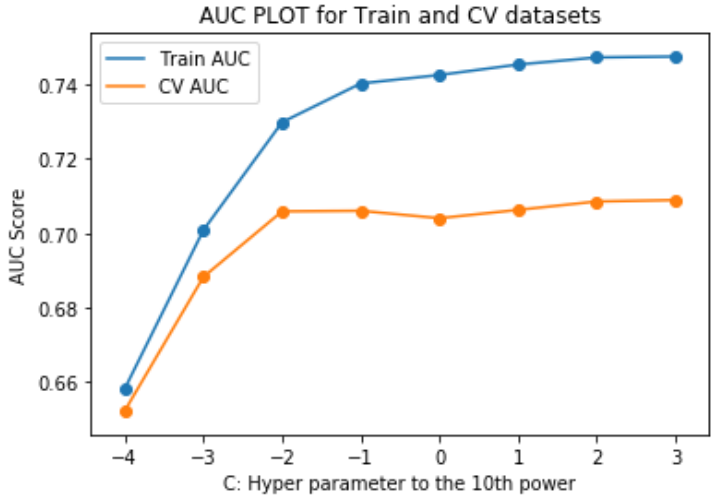
```
# Release the memory
del X_tr_title
del X_te_title

del X_tr_resource
del X_te_resource

del X_tr_essay
del X_te_essay
```

Find the right 'C' and build the Classifier

In [110]:

```
cross_validate_plot(c_values,X_train_tfidfawv,y_train)
```

```
100%|██████████| 8/8 [36:56<00:00, 277.06s/it]
```



In [111]:

```
print("'C' Values to the power  of 10:",np.log10(c_values))
```

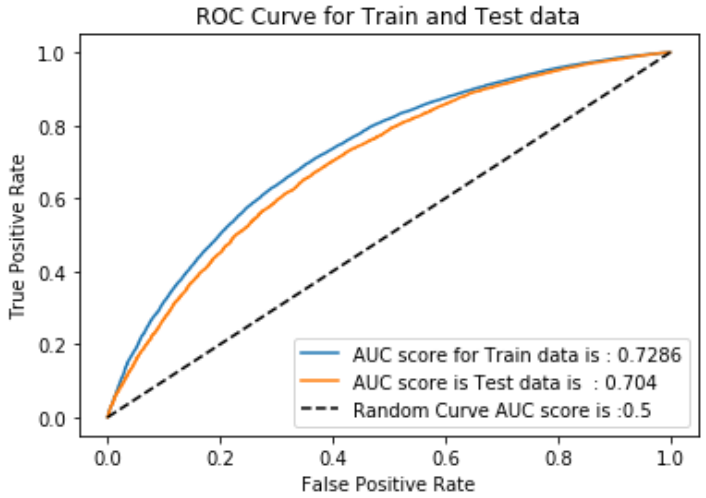
```
'C' Values to the power  of 10: [-4. -3. -2. -1.  0.  1.  2.  3.]
```

In [112]:

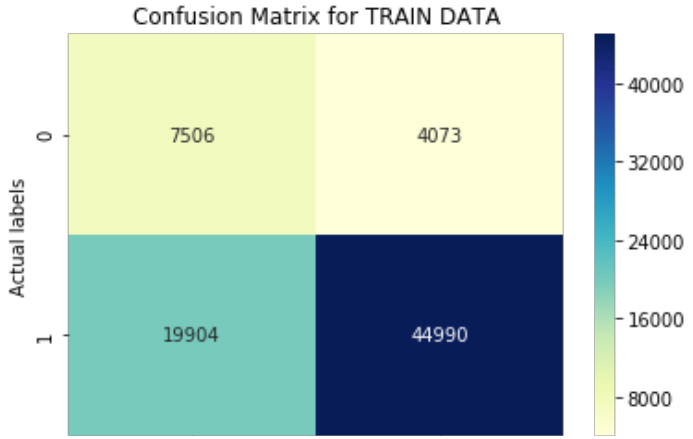
```
best_model.C=10**-2
```

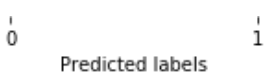
In [113]:

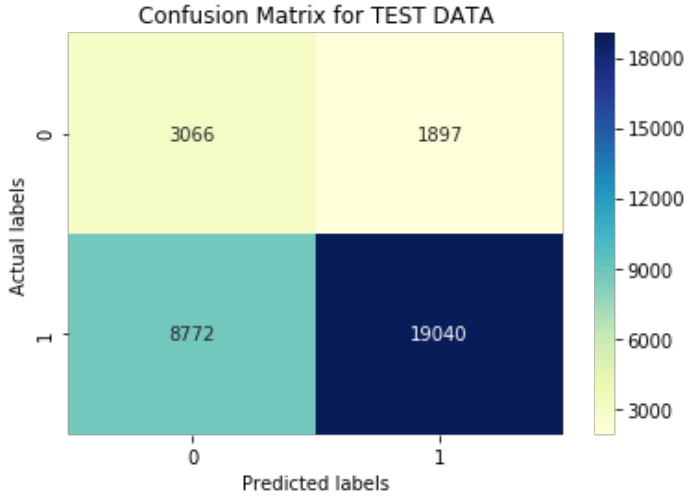
```
build_best_model_plot_roc(best_model,X_train_tfidfawv,y_train,X_test_tfidfawv,y_test)
```




```
----------------------------------------------------------------------------------------------------
The Maximum value of 'TPR*(1-FPR)' is 0.4494164396463019 for 'THRESHOLD VALUE'of 0.483
----------------------------------------------------------------------------------------------------
```








```
+--------------------+---------------+
| Training Accuracy | Test Accuracy |
+--------------------+---------------+
|       0.686       |     0.674     |
+--------------------+---------------+
```

In [114]:

```
del X_train_tfidfawv
del X_test_tfidfawv
```

Task-2

Dealing without Text data

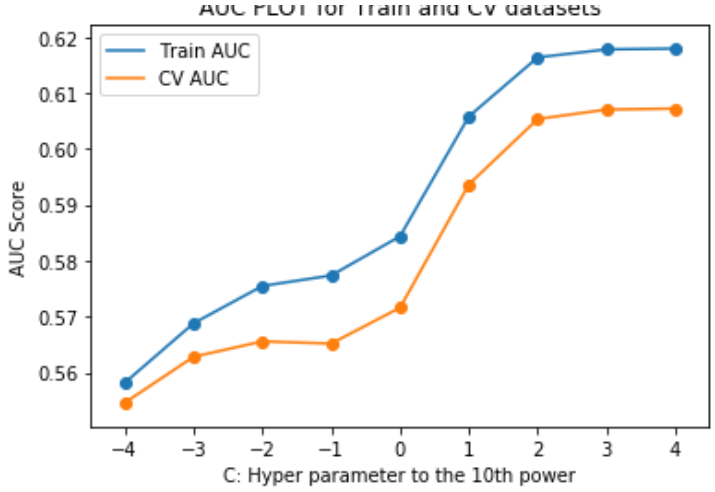
In [118]:

```
c_values_no_text=[10**i  for i in range(-4,5)]
#to getclear understanding of overfiting added one more value to c_values
```

In [119]:

```
cross_validate_plot(c_values_no_text,X_tr_vec,y_train)
```

```
100%|██████████| 9/9 [01:12<00:00,  8.04s/it]
```

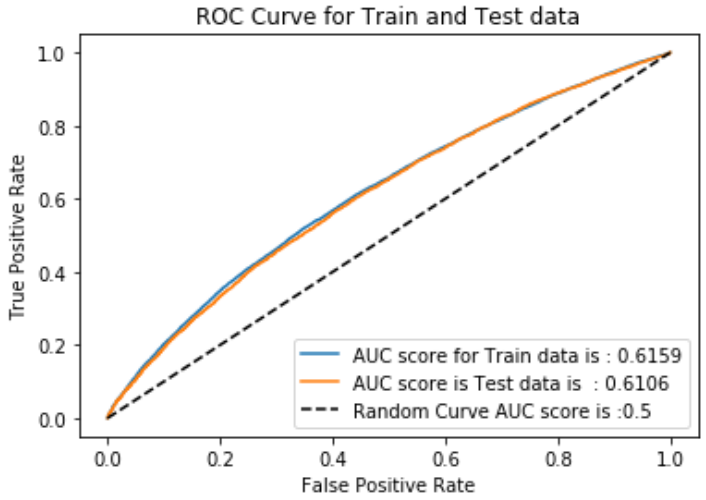


In [120]:

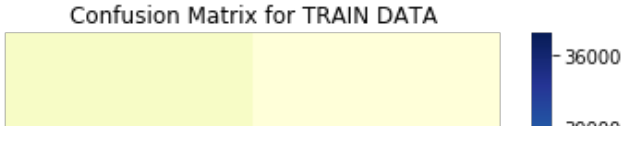
```
best_model.C=10**2
```

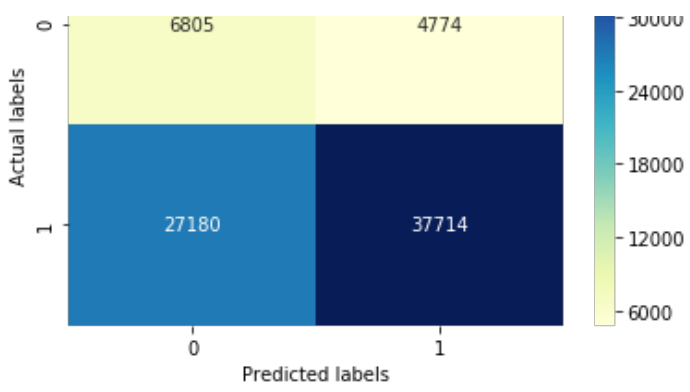
In [121]:

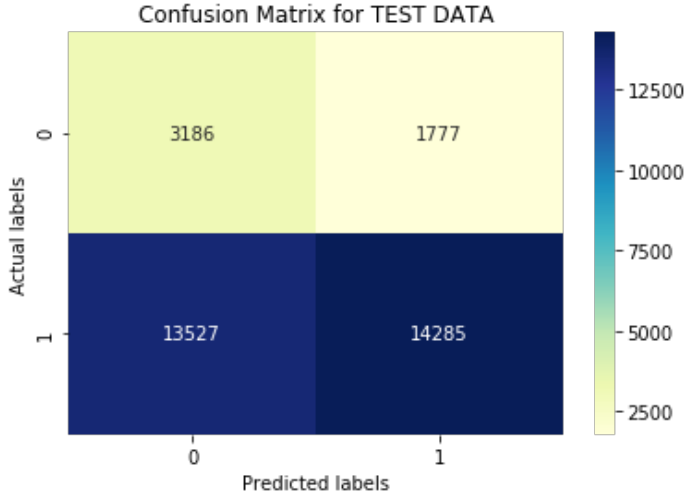
```
build_best_model_plot_roc(best_model,X_tr_vec,y_train,X_te_vec,y_test)
```




```
--------------------------------------------------------------------------------
The Maximum value of 'TPR*(1-FPR)' is 0.34155065921569444 for 'THRESHOLD VALUE'of 0.5
--------------------------------------------------------------------------------
```








```
+--------------------+---------------+
| Training Accuracy | Test Accuracy |
+--------------------+---------------+
|       0.582       |     0.533     |
+--------------------+---------------+
```

Summary

In [124]:

```
summary_table = PrettyTable()
summary_table.hrules=True

summary_table.field_names=['Model',"Vectorizer", "Method for CV","Opt Parm","Train AUC", "Test AUC",'Accuracy(Train & Test)']
summary_table.add_row(['Logistic',"BOW", 'Cross_validate', 'L2 & C:.001', .7811,.7292,'0.713 & 0.686'])
summary_table.add_row(['Logistic',"TF-IDF", 'Cross_validate', 'L2 & C:0.1', .8225,.7303,'0.736 & 0.694'])
summary_table.add_row(['Logistic',"AVG W2V", 'Cross_validate', 'L2 & C:0.1', .7417,.7131,'0.674 & 0.661'])
summary_table.add_row(['Logistic',"TF-IDF AVGW2V", 'Cross_validate','L2 & C:0.01', .7286,.704,'0.686 & 0.674'])
```

```
summary_table.add_row(['Logistic',"-", 'Cross_validate','L2 & C:100', .6159,.6106,'0.582 & 0.533'])

summary_table.sortby='Test AUC'
summary_table.reversesort=True
print(summary_table)
```

```
+----------+---------------+----------------+-------------+-----------+----------+-------------------------+
|  Model   |   Vectorizer  | Method for CV  |   Opt Parm  | Train AUC | Test AUC | Accuracy(Train & Test)  |
+----------+---------------+----------------+-------------+-----------+----------+-------------------------+
| Logistic |     TF-IDF    | Cross_validate |  L2 & C:0.1 |   0.8225  |  0.7303  |     0.736 & 0.694       |
+----------+---------------+----------------+-------------+-----------+----------+-------------------------+
| Logistic |      BOW      | Cross_validate | L2 & C:.001 |   0.7811  |  0.7292  |     0.713 & 0.686       |
+----------+---------------+----------------+-------------+-----------+----------+-------------------------+
| Logistic |    AVG W2V    | Cross_validate |  L2 & C:0.1 |   0.7417  |  0.7131  |     0.674 & 0.661       |
+----------+---------------+----------------+-------------+-----------+----------+-------------------------+
| Logistic | TF-IDF AVGW2V | Cross_validate | L2 & C:0.01 |   0.7286  |  0.704   |     0.686 & 0.674       |
+----------+---------------+----------------+-------------+-----------+----------+-------------------------+
| Logistic |       -       | Cross_validate |  L2 & C:100 |   0.6159  |  0.6106  |     0.582 & 0.533       |
+----------+---------------+----------------+-------------+-----------+----------+-------------------------+
```

In [ ]:

```
```