Naive Bayes On Donors Choose Dataset

In [1]:

```
# Importing all the necessary libraries and packages

%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
import re
import string
import pickle
import os
import math

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

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


from tqdm import tqdm
from chart_studio import plotly  #Importing plotly from chart_studio as plotly is deprecated
according to jupyter
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

Reading Data

In [2]:

```
# Importing data with pandas
# For avoid memory issues and to reduce run time I'm only taking 70k points

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

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

```
Number of data points in train data (109248, 17)

 --------------------------------------------------

The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' '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 [3]:

```
# 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()
```

Out[3]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Grades PreK-2	I
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Grades PreK-2	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5	I

In [4]:

```
# Printing total no. of data points in Resource Data and the features it have.

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

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

Out[4]:

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

Preprocessing project_subject_categories

In [5]:

```
# 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

catogories = list(project_data['project_subject_categories'].values)
cat_list = []
for i in catogories:
    temp = ""
    for j in i.split(','):
        if 'The' in j.split():
            j=j.replace('The','')
        j = j.replace(' ','')
        temp+=j.strip()+" "
        temp = temp.replace('&','_')
    cat_list.append(temp.strip())

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

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

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

Preprocessing project_subject_subcategories

In [6]:

```
# 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_catogories = list(project_data['project_subject_subcategories'].values)
sub_cat_list = []
for i in sub_catogories:
    temp = ""
    for j in i.split(','):
        if 'The' in j.split():
            j=j.replace('The','')
        j = j.replace(' ','')
        temp +=j.strip()+" "
        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)

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

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

Preprocessing of Project_grade_category

In [7]:

```
# Preprocessing Project_grade_category
# Removing Special characters and 'Grade' word to make this category ready for the vectorization

sub_grade = list(project_data['project_grade_category'].values)
```

```
grade_cat_list = []
for i in sub_grade:
    for j in i.split(' '):
        j=j.replace('Grades','')
        j=j.replace('-', '_')
    grade_cat_list.append(j.lower().strip())

project_data['clean_grade_category'] = grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)

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

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

In [8]:

```
# Printing top values to see the changes and our updated data

project_data.head()
```

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	project_essay_1	pr
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	I have been fortunate enough to use the Fairy ...	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagine being 8-9 years old. You're in your th...	
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Mobile Learning with a Mobile Listening Center	Having a class of 24 students comes with diver...	I
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	I
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	W

Merging Project_essay

In [9]:

```
# Merging all the subcategory of project_data into one category

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)

# Printing top values to see the updated Data
project_data.head()
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	project_essay_1	pr
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27	Engineering STEAM into	I have been fortunate enough	

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	00:27:36 Date	the Primary Classroom project_title	to use the Fairy project_essay_1	pr
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagine being 8-9 years old. You're in your th...	
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Mobile Learning with a Mobile Listening Center	Having a class of 24 students comes with diver...	I
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	I
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	W

In [10]:

```
# Merging price from resource_data to project_data before splitiing the data

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')

# 'techer_prefix' has some missing values so we're filling it with the most common value which is
'Mrs.'
project_data["teacher_prefix"].fillna("Mrs.", inplace= True)
```

Splitting Data in train, CV and test data

In [11]:

```
# I have divided my train, cv and test in 60:25:20
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data,
project_data['project_is_approved'], test_size=0.25, stratify=project_data['project_is_approved'])
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.25, stratify=y_train)
```

In [12]:

```
# Printing no. of total values my Train, Cv and Test data have

print(y_train.value_counts())
print(y_cv.value_counts())
print(y_test.value_counts())
```

```
1    52148
0     9304
Name: project_is_approved, dtype: int64
1    17382
0     3102
Name: project_is_approved, dtype: int64
1    23176
0     4136
Name: project_is_approved, dtype: int64
```

Observations

- As we can see that we have an imbalance dataset and that leads to the failure of Naive Bayes

- So to avoid this problem we need to perform upsampling

Upsampling the data

In [13]:

```
# Dividing data into majority and minority so that we can upsample minority class

majority_data = X_train[X_train.project_is_approved==1]
minority_data = X_train[X_train.project_is_approved==0]
```

In [14]:

```
from sklearn.utils import resample

minority_data_upsampled = resample(minority_data, replace=True, n_samples=52147, random_state=10)
x_train_upsampled = pd.concat([majority_data, minority_data_upsampled])

# After applying Upsampling checking and printing total no. of datapoints for each class (i.e 0 an
d 1 class)
x_train_upsampled.project_is_approved.value_counts()
```

Out[14]:

```
1    52148
0    52147
Name: project_is_approved, dtype: int64
```

In [15]:

```
# Updating y_train according to the upsampled data

y_train_upsampled = x_train_upsampled.project_is_approved
print(y_train_upsampled.value_counts())
```

```
1    52148
0    52147
Name: project_is_approved, dtype: int64
```

In [16]:

```
# Dropping 'project_is_approved' column form train, cv and test data

X_train.drop(["project_is_approved"], axis = 1, inplace = True)
x_train_upsampled.drop(["project_is_approved"], axis = 1, inplace = True)
X_test.drop(["project_is_approved"], axis = 1, inplace = True)
X_cv.drop(["project_is_approved"], axis = 1, inplace = True)
```

Preparing Data for model

In [17]:

```
# Printing All the features after preprocessing data

project_data.columns
```

Out[17]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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', 'clean_grade_category',
       'essay', 'price', 'quantity'],
      dtype='object')
```

Text preprocessing for Train, CV and Train Data

Preprocessing of Project_essay

In [18]:

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

def decontracted(phrase):
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    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 [19]:

```
# https://gist.github.com/sebleier/554280

stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [20]:

```
# Preprocessing Project_essay on Train_data

train_preprocessed_essays = []

for sentance in tqdm(x_train_upsampled['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)
    train_preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 104295/104295
```

```
[01:14<00:00, 1395.43it/s]
```

In [21]:

```
# Preprocessing Project_essay on CV data

cv_preprocessed_essays = []

for sentance in tqdm(X_cv['essay'].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)
    cv_preprocessed_essays.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████| 20484/20484
[00:13<00:00, 1486.56it/s]
```

In [22]:

```
# Preprocessing Project_essay on Test data

test_preprocessed_essays = []

for sentance in tqdm(X_test['essay'].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)
    test_preprocessed_essays.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████| 27312/27312
[00:18<00:00, 1439.93it/s]
```

Preprocessing Project_title

In [23]:

```
def decontracted2(phrase):
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\s'", "s", phrase)
    return phrase
```

In [24]:

```
# Preprocessing Project_title on Train_data

train_preprocessed_title = []
for title in tqdm(x_train_upsampled['project_title'].values):
    sent = decontracted2(title)
    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 not in stopwords)
    train_preprocessed_title.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████| 104295/104295
[00:02<00:00, 44401.41it/s]
```

In [25]:

```
# Preprocessing Project_title on CV_data
```

```
cv_preprocessed_title = []
for title in tqdm(X_cv['project_title'].values):
    sent = decontracted2(title)
    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 not in stopwords)
    cv_preprocessed_title.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████| 20484/20484
[00:00<00:00, 44357.02it/s]
```

In [26]:

```
# Preprocessing Project_title on Test_data

test_preprocessed_title = []
for title in tqdm(X_test['project_title'].values):
    sent = decontracted2(title)
    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 not in stopwords)
    test_preprocessed_title.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████| 27312/27312
[00:00<00:00, 43058.30it/s]
```

Vectorizing Categorical Data

One-hot encoding on clean_categories

In [27]:

```
# Performing one-hot encoding on clean_categories for Train, CV and Test Data
Feature_name_BoW = []

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
)
vectorizer.fit(x_train_upsampled['clean_categories'].values)

X_train_cat_onehot = vectorizer.transform(x_train_upsampled['clean_categories'].values)
X_cv_cat_onehot = vectorizer.transform(X_cv['clean_categories'].values)
X_test_cat_onehot = vectorizer.transform(X_test['clean_categories'].values)

print(vectorizer.get_feature_names())

Feature_name_BoW.extend(vectorizer.get_feature_names())
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
```

In [28]:

```
print("Printing shape of Train, CV and Test data after vectorizing clean_categories")

print(X_train_cat_onehot.shape, y_train_upsampled.shape)
print(X_cv_cat_onehot.shape, y_cv.shape)
print(X_test_cat_onehot.shape, y_test.shape)
```

```
Printing shape of Train, CV and Test data after vectorizing clean_categories
(104295, 9) (104295,)
(20484, 9) (20484,)
(27312, 9) (27312,)
```

one-hot encoding on clean_sub_categories

In [29]:

```
# Performing one-hot encoding on clean_sub_categories for Train, CV and Test Data

vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=
True)
vectorizer.fit(x_train_upsampled['clean_subcategories'].values)

X_train_subcat_onehot = vectorizer.transform(x_train_upsampled['clean_subcategories'].values)
X_cv_subcat_onehot = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_subcat_onehot = vectorizer.transform(X_test['clean_subcategories'].values)
print(vectorizer.get_feature_names())

Feature_name_BoW.extend(vectorizer.get_feature_names())
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
```

In [30]:

```
print("Printing shape of Train, CV and Test data after vectorizing clean_subcategories")

print(X_train_subcat_onehot.shape, y_train_upsampled.shape)
print(X_cv_subcat_onehot.shape, y_cv.shape)
print(X_test_subcat_onehot.shape, y_test.shape)
```

```
Printing shape of Train, CV and Test data after vectorizing clean_subcategories
(104295, 30) (104295,)
(20484, 30) (20484,)
(27312, 30) (27312,)
```

One-hot encoding on school_state

In [31]:

```
# Performing one-hot encoding on school_state for Train, CV and Test Data

vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(x_train_upsampled['school_state'].values)

X_train_school_state_onehot = vectorizer.transform(x_train_upsampled['clean_subcategories'].values
)
X_cv_school_state_onehot = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_school_state_onehot = vectorizer.transform(X_test['clean_subcategories'].values)
print(vectorizer.get_feature_names())

Feature_name_BoW.extend(vectorizer.get_feature_names())
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
```

In [32]:

```
print("Printing shape of Train, CV and Test data after vectorizing school_state")

print(X_train_school_state_onehot.shape, y_train_upsampled.shape)
print(X_cv_school_state_onehot.shape, y_cv.shape)
print(X_test_school_state_onehot.shape, y_test.shape)
```

```
Printing shape of Train, CV and Test data after vectorizing school_state
```

```
Printing shape of Train, CV and Test data after vectorizing school_state
(104295, 51) (104295,)
(20484, 51) (20484,)
(27312, 51) (27312,)
```

one-hot encoding on teacher_prefix

In [33]:

```
# Performing one-hot encoding on teacher_prefix for Train, CV and Test Data

x_train_upsampled["teacher_prefix"].fillna("Mrs.", inplace= True)

vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(x_train_upsampled['teacher_prefix'].values)

X_train_teacher_onehot = vectorizer.transform(x_train_upsampled['teacher_prefix'].values)
X_cv_teacher_onehot = vectorizer.transform(X_cv['teacher_prefix'].values)
X_test_teacher_onehot = vectorizer.transform(X_test['teacher_prefix'].values)

Feature_name_BoW.extend(vectorizer.get_feature_names())
```

In [34]:

```
print("Printing shape of Train, CV and Test data after vectorizing teacher_prefix")

print(X_train_teacher_onehot.shape, y_train_upsampled.shape)
print(X_cv_teacher_onehot.shape, y_cv.shape)
print(X_test_teacher_onehot.shape, y_test.shape)
```

```
Printing shape of Train, CV and Test data after vectorizing teacher_prefix
(104295, 5) (104295,)
(20484, 5) (20484,)
(27312, 5) (27312,)
```

one-hot encoding on clean_grade_category

In [35]:

```
# Performing one-hot encoding on clean_grade_category for Train, CV and Test Data

vectorizer = CountVectorizer(vocabulary=list(sorted_sub_grade_cat_dict.keys()), lowercase=False, b
inary=True)
vectorizer.fit(x_train_upsampled['clean_grade_category'].values)

X_train_clean_grade_onehot = vectorizer.transform(x_train_upsampled['clean_grade_category'].values
)
X_cv_clean_grade_onehot = vectorizer.transform(X_cv['clean_grade_category'].values)
X_test_clean_grade_onehot = vectorizer.transform(X_test['clean_grade_category'].values)
print(vectorizer.get_feature_names())

Feature_name_BoW.extend(vectorizer.get_feature_names())
```

```
['9_12', '6_8', '3_5', 'prek_2']
```

In [36]:

```
print("Printing shape of Train, CV and Test data after vectorizing clean_grade_category")

print(X_train_clean_grade_onehot.shape, y_train_upsampled.shape)
print(X_cv_clean_grade_onehot.shape, y_cv.shape)
print(X_test_clean_grade_onehot.shape, y_test.shape)
```

```
Printing shape of Train, CV and Test data after vectorizing clean_grade_category
(104295, 4) (104295,)
(20484, 4) (20484,)
(27312, 4) (27312,)
```

Vectorizing Numerical Features

In [37]:

```
# Vectorizing Price Feature
# We can not use MinMaxScaler since it will also give us negative values which will cause error wh
ile working with
# Multinimial NB since it requires non-negative values only
# I am using MinMaxScaler instead of Normalization because it's non-distorting, Beside that
Normalizer works on the rows
# not the columns so I am using MinMaxScaler

from sklearn.preprocessing import MinMaxScaler

price_scalar = MinMaxScaler()
price_scalar.fit(x_train_upsampled['price'].values.reshape(-1,1))

X_train_price_std = price_scalar.transform(x_train_upsampled['price'].values.reshape(-1,1))
X_cv_price_std = price_scalar.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_std = price_scalar.transform(X_test['price'].values.reshape(-1,1))

# Printing train data after applying minmaxscaler to see the changes
print(X_train_price_std)

Feature_name_BoW.append('price')
```

```
[[0.00065311]
 [0.05284277]
 [0.08604228]
 ...
 [0.04333119]
 [0.02051941]
 [0.02152757]]
```

In [38]:

```
print("Printing shape of Train, CV and Test data after vectorizing price")

print(X_train_price_std.shape, y_train_upsampled.shape)
print(X_cv_price_std.shape, y_cv.shape)
print(X_test_price_std.shape, y_test.shape)
```

```
Printing shape of Train, CV and Test data after vectorizing price
(104295, 1) (104295,)
(20484, 1) (20484,)
(27312, 1) (27312,)
```

In [39]:

```
# Vectorizing teacher_number_of_previously_posted_projects

previously_posted_projects_scalar = MinMaxScaler()
previously_posted_projects_scalar.fit(x_train_upsampled['teacher_number_of_previously_posted_projec
s'].values.reshape(-1,1))

X_train_posted_projects_std = previously_posted_projects_scalar.transform(x_train_upsampled['teach
er_number_of_previously_posted_projects'].values.reshape(-1,1))
X_cv_posted_projects_std =
previously_posted_projects_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].v
alues.reshape(-1,1))
X_test_posted_projects_std =
previously_posted_projects_scalar.transform(X_test['teacher_number_of_previously_posted_projects']
.values.reshape(-1,1))

print(X_train_posted_projects_std)

Feature_name_BoW.append('teacher_number_of_previously_posted_projects')
```

```
[[0.02660754]
 [0.00665188]
```

```
 [0.00443459]
 ...
 [0.        ]
 [0.00443459]
 [0.00221729]]
```

In [40]:

```
print("Printing shape of Train, CV and Test data after vectorizing
teacher_number_of_previously_posted_projects")

print(X_train_posted_projects_std.shape, y_train_upsampled.shape)
print(X_cv_posted_projects_std.shape, y_cv.shape)
print(X_test_posted_projects_std.shape, y_test.shape)
```

```
Printing shape of Train, CV and Test data after vectorizing
teacher_number_of_previously_posted_projects
(104295, 1) (104295,)
(20484, 1) (20484,)
(27312, 1) (27312,)
```

Bag of Words on Project Essay for train, cv and test data

In [41]:

```
vectorizer = CountVectorizer(min_df=10, binary=True)
vectorizer.fit(train_preprocessed_essays)

X_train_essay_bow = vectorizer.transform(train_preprocessed_essays)
X_cv_essay_bow = vectorizer.transform(cv_preprocessed_essays)
X_test_essay_bow = vectorizer.transform(test_preprocessed_essays)

Feature_name_BoW.extend(vectorizer.get_feature_names())
```

In [42]:

```
print("Shape of train_matrix after BoW on project_essay : ", X_train_essay_bow.shape,
y_train_upsampled.shape)
print("\nShape of cv_matrix after BoW on project_essay : ", X_cv_essay_bow.shape, y_cv.shape)
print("\nShape of test_matrix after BoW on project_essay : ", X_test_essay_bow.shape, y_test.shape
)
```

```
Shape of train_matrix after BoW on project_essay :  (104295, 16518) (104295,)

Shape of cv_matrix after BoW on project_essay :  (20484, 16518) (20484,)

Shape of test_matrix after BoW on project_essay :  (27312, 16518) (27312,)
```

Bag of Words on Project title for train, cv and test data

In [43]:

```
vectorizer = CountVectorizer(min_df=6, binary=True)
vectorizer.fit(train_preprocessed_title)

X_train_title_bow = vectorizer.transform(train_preprocessed_title)
X_cv_title_bow = vectorizer.transform(cv_preprocessed_title)
X_test_title_bow = vectorizer.transform(test_preprocessed_title)

Feature_name_BoW.extend(vectorizer.get_feature_names())
```

In [44]:

```
print("Shape of train_matrix after BoW on project_title : ", X_train_title_bow.shape,
y_train_upsampled.shape)
print("\nShape of cv_matrix after BoW on project_title : ", X_cv_title_bow.shape, y_cv.shape)
print("\nShape of test_matrix after BoW on project_title : ", X_test_title_bow.shape, y_test.shape
```

```
)
```

```
Shape of train_matrix after BoW on project_title :  (104295, 4958) (104295,)

Shape of cv_matrix after BoW on project_title :  (20484, 4958) (20484,)

Shape of test_matrix after BoW on project_title :  (27312, 4958) (27312,)
```

Tf-IDF Vectorizer on preprocessed_essays for train, cv and test data

In [45]:

```
vectorizer = TfidfVectorizer(min_df=10, binary=True)
vectorizer.fit(train_preprocessed_essays)

X_train_essay_tf = vectorizer.transform(train_preprocessed_essays)
X_cv_essay_tf = vectorizer.transform(cv_preprocessed_essays)
X_test_essay_tf = vectorizer.transform(test_preprocessed_essays)
```

In [46]:

```
print("Shape of train_matrix after tfidf on project_essay : ", X_train_essay_tf.shape,
y_train_upsampled.shape)
print("\nShape of cv_matrix after tfidf on project_essay : ", X_cv_essay_tf.shape, y_cv.shape)
print("\nShape of test_matrix after tfidf on project_essay : ", X_test_essay_tf.shape,
y_test.shape)
```

```
Shape of train_matrix after tfidf on project_essay :  (104295, 16518) (104295,)

Shape of cv_matrix after tfidf on project_essay :  (20484, 16518) (20484,)

Shape of test_matrix after tfidf on project_essay :  (27312, 16518) (27312,)
```

Tf-IDF Vectorizer on preprocessed_title for train, cv and test data

In [47]:

```
vectorizer = TfidfVectorizer(min_df=6, binary=True)
vectorizer.fit(train_preprocessed_title)

X_train_title_tf = vectorizer.transform(train_preprocessed_title)
X_cv_title_tf = vectorizer.transform(cv_preprocessed_title)
X_test_title_tf = vectorizer.transform(test_preprocessed_title)
```

In [48]:

```
print("Shape of train_matrix after tfidf on project_title : ", X_train_title_tf.shape,
y_train_upsampled.shape)
print("\nShape of cv_matrix after tfidf on project_title : ", X_cv_title_tf.shape, y_cv.shape)
print("\nShape of test_matrix after tfidf on project_title : ", X_test_title_tf.shape,
y_test.shape)
```

```
Shape of train_matrix after tfidf on project_title :  (104295, 4958) (104295,)

Shape of cv_matrix after tfidf on project_title :  (20484, 4958) (20484,)

Shape of test_matrix after tfidf on project_title :  (27312, 4958) (27312,)
```

Merging all the features

In [49]:

```
# Merging all the features for Set-1 (BoW)
from scipy.sparse import hstack

X_train_s1 = hstack((X_train_cat_onehot, X_train_subcat_onehot, X_train_school_state_onehot,
X_train_teacher_onehot, X_train_clean_grade_onehot, X_train_price_std, X_train_posted_projects_std
, X_train_essay_bow, X_train_title_bow)).tocsr()
X_cv_s1 = hstack((X_cv_cat_onehot, X_cv_subcat_onehot, X_cv_school_state_onehot,
X_cv_teacher_onehot, X_cv_clean_grade_onehot, X_cv_price_std, X_cv_posted_projects_std, X_cv_essay_
bow, X_cv_title_bow)).tocsr()
X_test_s1 = hstack((X_test_cat_onehot, X_test_subcat_onehot, X_test_school_state_onehot,
X_test_teacher_onehot, X_test_clean_grade_onehot, X_test_price_std, X_test_posted_projects_std,
X_test_essay_bow, X_test_title_bow)).tocsr()

print("Final Data matrix of Set-1\n")
print(X_train_s1.shape, y_train_upsampled.shape)
print(X_cv_s1.shape, y_cv.shape)
print(X_test_s1.shape, y_test.shape)
```

```
Final Data matrix of Set-1

(104295, 21577) (104295,)
(20484, 21577) (20484,)
(27312, 21577) (27312,)
```

In [50]:

```
# Merging all the features for Set-2 (Tf-IDF)

X_train_s2 = hstack((X_train_essay_tf, X_train_title_tf, X_train_posted_projects_std,
X_train_price_std, X_train_clean_grade_onehot, X_train_teacher_onehot, X_train_school_state_onehot
, X_train_subcat_onehot, X_train_cat_onehot)).tocsr()
X_cv_s2 = hstack((X_cv_essay_tf, X_cv_title_tf, X_cv_posted_projects_std, X_cv_price_std, X_cv_clea
n_grade_onehot, X_cv_teacher_onehot, X_cv_school_state_onehot, X_cv_subcat_onehot, X_cv_cat_onehot)
).tocsr()
X_test_s2 = hstack((X_test_essay_tf, X_test_title_tf, X_test_posted_projects_std, X_test_price_std
, X_test_clean_grade_onehot, X_test_teacher_onehot, X_test_school_state_onehot,
X_test_subcat_onehot, X_test_cat_onehot)).tocsr()

print("Final Data matrix of Set-2\n")
print(X_train_s2.shape, y_train_upsampled.shape)
print(X_cv_s2.shape, y_cv.shape)
print(X_test_s2.shape, y_test.shape)
```

```
Final Data matrix of Set-2

(104295, 21577) (104295,)
(20484, 21577) (20484,)
(27312, 21577) (27312,)
```

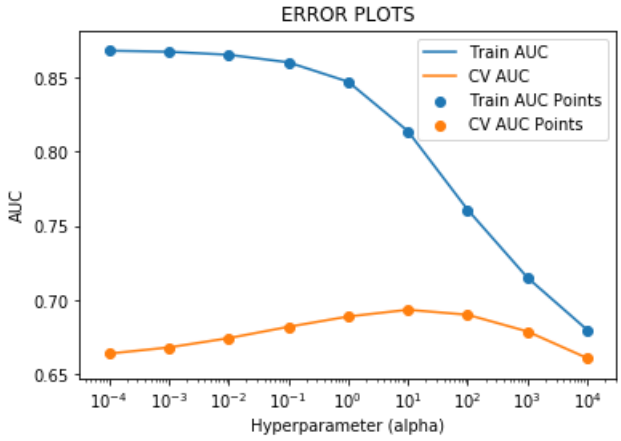
Multinomial NB on Set-1 (BoW)

In [51]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
alpha =[0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]

for i in tqdm(alpha):
    nb1 = MultinomialNB(alpha = i)
    nb1.fit(X_train_s1, y_train_upsampled)

    y_train_pred = nb1.predict_proba(X_train_s1)[:,1]
    y_cv_pred = nb1.predict_proba(X_cv_s1)[:,1]

    train_auc.append(roc_auc_score(y_train_upsampled, y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(alpha, train_auc, label='Train AUC')
```

```
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')
plt.scatter(alpha, train_auc, label='Train AUC Points')
plt.scatter(alpha, cv_auc, label='CV AUC Points')
plt.xscale('log')
plt.legend()
plt.xlabel("Hyperparameter (alpha)")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 9/9 [00
:01<00:00,  4.78it/s]
```



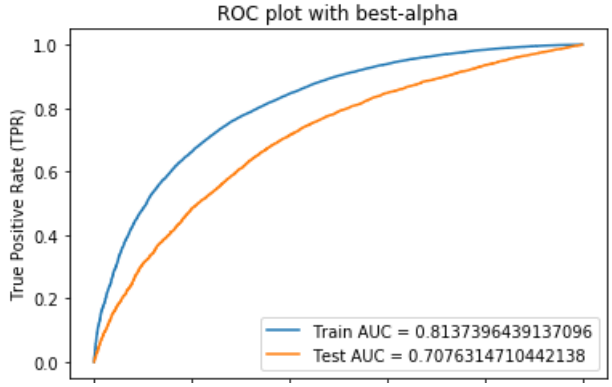
In [52]:

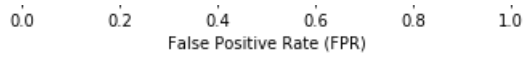
```
Aoc_score_cv = [x for x in cv_auc]
best_alpha_cv = alpha[Aoc_score_cv.index(max(Aoc_score_cv))]
print("Maximum AUC score of cv is : " +  str(max(Aoc_score_cv)))
print("Corresponding best alpha value of cv is : ", best_alpha_cv)
```

```
Maximum AUC score of cv is : 0.6935128983561332
Corresponding best alpha value of cv is :  10
```

In [53]:

```
nb1 = MultinomialNB(alpha = 10)
nb1.fit(X_train_s1, y_train_upsampled)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_upsampled, nb1.predict_proba(X_train_s1)
[:, 1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, nb1.predict_proba(X_test_s1)[:, 1])

plt.plot(train_fpr, train_tpr, label="Train AUC = " + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC = " + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.ylabel("True Positive Rate (TPR)")
plt.xlabel("False Positive Rate (FPR)")
plt.title("ROC plot with best-alpha")
plt.show()
```



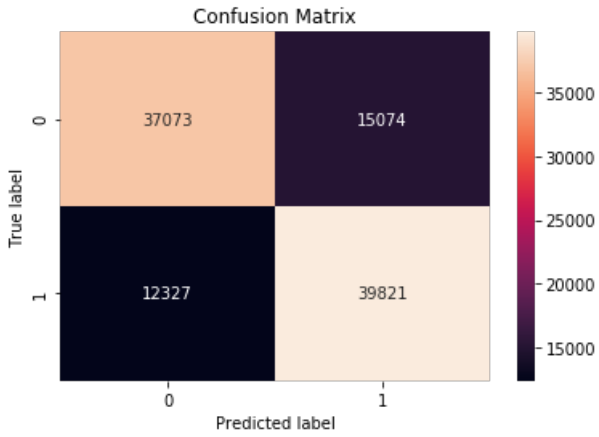


In [54]:

```
# Plotiing Plot for Confusion Matrix fot train data
ax = plt.subplot()

sns.heatmap(confusion_matrix(y_train_upsampled, nb1.predict(X_train_s1)), annot=True, ax = ax, fmt=
'g')
ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.set_title('Confusion Matrix')
```

Out[54]:

```
Text(0.5,1,'Confusion Matrix')
```

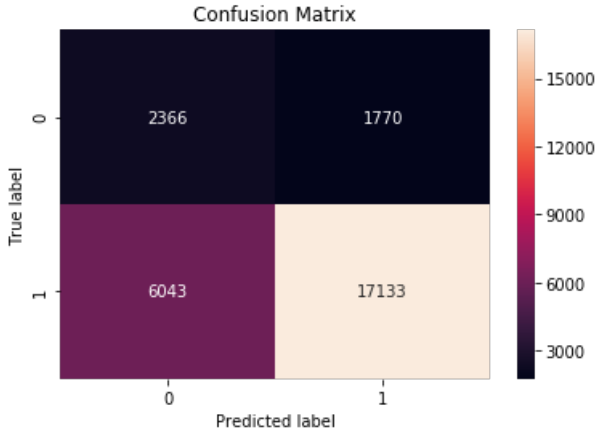


In [55]:

```
# Plotiing Plot for Confusion Matrix for Test Data

ax = plt.subplot()
sns.heatmap(confusion_matrix(y_test, nb1.predict(X_test_s1)), annot=True, ax = ax, fmt='g');
ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.set_title('Confusion Matrix')
```

Out[55]:

```
Text(0.5,1,'Confusion Matrix')
```



Observations of Multinomial NB on Set-1 (BoW)

- Maximum AUC score that we got on CV data is 0.693
- AUC score on Test data after the hyperparameter tuning is 0.707

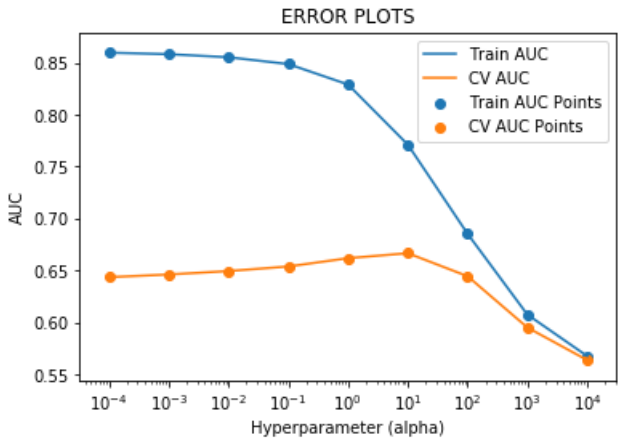
- In Confusion matrix of Train data we are getting good results since the values of TN and TP is large as compare to FP and FN values
- In confusion matrix of Test data we're getting high values of TP and FN. We're getting high values of FN because we have decided best hyperparameter based on CV data which is imbalanced data and because of that imbalancing our model is predicting a datapoint as negative even though it's a positive datapoint.

Multinomial NB on Set-2 (Tf-Idf)

In [56]:

```
train_auc = []
cv_auc = []
alpha =[0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]

for i in tqdm(alpha):
    nb2 = MultinomialNB(alpha = i)
    nb2.fit(X_train_s2, y_train_upsampled)

    y_train_pred = nb2.predict_proba(X_train_s2)[:,1]
    y_cv_pred = nb2.predict_proba(X_cv_s2)[:,1]

    train_auc.append(roc_auc_score(y_train_upsampled, y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')
plt.scatter(alpha, train_auc, label='Train AUC Points')
plt.scatter(alpha, cv_auc, label='CV AUC Points')
plt.xscale('log')
plt.legend()
plt.xlabel("Hyperparameter (alpha)")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████| 9/9 [00
:01<00:00,  4.79it/s]
```



In [57]:

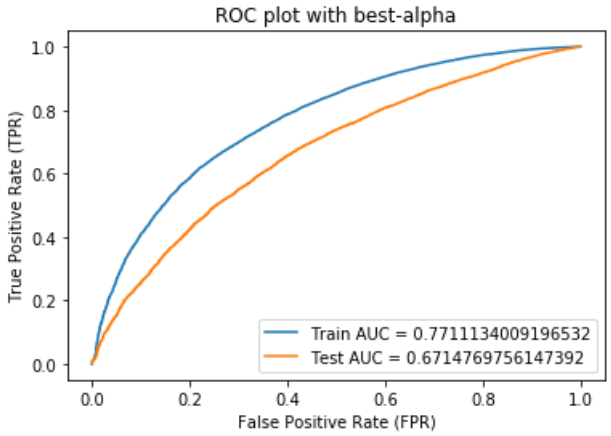
```
Aoc_score_cv = [x for x in cv_auc]
best_alpha_cv = alpha[Aoc_score_cv.index(max(Aoc_score_cv))]
print("Maximum AUC score of cv is : " +  str(max(Aoc_score_cv)))
print("Corresponding best alpha value of cv is : ", best_alpha_cv)
```

```
Maximum AUC score of cv is : 0.6666424636793837
Corresponding best alpha value of cv is :  10
```

In [58]:

```
nb2 = MultinomialNB(alpha = 10)
```

```
nb2.fit(X_train_s2, y_train_upsampled)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_upsampled, nb2.predict_proba(X_train_s2)
[:, 1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, nb2.predict_proba(X_test_s2)[:, 1])

plt.plot(train_fpr, train_tpr, label="Train AUC = " + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC = " + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.ylabel("True Positive Rate (TPR)")
plt.xlabel("False Positive Rate (FPR)")
plt.title("ROC plot with best-alpha")
plt.show()
```

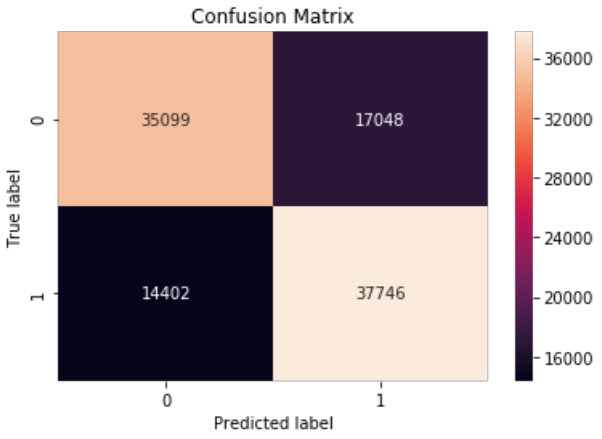


In [59]:

```
# Plotiing Plot for Confusion Matrix for train data
ax = plt.subplot()

sns.heatmap(confusion_matrix(y_train_upsampled, nb2.predict(X_train_s2)), annot=True, ax = ax, fmt=
'g')
ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.set_title('Confusion Matrix')
```

Out[59]:

```
Text(0.5,1,'Confusion Matrix')
```

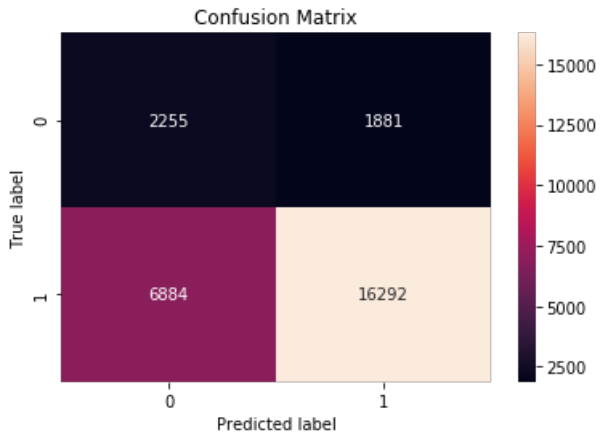


In [60]:

```
# Plotiing Plot for Confusion Matrix for Test Data

ax = plt.subplot()
sns.heatmap(confusion_matrix(y_test, nb2.predict(X_test_s2)), annot=True, ax = ax, fmt='g');
ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.set_title('Confusion Matrix')
```

Out[60]:

```
Text(0.5,1,'Confusion Matrix')
```



Observation on Multinomial NB on Set-2 (Tf-Idf)

- Maximum AUC score that we got on CV data is 0.666
- AUC score on Test data after the hyperparameter tuning is 0.671
- Here also we're getting good confusion matrix on training data but on test data as stated earlier we're getting high number of datapoints belonging to TP and FN.

Finding top 20 features from Set-1 (BoW)

In [61]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

sorted_negative_class_prob = nb1.feature_log_prob_[0, :].argsort()
sorted_positive_class_prob = nb1.feature_log_prob_[1, :].argsort()

print('Top 20 Features of Negative Class : \n')
print(np.take(Feature_name_BoW, sorted_negative_class_prob[-20:]))
print('\n', '='*100, '\n')
print('Top 20 Features of Positive Class : \n')
print(np.take(Feature_name_BoW, sorted_positive_class_prob[-20:]))
```

```
Top 20 Features of Negative Class :

['make' 'use' 'Math_Science' 'prek_2' 'love' 'able' 'Literacy_Language'
 'come' 'work' 'Mrs' 'need' 'many' 'help' 'learn' 'not' 'classroom'
 'learning' 'school' 'nannan' 'students']


====================================================================================================


Top 20 Features of Positive Class :

['class' 'day' 'also' 'love' 'able' 'use' 'come' 'Literacy_Language'
 'work' 'need' 'Mrs' 'many' 'help' 'learn' 'not' 'classroom' 'learning'
 'school' 'nannan' 'students']
```

Summarization

In [63]:

```
from prettytable import PrettyTable

t = PrettyTable()
t.field_names= ("Vectorizer", "Model","HyperParameter : alpha" ,"AUC")
```

```
t.add_row(["BOW", "Multinomial NB", 10, 0.707])
t.add_row(["Tf-Idf", "Multinomial NB", 10, 0.671])

print(t)
```

```
+------------+----------------+------------------------+-------+
| Vectorizer |     Model      | HyperParameter : alpha |  AUC  |
+------------+----------------+------------------------+-------+
|    BOW     | Multinomial NB |           10           | 0.707 |
|   Tf-Idf   | Multinomial NB |           10           | 0.671 |
+------------+----------------+------------------------+-------+
```

Conclusion

- Multinomial NB accepts only non-negative values so we cannot directly use standardization so instead we use either MinMaxScaler or Normalization to overcome the error of negative values.
- I am getting Test AUC score of 0.697 on Set-1 (BoW) and 0.666 on Set-2 (Tf-Idf).
- We're getting better results on BoW as compare to Tf-Idf.
- Since Naive Bayes is super interpretable we can easily obtain which features are most important.
- For selecting top features I am using one of attributes of Multinomial NB which is feature_log_prob which will give us the probability of the same and by the probability values we can select top features.