Qoura question pair similarity (Case-Study)

1. Business Problem

1.1 Description

Quora is a place to gain and share knowledge—about anything. It’s a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world.

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term.

> Credits: Kaggle

_ Problem Statement __

- Identify which questions asked on Quora are duplicates of questions that have already been asked.
- This could be useful to instantly provide answers to questions that have already been answered.
- We are tasked with predicting whether a pair of questions are duplicates or not.

1.2 Sources/Useful Links

- Source : https://www.kaggle.com/c/quora-question-pairs

  Useful Links
- Discussions : https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb/comments
- Kaggle Winning Solution and other approaches: https://www.dropbox.com/sh/93968nfnrzh8bp5/AACZdtsApc1QSTQc7X0H3QZ5a?dl=0
- Blog 1 : https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning
- Blog 2 : https://towardsdatascience.com/identifying-duplicate-questions-on-quora-top-12-on-kaggle-4c1cf93f1c30

1.3 Real world/Business Objectives and Constraints

1. The cost of a mis-classification can be very high.
2. You would want a probability of a pair of questions to be duplicates so that you can choose any threshold of choice.
3. No strict latency concerns.
4. Interpretability is partially important.

2. Machine Learning Probelm

2.1 Data

2.1.1 Data Overview

- Data will be in a file Train.csv
- Train.csv contains 5 columns : qid1, qid2, question1, question2, is_duplicate
- Size of Train.csv - 60MB
- Number of rows in Train.csv = 404,290

2.1.2 Example Data point

```
"id","qid1","qid2","question1","question2","is_duplicate"
```

```
"0","1","2","What is the step by step guide to invest in share market in india?","What is the s
tep by step guide to invest in share market?","0"
"1","3","4","What is the story of Kohinoor (Koh-i-Noor) Diamond?","What would happen if the Ind
ian government stole the Kohinoor (Koh-i-Noor) diamond back?","0"
"7","15","16","How can I be a good geologist?","What should I do to be a great geologist?","1"
"11","23","24","How do I read and find my YouTube comments?","How can I see all my Youtube comm
ents?","1"
```

2.2 Mapping the real world problem to an ML problem

2.2.1 Type of Machine Leaning Problem

It is a binary classification problem, for a given pair of questions we need to predict if they are duplicate or not.

2.2.2 Performance Metric

Source: https://www.kaggle.com/c/quora-question-pairs#evaluation

Metric(s):

- log-loss : https://www.kaggle.com/wiki/LogarithmicLoss
- Binary Confusion Matrix

3. Exploratory Data Analysis

In [35]:
```
#Importing relevant Libraries
import sqlite3
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from subprocess import check_output
%matplotlib inline
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import os
import gc
from sklearn.linear_model import SGDClassifier
from mlxtend.classifier import StackingClassifier
from sklearn.calibration import CalibratedClassifierCV
import re
from nltk.corpus import stopwords
import distance
from nltk.stem import PorterStemmer
from bs4 import BeautifulSoup
import re
from nltk.corpus import stopwords
# This package is used for finding longest common subsequence between two strings
# you can write your own dp code for this
import distance
from nltk.stem import PorterStemmer
from bs4 import BeautifulSoup
from fuzzywuzzy import fuzz
from sklearn.manifold import TSNE
# Import the Required lib packages for WORD-Cloud generation
# https://stackoverflow.com/questions/45625434/how-to-install-wordcloud-in-python3-6
from wordcloud import WordCloud, STOPWORDS
from os import path
from PIL import Image
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from scipy.sparse import hstack
import nltk
nltk.download('stopwords')
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
import sys
import time
from prettytable import PrettyTable
from tqdm import tqdm
import spacy
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

3.1 Reading data and basic stats

In [2]:
```
df = pd.read_csv("train.csv")

print("The total Number of data points present in the dataset:",df.shape[0])
```

```
The total Number of data points present in the dataset: 404290
```

In [3]:
```
#Printing the data-frame
df.head()
```

Out[3]:

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

In [4]:
```
#Printing the basic information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404290 entries, 0 to 404289
Data columns (total 6 columns):
id              404290 non-null int64
qid1            404290 non-null int64
qid2            404290 non-null int64
question1       404289 non-null object
question2       404288 non-null object
is_duplicate    404290 non-null int64
dtypes: int64(4), object(2)
memory usage: 18.5+ MB
```

We are given a minimal number of data fields here, consisting of:

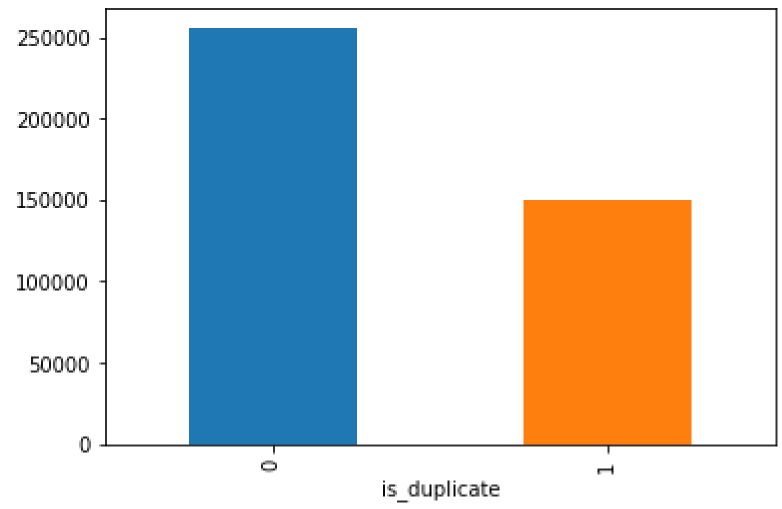
- id: Looks like a simple rowID
- qid{1, 2}: The unique ID of each question in the pair
- question{1, 2}: The actual textual contents of the questions.
- is_duplicate: The label that we are trying to predict - whether the two questions are duplicates of each other.

3.2.1 Distribution of data points among output classes

- Number of duplicate(smilar) and non-duplicate(non similar) questions

In [5]:
```
df.groupby("is_duplicate")['id'].count().plot.bar()
```

Out[5]: `<matplotlib.axes._subplots.AxesSubplot at 0x1e2f4acffd0>`



In [6]:
```
print('~> Total number of question pairs for training available are as follows:\n   {}'.format(len(df
)))
```

```
~> Total number of question pairs for training available are as follows:
   404290
```

In [7]:
```
print('~>The total percentage of Question pairs that are not Similar (is_duplicate = 0):\n   {}%'.form
at(100 - round(df['is_duplicate'].mean()*100, 2)))
print('\n~>The total percentage of Question pairs that are Similar (is_duplicate = 1):\n   {}%'.format
(round(df['is_duplicate'].mean()*100, 2)))
```

```
~>The total percentage of Question pairs that are not Similar (is_duplicate = 0):
   63.08%

~>The total percentage of Question pairs that are Similar (is_duplicate = 1):
   36.92%
```

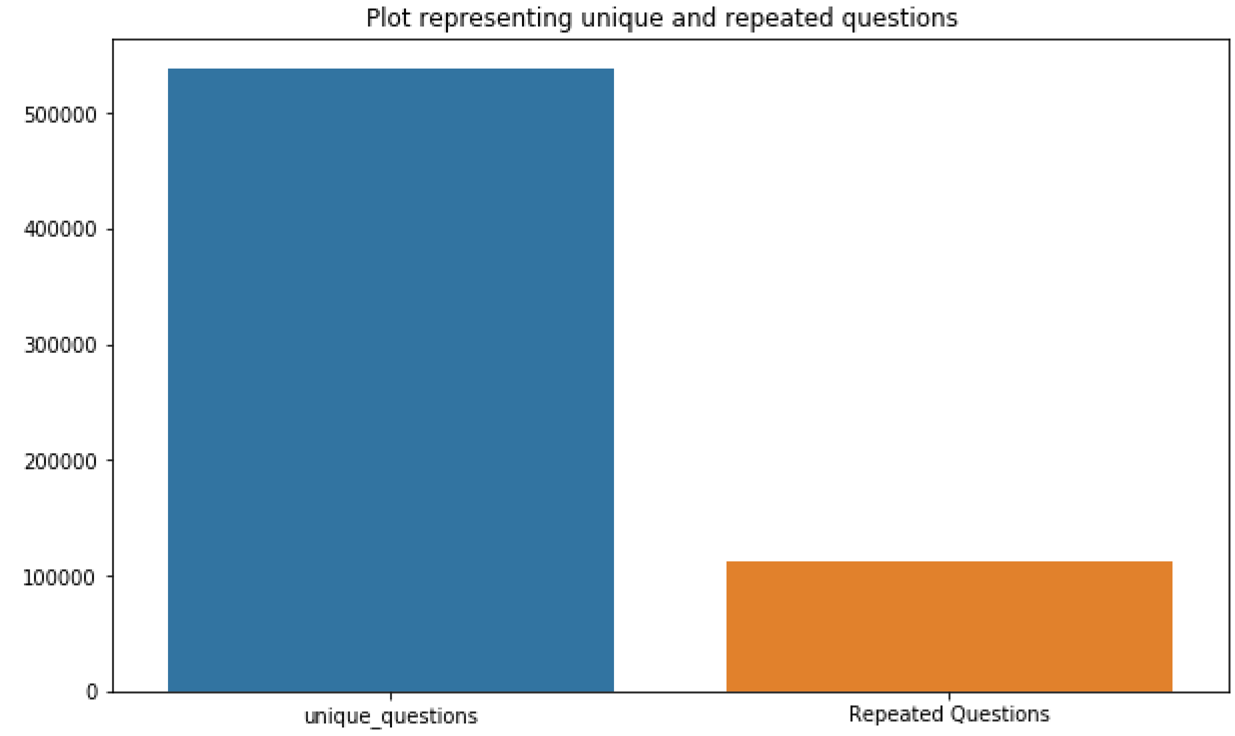
_So clearly there is a slight imbalance in the distribution of the class*labels*__

3.2.2 Number of unique questions

In [8]:
```
qids = pd.Series(df['qid1'].tolist() + df['qid2'].tolist())
unique_qs = len(np.unique(qids))
qs_morethan_onetime = np.sum(qids.value_counts() > 1)
print ('Total number of  Unique Questions are: {}\n'.format(unique_qs))
#print len(np.unique(qids))

print ('Number of unique questions that appear more than one time: {} ({}%)\n'.format(qs_morethan_onet
ime,qs_morethan_onetime/unique_qs*100))

print ('Max number of times a single question is repeated: {}\n'.format(max(qids.value_counts())))

q_vals=qids.value_counts()

q_vals=q_vals.values
```

```
Total number of  Unique Questions are: 537933

Number of unique questions that appear more than one time: 111780 (20.77953945937505%)

Max number of times a single question is repeated: 157
```

In [9]:
```
x = ["unique_questions" , "Repeated Questions"]
y =  [unique_qs , qs_morethan_onetime]

plt.figure(figsize=(10, 6))
plt.title ("Plot representing unique and repeated questions  ")
sns.barplot(x,y)
plt.show()
```



_Clearly the total number of unique questions is greater than the repeated questions___

3.2.3 Checking for Duplicates

In [10]:
```
#checking whether there are any repeated pair of questions

pair_duplicates = df[['qid1','qid2','is_duplicate']].groupby(['qid1','qid2']).count().reset_index()

print ("Number of duplicate questions",(pair_duplicates).shape[0] - df.shape[0])
```

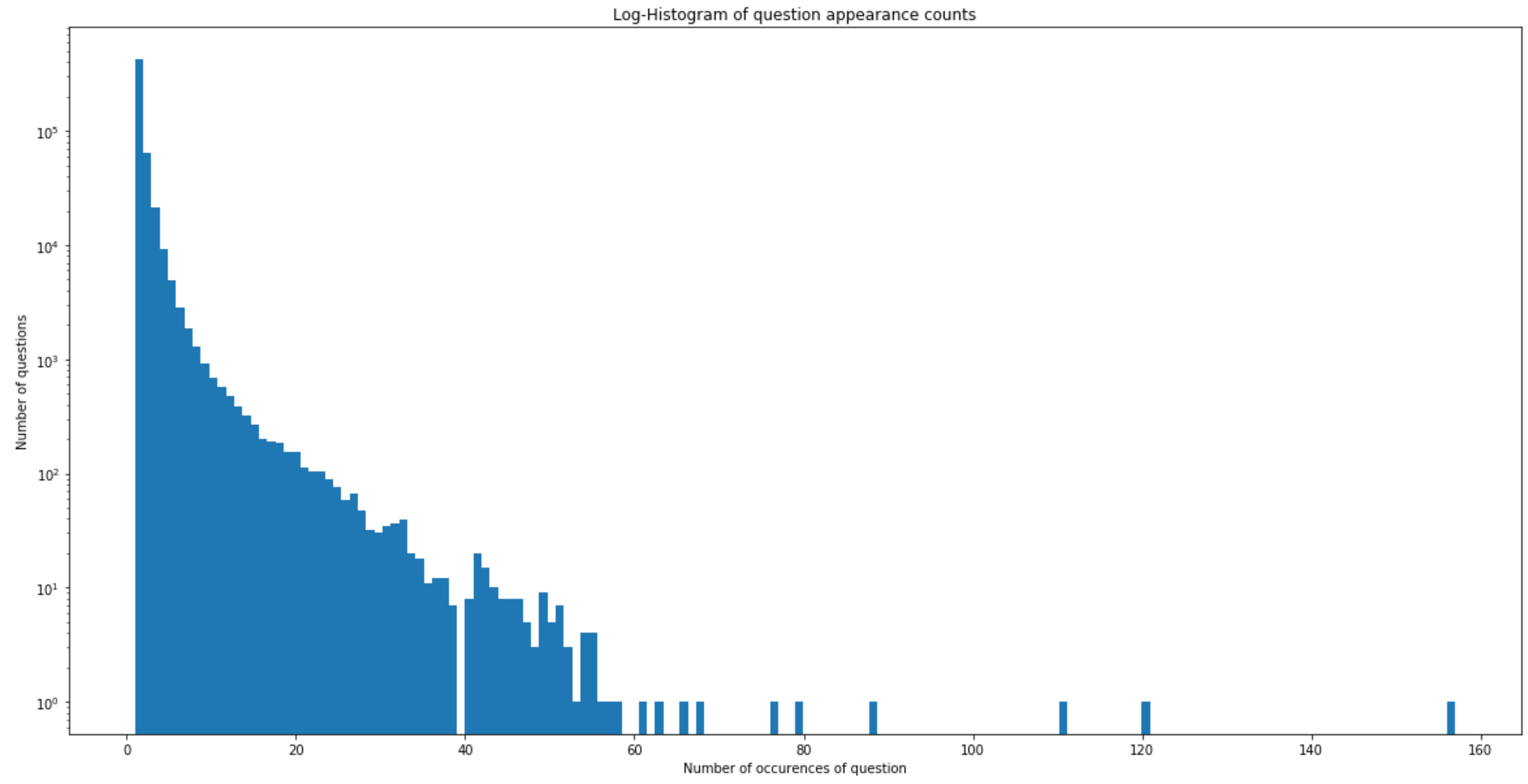
```
Number of duplicate questions 0
```

_So by the above analysis there is no duplicate questions present in the data___

3.2.4 Number of occurrences of each question

In [11]:
```
plt.figure(figsize=(20, 10))

plt.hist(qids.value_counts(), bins=160)

plt.yscale('log', nonposy='clip')

plt.title('Log-Histogram of question appearance counts')

plt.xlabel('Number of occurences of question')

plt.ylabel('Number of questions')

print ('Maximum number of times a single question is repeated: {}\n'.format(max(qids.value_counts())))
```

```
Maximum number of times a single question is repeated: 157
```



- The above distribution of occurrences of the each questions is very skewed and by studying the above plot following van be concluded.

  1. The distribution is very skewed.
  2. Very few number of single questions are repeated more than once.
  3. Only a single question is repeated a total 157 times.

3.2.5 Checking for NULL values

In [12]:
```
#Checking whether there are any rows with null values
nan_rows = df[df.isnull().any(1)]
print (nan_rows)
```

```
            id    qid1    qid2                         question1  \
105780  105780  174363  174364    How can I develop android app?
201841  201841  303951  174364  How can I create an Android app?
363362  363362  493340  493341                               NaN

                                               question2  is_duplicate
105780                                               NaN             0
201841                                               NaN             0
363362  My Chinese name is Haichao Yu. What English na...             0
```

- There are two rows with null values in question2

In [13]:
```
# Filling the null values with space as a character
df = df.fillna('')
nan_rows = df[df.isnull().any(1)]
print (nan_rows)
```

```
Empty DataFrame
Columns: [id, qid1, qid2, question1, question2, is_duplicate]
Index: []
```

3.3 Basic Feature Extraction (before cleaning)

Let us now construct a few features like:

- **freq_qid1** = Frequency of qid1's
- **freq_qid2** = Frequency of qid2's
- **q1len** = Length of q1
- **q2len** = Length of q2
- **q1_n_words** = Number of words in Question 1
- **q2_n_words** = Number of words in Question 2
- **word_Common** = (Number of common unique words in Question 1 and Question 2)
- **word_Total** =(Total num of words in Question 1 + Total num of words in Question 2)
- **word_share** = (word_common)/(word_Total)
- **freq_q1+freq_q2** = sum total of frequency of qid1 and qid2
- **freq_q1-freq_q2** = absolute difference of frequency of qid1 and qid2

In [14]:
```python
if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    df = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')
else:
    df['freq_qid1'] = df.groupby('qid1')['qid1'].transform('count')
    df['freq_qid2'] = df.groupby('qid2')['qid2'].transform('count')
    df['q1len'] = df['question1'].str.len()
    df['q2len'] = df['question2'].str.len()
    df['q1_n_words'] = df['question1'].apply(lambda row: len(row.split(" ")))
    df['q2_n_words'] = df['question2'].apply(lambda row: len(row.split(" ")))

    def normalized_word_Common(row):
        w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
        w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
        return 1.0 * len(w1 & w2)
    df['word_Common'] = df.apply(normalized_word_Common, axis=1)

    def normalized_word_Total(row):
        w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
        w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
        return 1.0 * (len(w1) + len(w2))
    df['word_Total'] = df.apply(normalized_word_Total, axis=1)

    def normalized_word_share(row):
        w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
        w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
        return 1.0 * len(w1 & w2)/(len(w1) + len(w2))
    df['word_share'] = df.apply(normalized_word_share, axis=1)

    df['freq_q1+q2'] = df['freq_qid1']+df['freq_qid2']
    df['freq_q1-q2'] = abs(df['freq_qid1']-df['freq_qid2'])

    df.to_csv("df_fe_without_preprocessing_train.csv", index=False)

df.head()
```

Out[14]:

	id	qid1	qid2	question1	question2	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_C
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0	1	1	66	57	14	12	
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	4	1	51	88	8	13	
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0	1	1	73	59	14	10	
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0	1	1	50	65	11	9	
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0	3	1	76	39	13	7	

3.3.1 Analysis of some of the extracted features

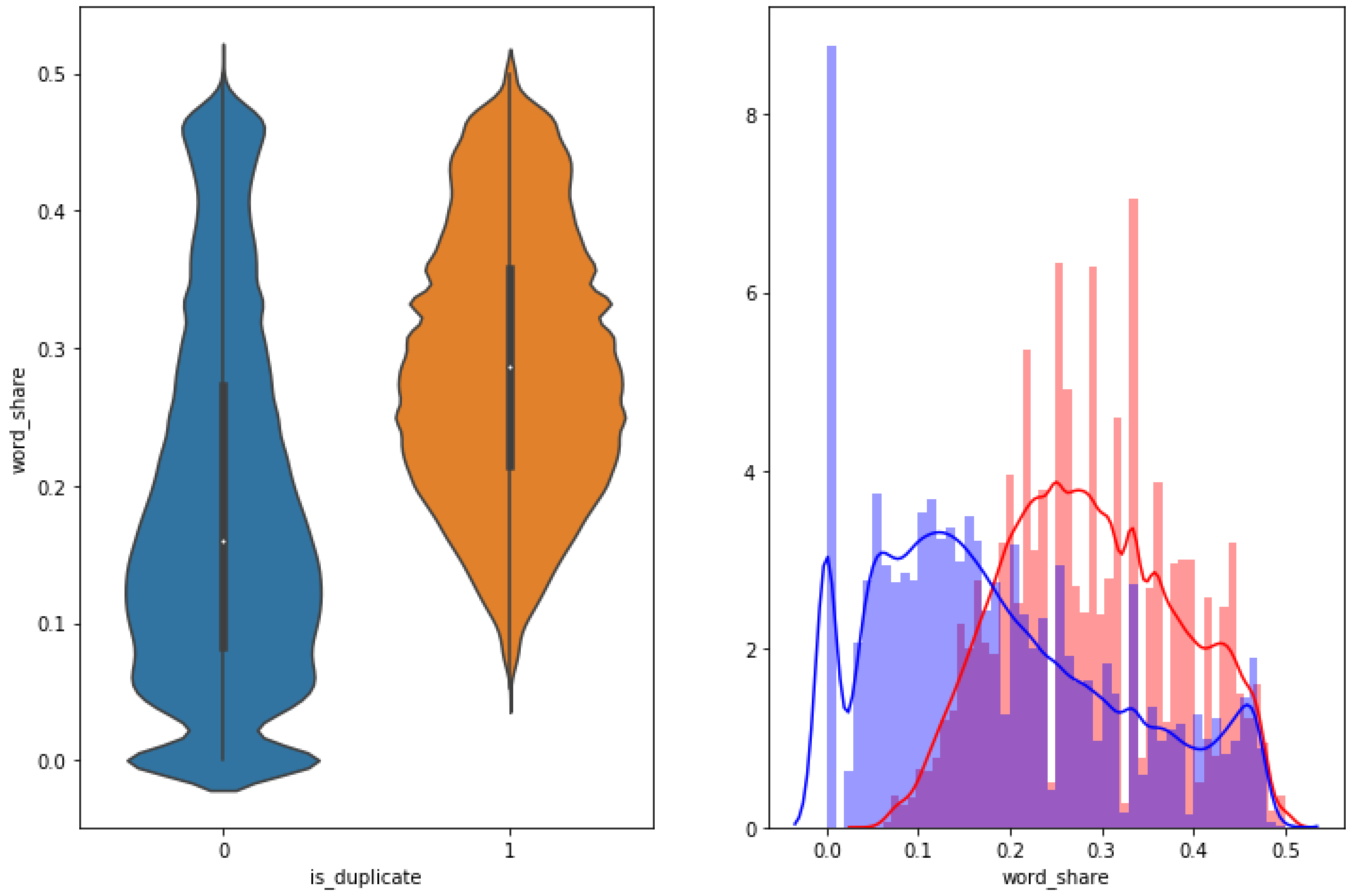
- Here are some questions have only one single words.

In [15]:
```python
print ("Minimum length of the questions in question1 : " , min(df['q1_n_words']))

print ("Minimum length of the questions in question2 : " , min(df['q2_n_words']))

print ("Number of Questions with minimum length [question1] :", df[df['q1_n_words']== 1].shape[0])
print ("Number of Questions with minimum length [question2] :", df[df['q2_n_words']== 1].shape[0])
```

```
Minimum length of the questions in question1 :  1
Minimum length of the questions in question2 :  1
Number of Questions with minimum length [question1] : 67
Number of Questions with minimum length [question2] : 24
```

3.3.1.1 Feature: word_share

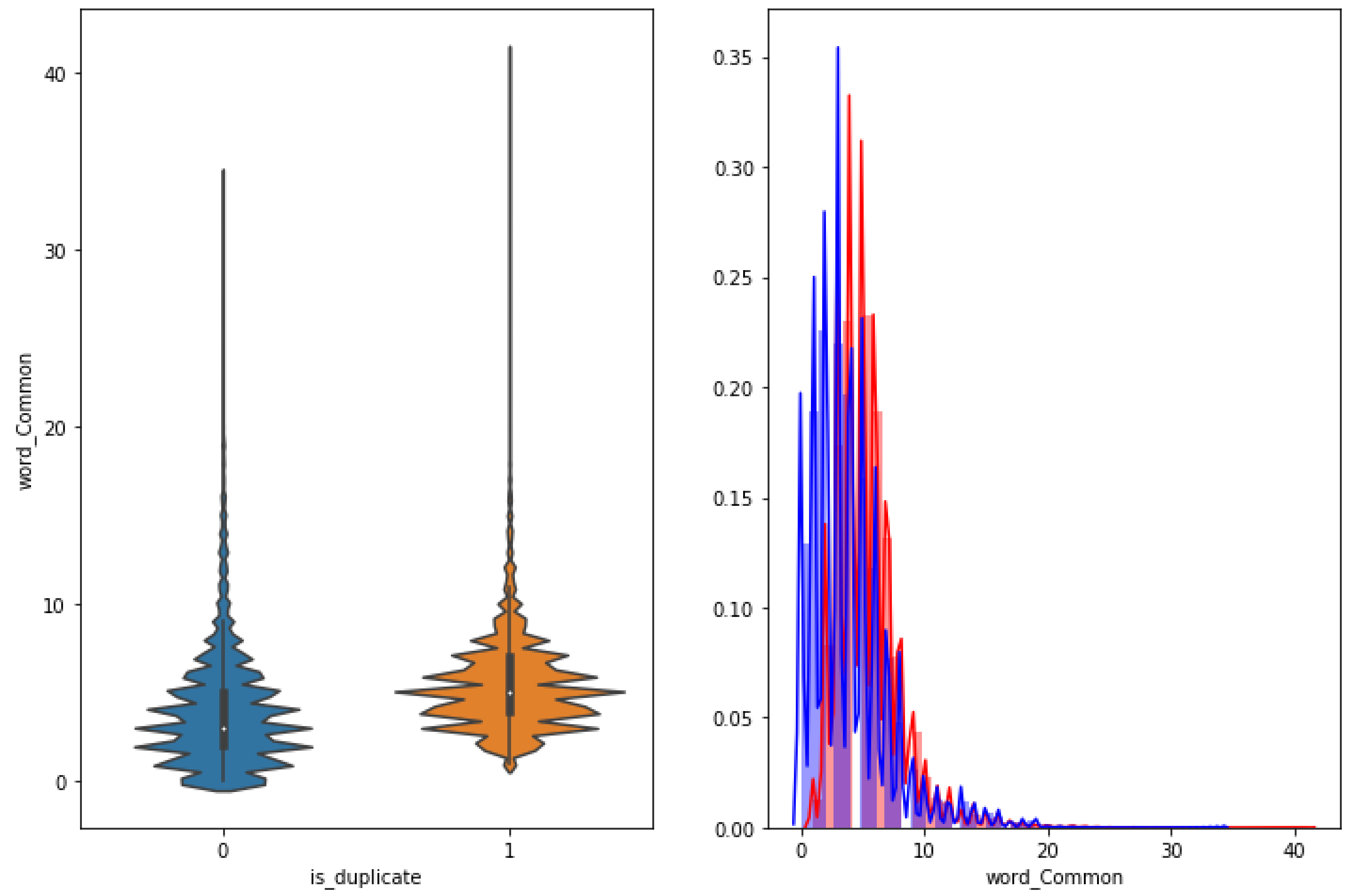
In [16]:
```
plt.figure(figsize=(12, 8))

plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'word_share', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['word_share'][0:] , label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['word_share'][0:] , label = "0" , color = 'blue' )
plt.show()
```



- The distributions for normalized word_share have some overlap on the far right-hand side, i.e., there are quite a lot of questions with high word similarity
- The average word share and Common no. of words of qid1 and qid2 is more when they are duplicate(Similar)

3.3.1.2 Feature: word_Common

In [17]:
```
plt.figure(figsize=(12, 8))

plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'word_Common', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['word_Common'][0:] , label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['word_Common'][0:] , label = "0" , color = 'blue' )
plt.show()
```



The distributions of the word_Common feature in similar and non-similar questions are highly overlapping

1.2.1 : EDA: Advanced Feature Extraction.

In [18]:
```
#https://stackoverflow.com/questions/12468179/unicodedecodeerror-utf8-codec-cant-decode-byte-0x9c
if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    df = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')
    df = df.fillna('')
    df.head()
else:
    print("get df_fe_without_preprocessing_train.csv from drive or run the previous notebook")
```

In [19]:
```
df.head(2)
```

Out[19]:

	id	qid1	qid2	question1	question2	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_Co
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0	1	1	66	57	14	12	
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	4	1	51	88	8	13	

3.4 Preprocessing of Text

- Preprocessing:
  - Removing html tags
  - Removing Punctuations
  - Performing stemming
  - Removing Stopwords
  - Expanding contractions etc.

In [20]:
```python
# To get the results in 4 decemal points

SAFE_DIV = 0.0001

STOP_WORDS = stopwords.words("english")


def preprocess(x):
    x = str(x).lower()
    x = x.replace(",000,000", "m").replace(",000", "k").replace("′", "'").replace("’", "'")\
                           .replace("won't", "will not").replace("cannot", "can not").replace("can't",
 "can not")\
                           .replace("n't", " not").replace("what's", "what is").replace("it's", "it i
s")\
                           .replace("'ve", " have").replace("i'm", "i am").replace("'re", " are")\
                           .replace("he's", "he is").replace("she's", "she is").replace("'s", " own")\
                           .replace("%", " percent ").replace("₹", " rupee ").replace("$", " dollar ")
\
                           .replace("€", " euro ").replace("'ll", " will")
    x = re.sub(r"([0-9]+)000000", r"\1m", x)
    x = re.sub(r"([0-9]+)000", r"\1k", x)


    porter = PorterStemmer()
    pattern = re.compile('\W')

    if type(x) == type(''):
        x = re.sub(pattern, ' ', x)


    if type(x) == type(''):
        x = porter.stem(x)
        example1 = BeautifulSoup(x)
        x = example1.get_text()


    return x
```

- Function to Compute and get the features : With 2 parameters of Question 1 and Question 2

3.5 Advanced Feature Extraction (NLP and Fuzzy Features)

Definition:

- **Token**: You get a token by splitting sentence a space
- **Stop_Word** : stop words as per NLTK.
- **Word** : A token that is not a stop_word

Features:

- **cwc_min** : Ratio of common_word_count to min lenghth of word count of Q1 and Q2
  cwc_min = common_word_count / (min(len(q1_words), len(q2_words))

- **cwc_max** : Ratio of common_word_count to max lenghth of word count of Q1 and Q2
  cwc_max = common_word_count / (max(len(q1_words), len(q2_words))

- **csc_min** : Ratio of common_stop_count to min lenghth of stop count of Q1 and Q2
  csc_min = common_stop_count / (min(len(q1_stops), len(q2_stops))

- **csc_max** : Ratio of common_stop_count to max lenghth of stop count of Q1 and Q2
  csc_max = common_stop_count / (max(len(q1_stops), len(q2_stops))

- **ctc_min** : Ratio of common_token_count to min lenghth of token count of Q1 and Q2
  ctc_min = common_token_count / (min(len(q1_tokens), len(q2_tokens))

- **ctc_max** : Ratio of common_token_count to max lenghth of token count of Q1 and Q2
  ctc_max = common_token_count / (max(len(q1_tokens), len(q2_tokens))

- **last_word_eq** : Check if First word of both questions is equal or not
  last_word_eq = int(q1_tokens[-1] == q2_tokens[-1])

- **first_word_eq** : Check if First word of both questions is equal or not
  first_word_eq = int(q1_tokens[0] == q2_tokens[0])

- **abs_len_diff** : Abs. length difference
  abs_len_diff = abs(len(q1_tokens) - len(q2_tokens))

- **mean_len** : Average Token Length of both Questions
mean_len = (len(q1_tokens) + len(q2_tokens))/2

- **fuzz_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

- **fuzz_partial_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

- **token_sort_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

- **token_set_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

- **longest_substr_ratio** : Ratio of length longest common substring to min lenghth of token count of Q1 and Q2
longest_substr_ratio = len(longest common substring) / (min(len(q1_tokens), len(q2_tokens))

In [21]:
```
def get_token_features(q1, q2):
    token_features = [0.0]*10

    # Converting the Sentence into Tokens:
    q1_tokens = q1.split()
    q2_tokens = q2.split()

    if len(q1_tokens) == 0 or len(q2_tokens) == 0:
        return token_features
    # Get the non-stopwords in Questions
    q1_words = set([word for word in q1_tokens if word not in STOP_WORDS])
    q2_words = set([word for word in q2_tokens if word not in STOP_WORDS])

    #Get the stopwords in Questions
    q1_stops = set([word for word in q1_tokens if word in STOP_WORDS])
    q2_stops = set([word for word in q2_tokens if word in STOP_WORDS])

    # Get the common non-stopwords from Question pair
    common_word_count = len(q1_words.intersection(q2_words))

    # Get the common stopwords from Question pair
    common_stop_count = len(q1_stops.intersection(q2_stops))

    # Get the common Tokens from Question pair
    common_token_count = len(set(q1_tokens).intersection(set(q2_tokens)))


    token_features[0] = common_word_count / (min(len(q1_words), len(q2_words)) + SAFE_DIV)
    token_features[1] = common_word_count / (max(len(q1_words), len(q2_words)) + SAFE_DIV)
    token_features[2] = common_stop_count / (min(len(q1_stops), len(q2_stops)) + SAFE_DIV)
    token_features[3] = common_stop_count / (max(len(q1_stops), len(q2_stops)) + SAFE_DIV)
    token_features[4] = common_token_count / (min(len(q1_tokens), len(q2_tokens)) + SAFE_DIV)
    token_features[5] = common_token_count / (max(len(q1_tokens), len(q2_tokens)) + SAFE_DIV)

    # Last word of both question is same or not
    token_features[6] = int(q1_tokens[-1] == q2_tokens[-1])

    # First word of both question is same or not
    token_features[7] = int(q1_tokens[0] == q2_tokens[0])

    token_features[8] = abs(len(q1_tokens) - len(q2_tokens))

    #Average Token Length of both Questions
    token_features[9] = (len(q1_tokens) + len(q2_tokens))/2
    return token_features

# get the Longest Common sub string

def get_longest_substr_ratio(a, b):
    strs = list(distance.lcsubstrings(a, b))
    if len(strs) == 0:
        return 0
    else:
        return len(strs[0]) / (min(len(a), len(b)) + 1)

def extract_features(df):
    # preprocessing each question
    df["question1"] = df["question1"].fillna("").apply(preprocess)
    df["question2"] = df["question2"].fillna("").apply(preprocess)

    print("token features...")

    # Merging Features with dataset

    token_features = df.apply(lambda x: get_token_features(x["question1"], x["question2"]), axis=1)

    df["cwc_min"]       = list(map(lambda x: x[0], token_features))
    df["cwc_max"]       = list(map(lambda x: x[1], token_features))
    df["csc_min"]       = list(map(lambda x: x[2], token_features))
    df["csc_max"]       = list(map(lambda x: x[3], token_features))
    df["ctc_min"]       = list(map(lambda x: x[4], token_features))
    df["ctc_max"]       = list(map(lambda x: x[5], token_features))
    df["last_word_eq"]  = list(map(lambda x: x[6], token_features))
    df["first_word_eq"] = list(map(lambda x: x[7], token_features))
    df["abs_len_diff"]  = list(map(lambda x: x[8], token_features))
    df["mean_len"]      = list(map(lambda x: x[9], token_features))

    #Computing Fuzzy Features and Merging with Dataset

    # do read this blog: http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/
    # https://stackoverflow.com/questions/31806695/when-to-use-which-fuzz-function-to-compare-2-string
s
    # https://github.com/seatgeek/fuzzywuzzy
    print("fuzzy features..")
```

```
    df["token_set_ratio"]       = df.apply(lambda x: fuzz.token_set_ratio(x["question1"], x["question
2"]), axis=1)
    # The token sort approach involves tokenizing the string in question, sorting the tokens alphabeti
cally, and
    # then joining them back into a string We then compare the transformed strings with a simple ratio
().
    df["token_sort_ratio"]      = df.apply(lambda x: fuzz.token_sort_ratio(x["question1"], x["question
2"]), axis=1)
    df["fuzz_ratio"]            = df.apply(lambda x: fuzz.QRatio(x["question1"], x["question2"]), axis
=1)
    df["fuzz_partial_ratio"]    = df.apply(lambda x: fuzz.partial_ratio(x["question1"], x["question2"]
]), axis=1)
    df["longest_substr_ratio"]  = df.apply(lambda x: get_longest_substr_ratio(x["question1"], x["quest
ion2"]), axis=1)
    return df
```

In [22]:

```
if os.path.isfile('nlp_features_train.csv'):
    df = pd.read_csv("nlp_features_train.csv",encoding='latin-1')
    df.fillna('')
else:
    print("Extracting features for train:")
    df = pd.read_csv("train.csv")
    df = extract_features(df)
    df.to_csv("nlp_features_train.csv", index=False)
df.head(2)
```

Out[22]:

	id	qid1	qid2	question1	question2	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	...	ctc_max	last_word_eq	fir
0	0	1	2	what is the step by step guide to invest in sh...	what is the step by step guide to invest in sh...	0	0.999980	0.833319	0.999983	0.999983	...	0.785709	0.0	
1	1	3	4	what is the story of kohinoor koh i noor dia...	what would happen if the indian government sto...	0	0.799984	0.399996	0.749981	0.599988	...	0.466664	0.0	

2 rows × 21 columns

3.5.1 Analysis of extracted features

3.5.1.1 Plotting Word clouds

- Creating Word Cloud of Duplicates and Non-Duplicates Question pairs
- We can observe the most frequent occuring words

In [23]:

```
df_duplicate = df[df['is_duplicate'] == 1]
dfp_nonduplicate = df[df['is_duplicate'] == 0]

# Converting 2d array of q1 and q2 and flatten the array: like {{1,2},{3,4}} to {1,2,3,4}
p = np.dstack([df_duplicate["question1"], df_duplicate["question2"]]).flatten()
n = np.dstack([dfp_nonduplicate["question1"], dfp_nonduplicate["question2"]]).flatten()

print ("Number of data points in class 1 (duplicate pairs) :",len(p))
print ("Number of data points in class 0 (non duplicate pairs) :",len(n))

#Saving the np array into a text file
np.savetxt('train_p.txt', p, delimiter=' ', fmt='%s',encoding='utf-8')
np.savetxt('train_n.txt', n, delimiter=' ', fmt='%s',encoding='utf-8')
```

```
Number of data points in class 1 (duplicate pairs) : 298526
Number of data points in class 0 (non duplicate pairs) : 510054
```

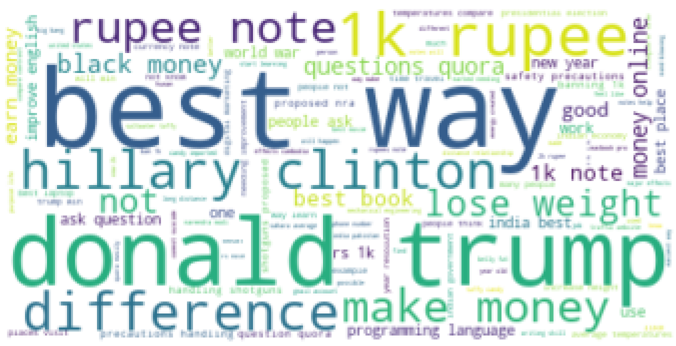
In [24]:
```python
# reading the text files and removing the Stop Words:
d = path.dirname('.')

textp_w = open(path.join(d, 'train_p.txt'),encoding='utf-8').read()
textn_w = open(path.join(d, 'train_n.txt'),encoding='utf-8').read()
stopwords = set(STOPWORDS)
stopwords.add("said")
stopwords.add("br")
stopwords.add(" ")
stopwords.remove("not")

stopwords.remove("no")
#stopwords.remove("good")
#stopwords.remove("love")
stopwords.remove("like")
#stopwords.remove("best")
#stopwords.remove("!")
print ("Total number of words in duplicate pair questions :",len(textp_w))
print ("Total number of words in non duplicate pair questions :",len(textn_w))
```

```
Total number of words in duplicate pair questions : 16110763
Total number of words in non duplicate pair questions : 33201102
```

Word Clouds generated from duplicate pair question's text

In [25]:
```python
wc = WordCloud(background_color="white", max_words=len(textp_w), stopwords=stopwords)
wc.generate(textp_w)
print ("Word Cloud for Duplicate Question pairs")
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Word Cloud for Duplicate Question pairs
```



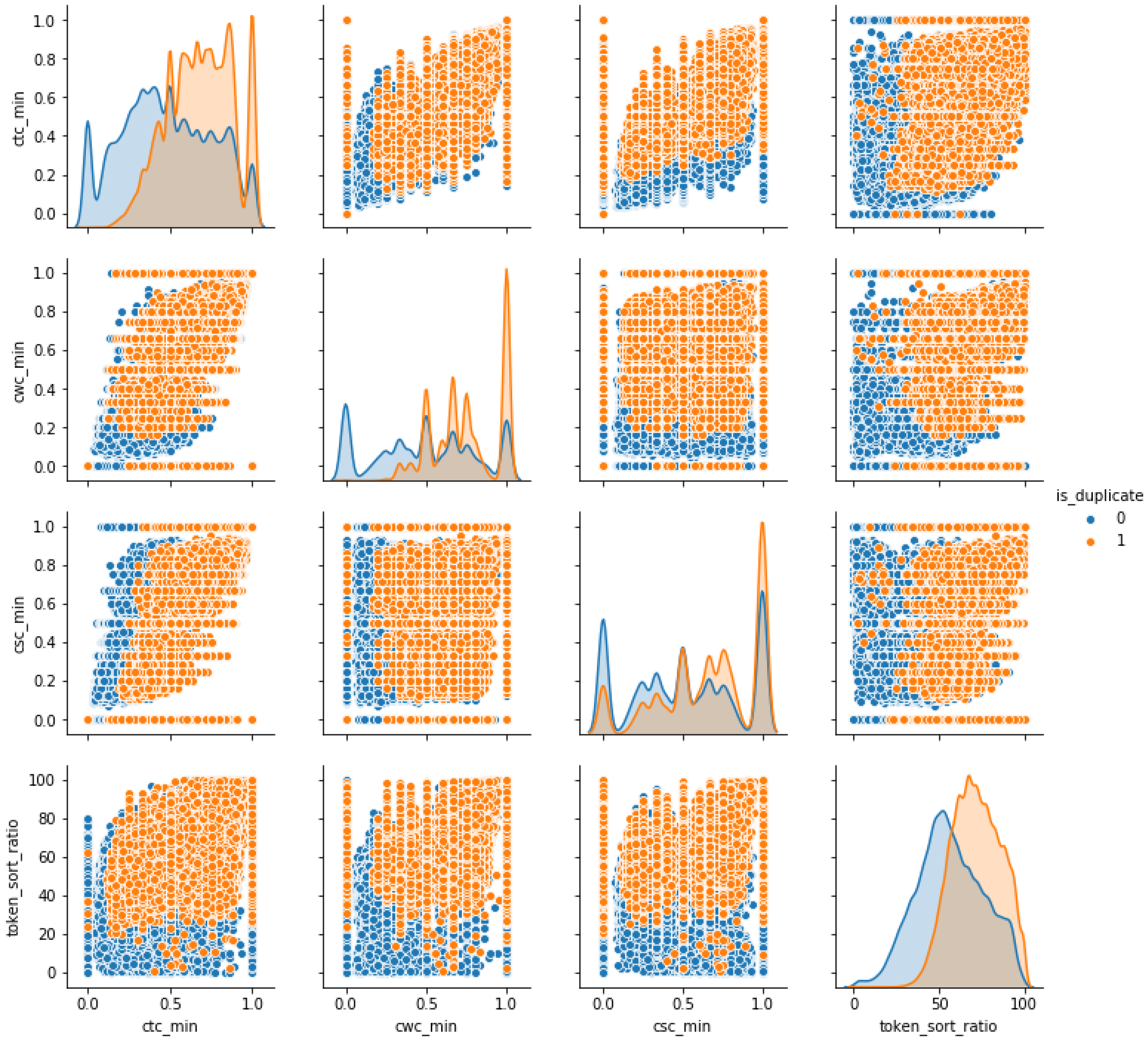
Word Clouds generated from non duplicate pair question's text

In [26]:
```python
wc = WordCloud(background_color="white", max_words=len(textn_w),stopwords=stopwords)
# generate word cloud
wc.generate(textn_w)
print ("Word Cloud for non-Duplicate Question pairs:")
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

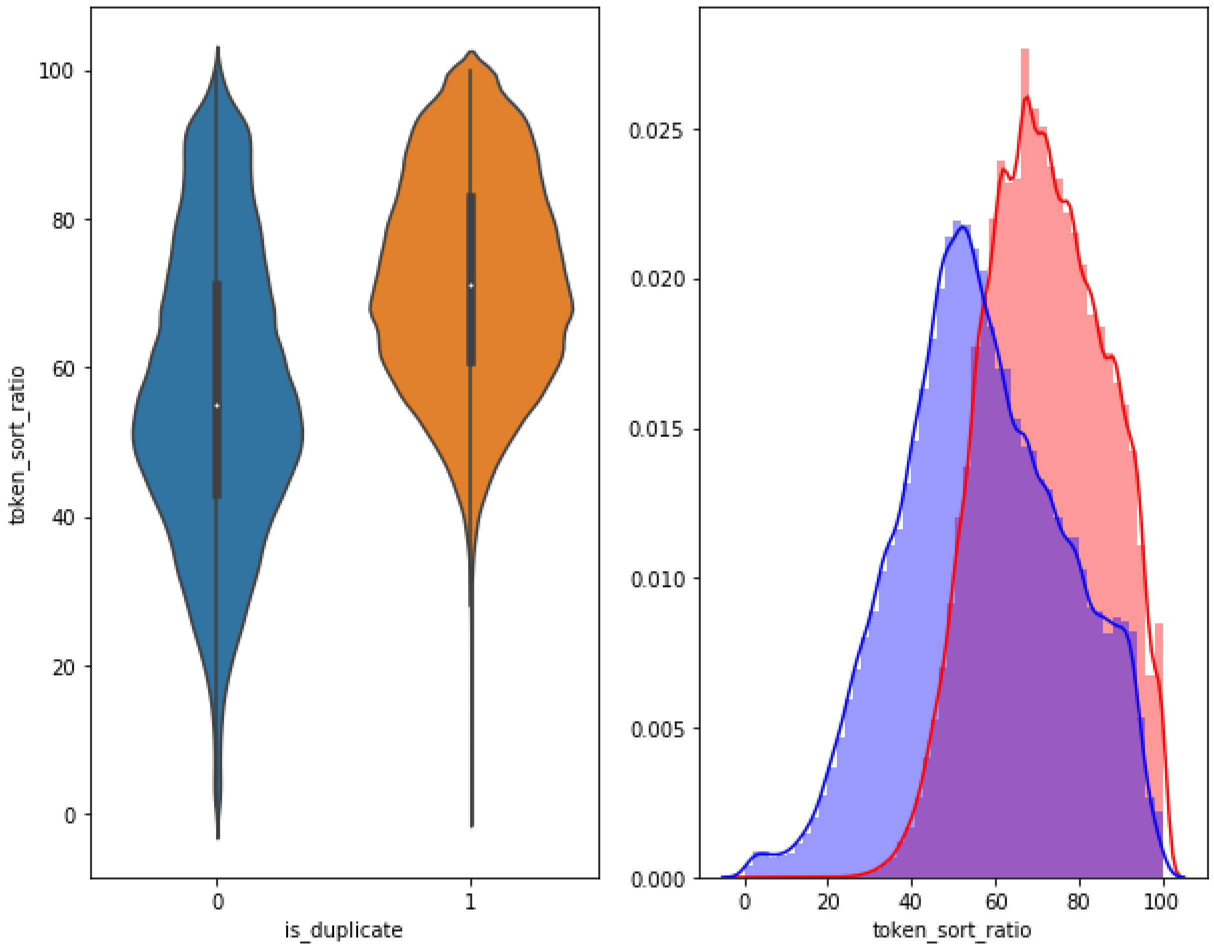
```
Word Cloud for non-Duplicate Question pairs:
```

3.5.1.2 Pair plot of features ['ctc_min', 'cwc_min', 'csc_min', 'token_sort_ratio']

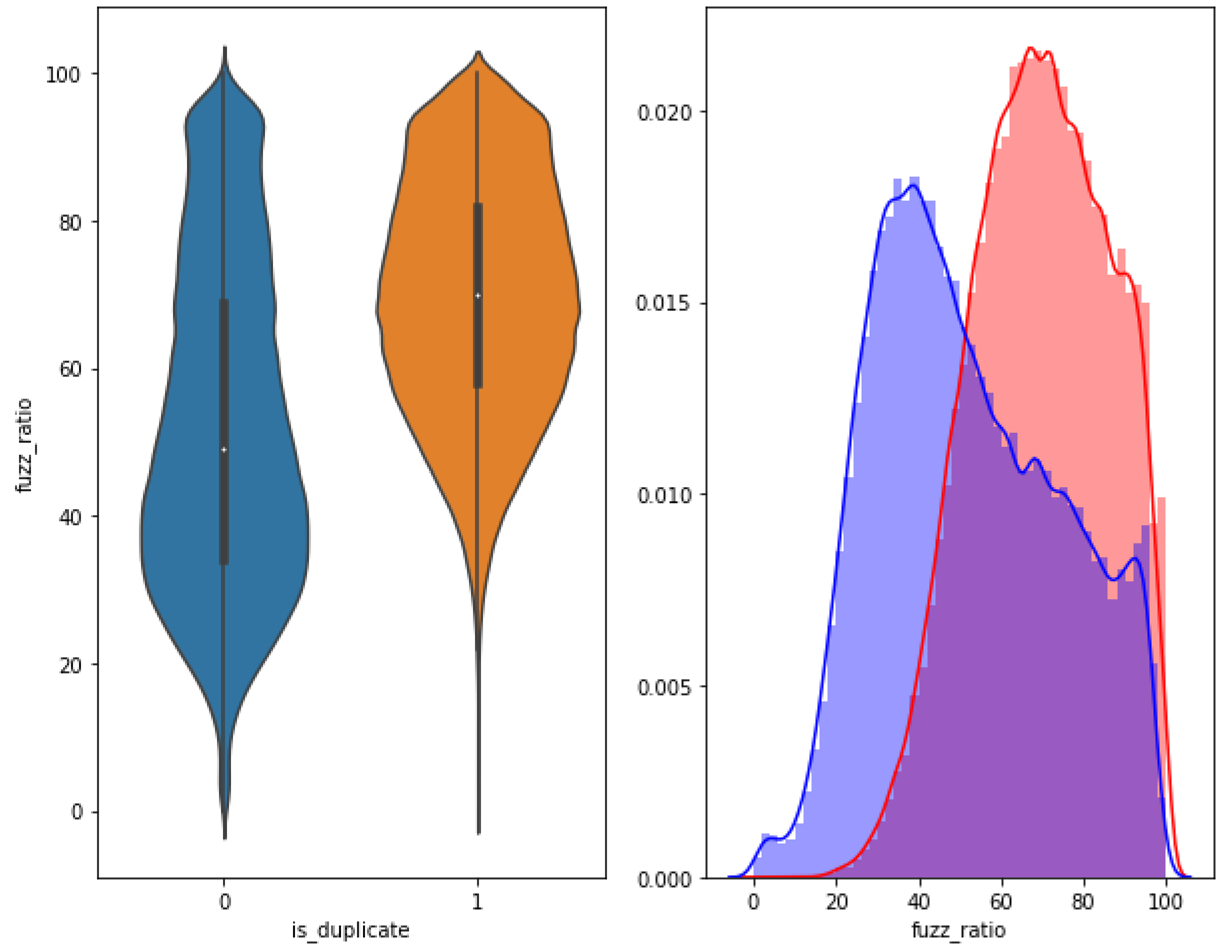
In [27]:
```python
n = df.shape[0]
sns.pairplot(df[['ctc_min', 'cwc_min', 'csc_min', 'token_sort_ratio', 'is_duplicate']][0:n], hue='is_d
uplicate', vars=['ctc_min', 'cwc_min', 'csc_min', 'token_sort_ratio'])
plt.show()
```



In [28]:
```python
# Distribution of the token_sort_ratio
plt.figure(figsize=(10, 8))

plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'token_sort_ratio', data = df[0:] , )

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['token_sort_ratio'][0:] , label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['token_sort_ratio'][0:] , label = "0" , color = 'blue' )
plt.show()
```



In [29]:
```
plt.figure(figsize=(10, 8))

plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'fuzz_ratio', data = df[0:] , )

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['fuzz_ratio'][0:] , label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['fuzz_ratio'][0:] , label = "0" , color = 'blue' )
plt.show()
```



3.5.2 Visualization

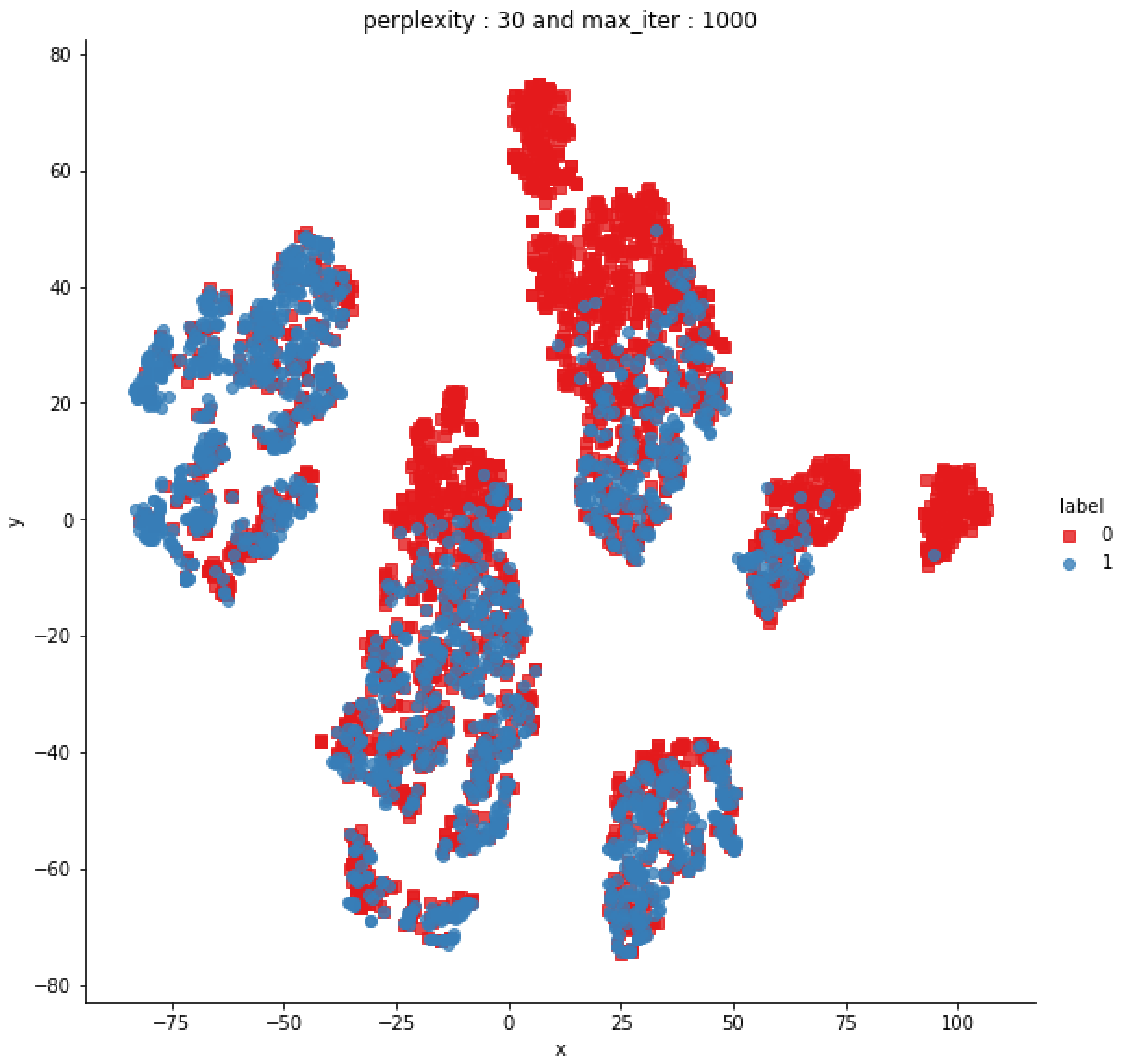
In [30]:
```
# Using TSNE for Dimentionality reduction for 15 Features(Generated after cleaning the data) to 3 dime
ntion

from sklearn.preprocessing import MinMaxScaler

dfp_subsampled = df[0:5000]
X = MinMaxScaler().fit_transform(dfp_subsampled[['cwc_min', 'cwc_max', 'csc_min', 'csc_max' , 'ctc_mi
n' , 'ctc_max' , 'last_word_eq', 'first_word_eq' , 'abs_len_diff' , 'mean_len' , 'token_set_ratio' ,
'token_sort_ratio' ,  'fuzz_ratio' , 'fuzz_partial_ratio' , 'longest_substr_ratio']])
y = dfp_subsampled['is_duplicate'].values
```

In [31]:
```
tsne2d = TSNE(
    n_components=2,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 0.009s...
[t-SNE] Computed neighbors for 5000 samples in 0.320s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 0.130446
[t-SNE] Computed conditional probabilities in 0.176s
[t-SNE] Iteration 50: error = 81.2897949, gradient norm = 0.0455700 (50 iterations in 6.056s)
[t-SNE] Iteration 100: error = 70.6164398, gradient norm = 0.0095177 (50 iterations in 4.406s)
[t-SNE] Iteration 150: error = 68.9172134, gradient norm = 0.0056736 (50 iterations in 4.286s)
[t-SNE] Iteration 200: error = 68.1004639, gradient norm = 0.0049672 (50 iterations in 4.406s)
[t-SNE] Iteration 250: error = 67.5914536, gradient norm = 0.0039700 (50 iterations in 4.488s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 67.591454
[t-SNE] Iteration 300: error = 1.7926962, gradient norm = 0.0011878 (50 iterations in 4.743s)
[t-SNE] Iteration 350: error = 1.3936826, gradient norm = 0.0004807 (50 iterations in 4.654s)
[t-SNE] Iteration 400: error = 1.2281071, gradient norm = 0.0002778 (50 iterations in 4.623s)
[t-SNE] Iteration 450: error = 1.1385784, gradient norm = 0.0001864 (50 iterations in 4.640s)
[t-SNE] Iteration 500: error = 1.0835493, gradient norm = 0.0001437 (50 iterations in 4.633s)
[t-SNE] Iteration 550: error = 1.0471643, gradient norm = 0.0001152 (50 iterations in 4.655s)
[t-SNE] Iteration 600: error = 1.0231258, gradient norm = 0.0001007 (50 iterations in 4.669s)
[t-SNE] Iteration 650: error = 1.0069925, gradient norm = 0.0000892 (50 iterations in 4.685s)
[t-SNE] Iteration 700: error = 0.9953420, gradient norm = 0.0000804 (50 iterations in 4.715s)
[t-SNE] Iteration 750: error = 0.9866475, gradient norm = 0.0000728 (50 iterations in 4.731s)
[t-SNE] Iteration 800: error = 0.9796536, gradient norm = 0.0000658 (50 iterations in 4.723s)
[t-SNE] Iteration 850: error = 0.9737327, gradient norm = 0.0000618 (50 iterations in 4.732s)
[t-SNE] Iteration 900: error = 0.9688665, gradient norm = 0.0000594 (50 iterations in 4.733s)
[t-SNE] Iteration 950: error = 0.9644679, gradient norm = 0.0000589 (50 iterations in 4.799s)
[t-SNE] Iteration 1000: error = 0.9610358, gradient norm = 0.0000559 (50 iterations in 4.767s)
[t-SNE] Error after 1000 iterations: 0.961036
```

In [33]:

```
df = pd.DataFrame({'x':tsne2d[:,0], 'y':tsne2d[:,1] ,'label':y})

# draw the plot in appropriate place in the grid
sns.lmplot(data=df, x='x', y='y', hue='label', fit_reg=False, height=8,palette="Set1",markers=['s','o'
])
plt.title("perplexity : {} and max_iter : {}".format(30, 1000))
plt.show()
```



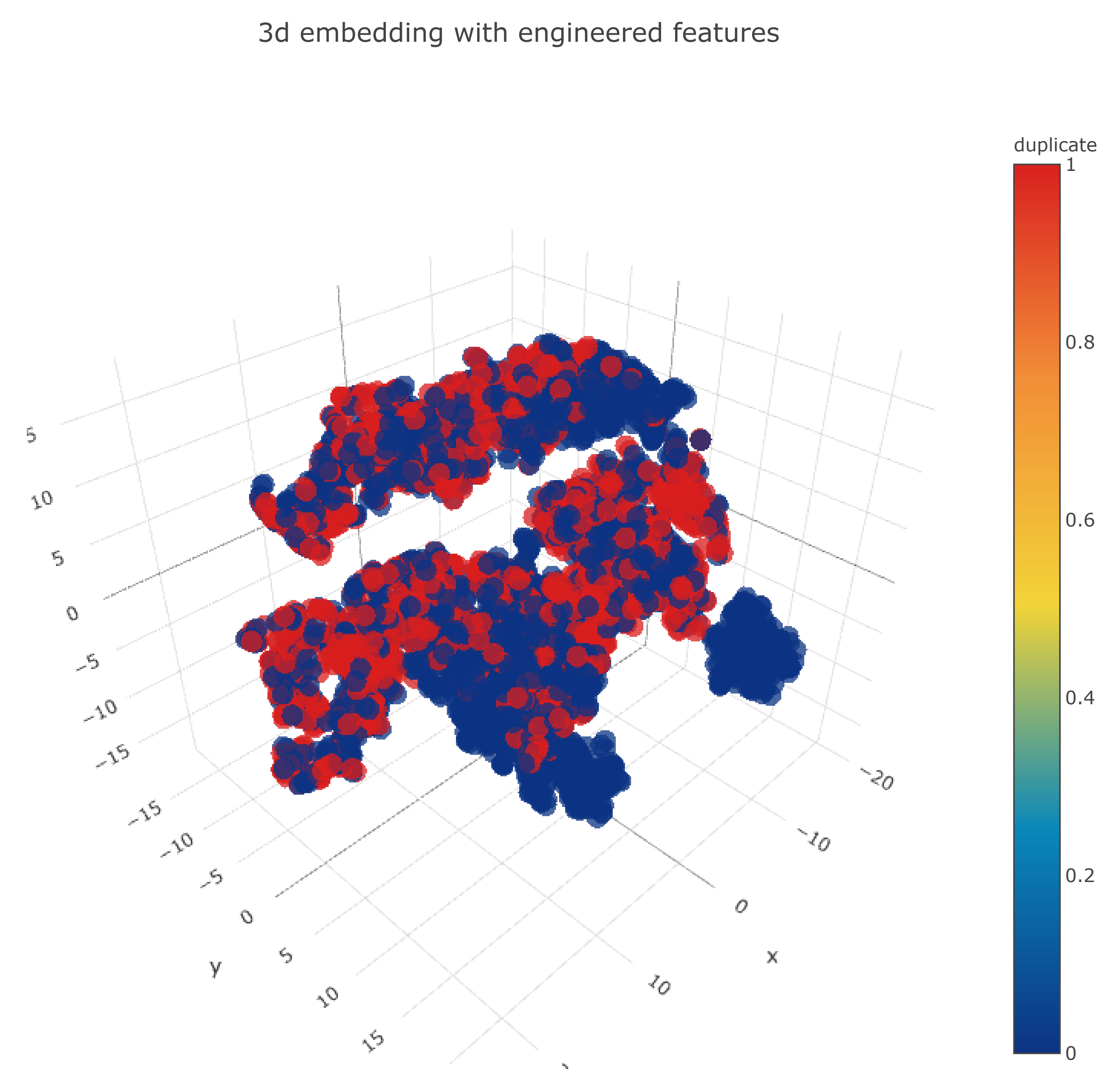
In [48]:

```
from sklearn.manifold import TSNE
tsne3d = TSNE(
    n_components=3,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 0.009s...
[t-SNE] Computed neighbors for 5000 samples in 0.320s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 0.130446
[t-SNE] Computed conditional probabilities in 0.164s
[t-SNE] Iteration 50: error = 80.5298615, gradient norm = 0.0306586 (50 iterations in 10.885s)
[t-SNE] Iteration 100: error = 69.3777008, gradient norm = 0.0037944 (50 iterations in 5.508s)
[t-SNE] Iteration 150: error = 67.9726028, gradient norm = 0.0017517 (50 iterations in 5.168s)
[t-SNE] Iteration 200: error = 67.4098892, gradient norm = 0.0013384 (50 iterations in 5.159s)
[t-SNE] Iteration 250: error = 67.0977859, gradient norm = 0.0009594 (50 iterations in 5.284s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 67.097786
[t-SNE] Iteration 300: error = 1.5276405, gradient norm = 0.0007237 (50 iterations in 6.282s)
[t-SNE] Iteration 350: error = 1.1820400, gradient norm = 0.0002119 (50 iterations in 7.505s)
[t-SNE] Iteration 400: error = 1.0407882, gradient norm = 0.0001023 (50 iterations in 7.305s)
[t-SNE] Iteration 450: error = 0.9688321, gradient norm = 0.0000652 (50 iterations in 7.368s)
[t-SNE] Iteration 500: error = 0.9303923, gradient norm = 0.0000554 (50 iterations in 7.414s)
[t-SNE] Iteration 550: error = 0.9110239, gradient norm = 0.0000524 (50 iterations in 7.275s)
[t-SNE] Iteration 600: error = 0.9016075, gradient norm = 0.0000421 (50 iterations in 7.390s)
[t-SNE] Iteration 650: error = 0.8924681, gradient norm = 0.0000360 (50 iterations in 7.363s)
[t-SNE] Iteration 700: error = 0.8837291, gradient norm = 0.0000353 (50 iterations in 7.488s)
[t-SNE] Iteration 750: error = 0.8771634, gradient norm = 0.0000316 (50 iterations in 7.425s)
[t-SNE] Iteration 800: error = 0.8718039, gradient norm = 0.0000295 (50 iterations in 7.373s)
[t-SNE] Iteration 850: error = 0.8669323, gradient norm = 0.0000276 (50 iterations in 7.479s)
[t-SNE] Iteration 900: error = 0.8628623, gradient norm = 0.0000262 (50 iterations in 7.449s)
[t-SNE] Iteration 950: error = 0.8591092, gradient norm = 0.0000241 (50 iterations in 7.294s)
[t-SNE] Iteration 1000: error = 0.8553245, gradient norm = 0.0000220 (50 iterations in 7.281s)
[t-SNE] Error after 1000 iterations: 0.855325
```

In [49]:

```
trace1 = go.Scatter3d(
    x=tsne3d[:,0],
    y=tsne3d[:,1],
    z=tsne3d[:,2],
    mode='markers',
    marker=dict(
        sizemode='diameter',
        color = y,
        colorscale = 'Portland',
        colorbar = dict(title = 'duplicate'),
        line=dict(color='rgb(255, 255, 255)'),
        opacity=0.75
    )
)

data=[trace1]
layout=dict(height=800, width=800, title='3d embedding with engineered features')
fig=dict(data=data, layout=layout)
py.iplot(fig, filename='3DBubble')
```



Export to plot.ly »

In [ ]:

3.6 Featurizing text data with tfidf weighted word-vectors

In [30]:
```
# avoid decoding problems
df = pd.read_csv("train.csv")

# encode questions to unicode
# https://stackoverflow.com/a/6812069
# ----------------- python 2 ---------------------
# df['question1'] = df['question1'].apply(lambda x: unicode(str(x),"utf-8"))
# df['question2'] = df['question2'].apply(lambda x: unicode(str(x),"utf-8"))
# ----------------- python 3 --------------------
df['question1'] = df['question1'].apply(lambda x: str(x))
df['question2'] = df['question2'].apply(lambda x: str(x))
```

In [31]:
```
df.head()
```

Out[31]:

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

In [36]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# merge texts
questions = list(df['question1']) + list(df['question2'])

tfidf = TfidfVectorizer(lowercase=False, )
tfidf.fit_transform(questions)

# dict key:word and value:tf-idf score
word2tfidf = dict(zip(tfidf.get_feature_names(), tfidf.idf_))
```

- After we find TF-IDF scores, we convert each question to a weighted average of word2vec vectors by these scores.
- here we use a pre-trained GLOVE model which comes free with "Spacy". https://spacy.io/usage/vectors-similarity
- It is trained on Wikipedia and therefore, it is stronger in terms of word semantics.

In [59]:
```python
# en_vectors_web_lg, which includes over 1 million unique vectors.
import en_core_web_sm
nlp = en_core_web_sm.load()
#nlp = spacy.load('en_core_web_sm')

vecs1 = []
# https://github.com/noamraph/tqdm
# tqdm is used to print the progress bar
for qu1 in tqdm(list(df['question1'])):
    doc1 = nlp(qu1)
    # 384 is the number of dimensions of vectors
    mean_vec1 = np.zeros([len(doc1), 384])
    for word1 in doc1:
        # word2vec
        vec1 = word1.vector
        # fetch df score
        try:
            idf = word2tfidf[str(word1)]
        except:
            idf = 0
        # compute final vec
        mean_vec1 += vec1 * idf
    mean_vec1 = mean_vec1.mean(axis=0)
    vecs1.append(mean_vec1)
df['q1_feats_m'] = list(vecs1)
```

```
100%|██████████████████████████████████████████████████| 404290/404290 [1:09:39<
00:00, 96.73it/s]
```

In [60]:
```python
vecs2 = []
for qu2 in tqdm(list(df['question2'])):
    doc2 = nlp(qu2)
    mean_vec2 = np.zeros([len(doc2), 384])
    for word2 in doc2:
        # word2vec
        vec2 = word2.vector
        # fetch df score
        try:
            idf = word2tfidf[str(word2)]
        except:
            #print word
            idf = 0
        # compute final vec
        mean_vec2 += vec2 * idf
    mean_vec2 = mean_vec2.mean(axis=0)
    vecs2.append(mean_vec2)
df['q2_feats_m'] = list(vecs2)
```

```
100%|██████████████████████████████████████████████████| 404290/404290 [1:09:01<
00:00, 97.62it/s]
```

In [32]:
```python
#prepro_features_train.csv (Simple Preprocessing Feartures)
#nlp_features_train.csv (NLP Features)
if os.path.isfile('nlp_features_train.csv'):
    dfnlp = pd.read_csv("nlp_features_train.csv",encoding='latin-1')
else:
    print("download nlp_features_train.csv from drive or run previous notebook")

if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    dfppro = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')
else:
    print("download df_fe_without_preprocessing_train.csv from drive or run previous notebook")
```

In [62]:
```python
df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3 = df.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
df3_q1 = pd.DataFrame(df3.q1_feats_m.values.tolist(), index= df3.index)
df3_q2 = pd.DataFrame(df3.q2_feats_m.values.tolist(), index= df3.index)
```

In [63]:
```python
# dataframe of nlp features
df1.head()
```

Out[63]:

	id	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_word_eq	first_word_eq	abs_len_diff	me
0	0	0	0.999980	0.833319	0.999983	0.999983	0.916659	0.785709	0.0	1.0	2.0	
1	1	0	0.799984	0.399996	0.749981	0.599988	0.699993	0.466664	0.0	1.0	5.0	
2	2	0	0.399992	0.333328	0.399992	0.249997	0.399996	0.285712	0.0	1.0	4.0	
3	3	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	2.0	
4	4	0	0.399992	0.199998	0.999950	0.666644	0.571420	0.307690	0.0	1.0	6.0	

In [64]:
```python
# data before preprocessing
df2.head()
```

Out[64]:

	id	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_Common	word_Total	word_share	freq_q1+q2	freq
0	0	1	1	66	57	14	12	10.0	23.0	0.434783	2	
1	1	4	1	51	88	8	13	4.0	20.0	0.200000	5	
2	2	1	1	73	59	14	10	4.0	24.0	0.166667	2	
3	3	1	1	50	65	11	9	0.0	19.0	0.000000	2	
4	4	3	1	76	39	13	7	2.0	20.0	0.100000	4	

In [65]:
```python
# Questions 1 tfidf weighted word2vec
df3_q1.head()
```

Out[65]:

	0	1	2	3	4	5	6	7	8	9	..
0	121.929927	100.083906	72.497900	115.641795	-48.370865	34.619070	-172.057790	-92.502626	113.223311	50.562456	..
1	-78.070935	54.843787	82.738495	98.191855	-51.234840	55.013509	-39.140733	-82.692374	45.161483	-9.556298	..
2	-5.355015	73.671810	14.376365	104.130241	1.433537	35.229116	-148.519385	-97.124595	41.972195	50.948731	..
3	5.778359	-34.712038	48.999631	59.699204	40.661263	-41.658731	-36.808594	24.170655	0.235601	-29.407290	..
4	51.138220	38.587312	123.639488	53.333041	-47.062739	37.356212	-298.722753	-106.421119	106.248914	65.880707	..

5 rows × 384 columns

In [66]:
```python
# Questions 2 tfidf weighted word2vec
df3_q2.head()
```

Out[66]:

	0	1	2	3	4	5	6	7	8	9	...
0	125.983301	95.636484	42.114717	95.449986	-37.386301	39.400084	-148.116068	-87.851481	110.371972	62.272816	...
1	-106.871899	80.290340	79.066300	59.302100	-42.175332	117.616657	-144.364242	-127.131506	22.962531	25.397579	...
2	7.072875	15.513378	1.846914	85.937583	-33.808811	94.702337	-122.256856	-114.009530	53.922293	60.131814	...
3	39.421539	44.136990	-24.010927	85.265864	-0.339028	-9.323141	-60.499653	-37.044767	49.407847	-23.350149	...
4	31.950109	62.854102	1.778147	36.218763	-45.130861	66.674880	-106.342344	-22.901031	59.835921	62.663957	...

5 rows × 384 columns

In [67]:
```python
print("Number of features in nlp dataframe :", df1.shape[1])
print("Number of features in preprocessed dataframe :", df2.shape[1])
print("Number of features in question1 w2v  dataframe :", df3_q1.shape[1])
print("Number of features in question2 w2v  dataframe :", df3_q2.shape[1])
print("Number of features in final dataframe  :", df1.shape[1]+df2.shape[1]+df3_q1.shape[1]+df3_q2.sha
pe[1])
```

```
Number of features in nlp dataframe : 17
Number of features in preprocessed dataframe : 12
Number of features in question1 w2v  dataframe : 384
Number of features in question2 w2v  dataframe : 384
Number of features in final dataframe  : 797
```

In [68]:
```python
# storing the final features to csv file
if not os.path.isfile('final_features.csv'):
    df3_q1['id']=df1['id']
    df3_q2['id']=df1['id']
    df1  = df1.merge(df2, on='id',how='left')
    df2  = df3_q1.merge(df3_q2, on='id',how='left')
    result  = df1.merge(df2, on='id',how='left')
    result.to_csv('final_features.csv')
```

4. Machine Learning Models

4.1 Reading data from file and storing into sql table

In [46]:

```python
#Creating db file from csv
from sqlalchemy import create_engine
import datetime as dt

if not os.path.isfile('train.db'):
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 180000
    j = 0
    index_start = 1
    for df in pd.read_csv('final_features.csv', names=['Unnamed: 0','id','is_duplicate','cwc_min','cwc_max','csc_min','csc_max','ctc_min','ctc_max','last_word_eq','first_word_eq','abs_len_diff','mean_len','token_set_ratio','token_sort_ratio','fuzz_ratio','fuzz_partial_ratio','longest_substr_ratio','freq_qid1','freq_qid2','q1len','q2len','q1_n_words','q2_n_words','word_Common','word_Total','word_share','freq_q1+q2','freq_q1-q2','0_x','1_x','2_x','3_x','4_x','5_x','6_x','7_x','8_x','9_x','10_x','11_x','12_x','13_x','14_x','15_x','16_x','17_x','18_x','19_x','20_x','21_x','22_x','23_x','24_x','25_x','26_x','27_x','28_x','29_x','30_x','31_x','32_x','33_x','34_x','35_x','36_x','37_x','38_x','39_x','40_x','41_x','42_x','43_x','44_x','45_x','46_x','47_x','48_x','49_x','50_x','51_x','52_x','53_x','54_x','55_x','56_x','57_x','58_x','59_x','60_x','61_x','62_x','63_x','64_x','65_x','66_x','67_x','68_x','69_x','70_x','71_x','72_x','73_x','74_x','75_x','76_x','77_x','78_x','79_x','80_x','81_x','82_x','83_x','84_x','85_x','86_x','87_x','88_x','89_x','90_x','91_x','92_x','93_x','94_x','95_x','96_x','97_x','98_x','99_x','100_x','101_x','102_x','103_x','104_x','105_x','106_x','107_x','108_x','109_x','110_x','111_x','112_x','113_x','114_x','115_x','116_x','117_x','118_x','119_x','120_x','121_x','122_x','123_x','124_x','125_x','126_x','127_x','128_x','129_x','130_x','131_x','132_x','133_x','134_x','135_x','136_x','137_x','138_x','139_x','140_x','141_x','142_x','143_x','144_x','145_x','146_x','147_x','148_x','149_x','150_x','151_x','152_x','153_x','154_x','155_x','156_x','157_x','158_x','159_x','160_x','161_x','162_x','163_x','164_x','165_x','166_x','167_x','168_x','169_x','170_x','171_x','172_x','173_x','174_x','175_x','176_x','177_x','178_x','179_x','180_x','181_x','182_x','183_x','184_x','185_x','186_x','187_x','188_x','189_x','190_x','191_x','192_x','193_x','194_x','195_x','196_x','197_x','198_x','199_x','200_x','201_x','202_x','203_x','204_x','205_x','206_x','207_x','208_x','209_x','210_x','211_x','212_x','213_x','214_x','215_x','216_x','217_x','218_x','219_x','220_x','221_x','222_x','223_x','224_x','225_x','226_x','227_x','228_x','229_x','230_x','231_x','232_x','233_x','234_x','235_x','236_x','237_x','238_x','239_x','240_x','241_x','242_x','243_x','244_x','245_x','246_x','247_x','248_x','249_x','250_x','251_x','252_x','253_x','254_x','255_x','256_x','257_x','258_x','259_x','260_x','261_x','262_x','263_x','264_x','265_x','266_x','267_x','268_x','269_x','270_x','271_x','272_x','273_x','274_x','275_x','276_x','277_x','278_x','279_x','280_x','281_x','282_x','283_x','284_x','285_x','286_x','287_x','288_x','289_x','290_x','291_x','292_x','293_x','294_x','295_x','296_x','297_x','298_x','299_x','300_x','301_x','302_x','303_x','304_x','305_x','306_x','307_x','308_x','309_x','310_x','311_x','312_x','313_x','314_x','315_x','316_x','317_x','318_x','319_x','320_x','321_x','322_x','323_x','324_x','325_x','326_x','327_x','328_x','329_x','330_x','331_x','332_x','333_x','334_x','335_x','336_x','337_x','338_x','339_x','340_x','341_x','342_x','343_x','344_x','345_x','346_x','347_x','348_x','349_x','350_x','351_x','352_x','353_x','354_x','355_x','356_x','357_x','358_x','359_x','360_x','361_x','362_x','363_x','364_x','365_x','366_x','367_x','368_x','369_x','370_x','371_x','372_x','373_x','374_x','375_x','376_x','377_x','378_x','379_x','380_x','381_x','382_x','383_x','0_y','1_y','2_y','3_y','4_y','5_y','6_y','7_y','8_y','9_y','10_y','11_y','12_y','13_y','14_y','15_y','16_y','17_y','18_y','19_y','20_y','21_y','22_y','23_y','24_y','25_y','26_y','27_y','28_y','29_y','30_y','31_y','32_y','33_y','34_y','35_y','36_y','37_y','38_y','39_y','40_y','41_y','42_y','43_y','44_y','45_y','46_y','47_y','48_y','49_y','50_y','51_y','52_y','53_y','54_y','55_y','56_y','57_y','58_y','59_y','60_y','61_y','62_y','63_y','64_y','65_y','66_y','67_y','68_y','69_y','70_y','71_y','72_y','73_y','74_y','75_y','76_y','77_y','78_y','79_y','80_y','81_y','82_y','83_y','84_y','85_y','86_y','87_y','88_y','89_y','90_y','91_y','92_y','93_y','94_y','95_y','96_y','97_y','98_y','99_y','100_y','101_y','102_y','103_y','104_y','105_y','106_y','107_y','108_y','109_y','110_y','111_y','112_y','113_y','114_y','115_y','116_y','117_y','118_y','119_y','120_y','121_y','122_y','123_y','124_y','125_y','126_y','127_y','128_y','129_y','130_y','131_y','132_y','133_y','134_y','135_y','136_y','137_y','138_y','139_y','140_y','141_y','142_y','143_y','144_y','145_y','146_y','147_y','148_y','149_y','150_y','151_y','152_y','153_y','154_y','155_y','156_y','157_y','158_y','159_y','160_y','161_y','162_y','163_y','164_y','165_y','166_y','167_y','168_y','169_y','170_y','171_y','172_y','173_y','174_y','175_y','176_y','177_y','178_y','179_y','180_y','181_y','182_y','183_y','184_y','185_y','186_y','187_y','188_y','189_y','190_y','191_y','192_y','193_y','194_y','195_y','196_y','197_y','198_y','199_y','200_y','201_y','202_y','203_y','204_y','205_y','206_y','207_y','208_y','209_y','210_y','211_y','212_y','213_y','214_y','215_y','216_y','217_y','218_y','219_y','220_y','221_y','222_y','223_y','224_y','225_y','226_y','227_y','228_y','229_y','230_y','231_y','232_y','233_y','234_y','235_y','236_y','237_y','238_y','239_y','240_y','241_y','242_y','243_y','244_y','245_y','246_y','247_y','248_y','249_y','250_y','251_y','252_y','253_y','254_y','255_y','256_y','257_y','258_y','259_y','260_y','261_y','262_y','263_y','264_y','265_y','266_y','267_y','268_y','269_y','270_y','271_y','272_y','273_y','274_y','275_y','276_y','277_y','278_y','279_y','280_y','281_y','282_y','283_y','284_y','285_y','286_y','287_y','288_y','289_y','290_y','291_y','292_y','293_y','294_y','295_y','296_y','297_y','298_y','299_y','300_y','301_y','302_y','303_y','304_y','305_y','306_y','307_y','308_y','309_y','310_y','311_y','312_y','313_y','314_y','315_y','316_y','317_y','318_y','319_y','320_y','321_y','322_y','323_y','324_y','325_y','326_y','327_y','328_y','329_y','330_y','331_y','332_y','333_y','334_y','335_y','336_y','337_y','338_y','339_y','340_y','341_y','342_y','343_y','344_y','345_y','346_y','347_y','348_y','349_y','350_y','351_y','352_y','353_y','354_y','355_y','356_y','357_y','358_y','359_y','360_y','361_y','362_y','363_y','364_y','365_y','366_y','367_y','368_y','369_y','370_y','371_y','372_y','373_y','374_y','375_y','376_y','377_y','378_y','379_y','380_y','381_y','382_y','383_y'], chunksize=chunksize, iterator=True, encoding='utf-8', ):
        df.index += index_start
        j+=1
        print('{} rows'.format(j*chunksize))
        df.to_sql('data', disk_engine, if_exists='append')
        index_start = df.index[-1] + 1
```

In [47]:
```
#http://www.sqlitetutorial.net/sqlite-python/create-tables/
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None


def checkTableExists(dbcon):
    cursr = dbcon.cursor()
    str = "select name from sqlite_master where type='table'"
    table_names = cursr.execute(str)
    print("Tables in the databse:")
    tables =table_names.fetchall()
    print(tables[0][0])
    return(len(tables))
```

In [78]:
```
read_db = 'train.db'
conn_r = create_connection(read_db)
checkTableExists(conn_r)
conn_r.close()
```

```
Tables in the databse:
data
```

In [79]:
```
# try to sample data according to the computing power you have
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        # for selecting first 1M rows
        # data = pd.read_sql_query("""SELECT * FROM data LIMIT 100001;""", conn_r)

        # for selecting random points
        data = pd.read_sql_query("SELECT * From data ORDER BY RANDOM() LIMIT 100001;", conn_r)
        conn_r.commit()
        conn_r.close()
```

In [80]:
```
# remove the first row
data.drop(data.index[0], inplace=True)
y_true = data['is_duplicate']
data.drop(['Unnamed: 0', 'id','index','is_duplicate'], axis=1, inplace=True)
```

In [81]:
```
data.head()
```

Out[81]:

	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_
1	0.33332222259258	0.199996000079998	0.499987500312492	0.399992000159997	0.428565306209911	0.29999700003	
2	0.749981250468738	0.749981250468738	0.749981250468738	0.749981250468738	0.749990625117186	0.749990625117186	
3	0.555549382784636	0.49999500005	0.999975000624984	0.666655555740738	0.69230236690487	0.562496484396973	
4	0.399992000159997	0.399992000159997	0.249993750156246	0.199996000079998	0.29999700003	0.29999700003	
5	0.14285510206997	0.0999990000099999	0.666661111157407	0.470585467144311	0.44999775001125	0.321427423473488	

5 rows × 794 columns

4.2 Converting strings to numerics

In [82]:
```
# after we read from sql table each entry was read it as a string
# we convert all the features into numaric before we apply any model
cols = list(data.columns)
for i in cols:
    data[i] = data[i].apply(pd.to_numeric)
    print(i)
```

```
cwc_min
cwc_max
csc_min
csc_max
ctc_min
ctc_max
last_word_eq
first_word_eq
abs_len_diff
mean_len
token_set_ratio
token_sort_ratio
fuzz_ratio
fuzz_partial_ratio
longest_substr_ratio
freq_qid1
freq_qid2
q1len
q2len
q1_n_words
q2_n_words
word_Common
word_Total
word_share
freq_q1+q2
freq_q1-q2
0_x
1_x
2_x
3_x
4_x
5_x
6_x
7_x
8_x
9_x
10_x
11_x
12_x
13_x
14_x
15_x
16_x
17_x
18_x
19_x
20_x
21_x
22_x
23_x
24_x
25_x
26_x
27_x
28_x
29_x
30_x
31_x
32_x
33_x
34_x
35_x
36_x
37_x
38_x
39_x
40_x
41_x
42_x
43_x
44_x
45_x
46_x
47_x
48_x
49_x
50_x
51_x
52_x
53_x
54_x
55_x
56_x
57_x
58_x
59_x
60_x
```

322_x
323_x
324_x
325_x
326_x
327_x
328_x
329_x
330_x
331_x
332_x
333_x
334_x
335_x
336_x
337_x
338_x
339_x
340_x
341_x
342_x
343_x
344_x
345_x
346_x
347_x
348_x
349_x
350_x
351_x
352_x
353_x
354_x
355_x
356_x
357_x
358_x
359_x
360_x
361_x
362_x
363_x
364_x
365_x
366_x
367_x
368_x
369_x
370_x
371_x
372_x
373_x
374_x
375_x
376_x
377_x
378_x
379_x
380_x
381_x
382_x
383_x
0_y
1_y
2_y
3_y
4_y
5_y
6_y
7_y
8_y
9_y
10_y
11_y
12_y
13_y
14_y
15_y
16_y
17_y
18_y
19_y
20_y
21_y
22_y
23_y
24_y

```
373_y
374_y
375_y
376_y
377_y
378_y
379_y
380_y
381_y
382_y
383_y
```

In [83]:
```
# https://stackoverflow.com/questions/7368789/convert-all-strings-in-a-list-to-int
y_true = list(map(int, y_true.values))
```

4.3 Random train test split(70:30)

In [85]:
```
from sklearn.cross_validation import train_test_split
X_train,X_test, y_train, y_test = train_test_split(data, y_true, stratify=y_true, test_size=0.3)
```

In [86]:
```
print("Number of data points in train data :",X_train.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (70000, 794)
Number of data points in test data : (30000, 794)
```

In [19]:
```
from collections import Counter
print("-"*10, "Distribution of output variable in train data", "-"*10)
train_distr = Counter(y_train)
train_len = len(y_train)
print("Class 0: ",int(train_distr[0])/train_len,"Class 1: ", int(train_distr[1])/train_len)
print("-"*10, "Distribution of output variable in train data", "-"*10)
test_distr = Counter(y_test)
test_len = len(y_test)
print("Class 0: ",int(test_distr[1])/test_len, "Class 1: ",int(test_distr[1])/test_len)
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.6308025003268517 Class 1:  0.36919749967314835
---------- Distribution of output variable in train data ----------
Class 0:  0.3691986775169639 Class 1:  0.3691986775169639
```

In [16]:

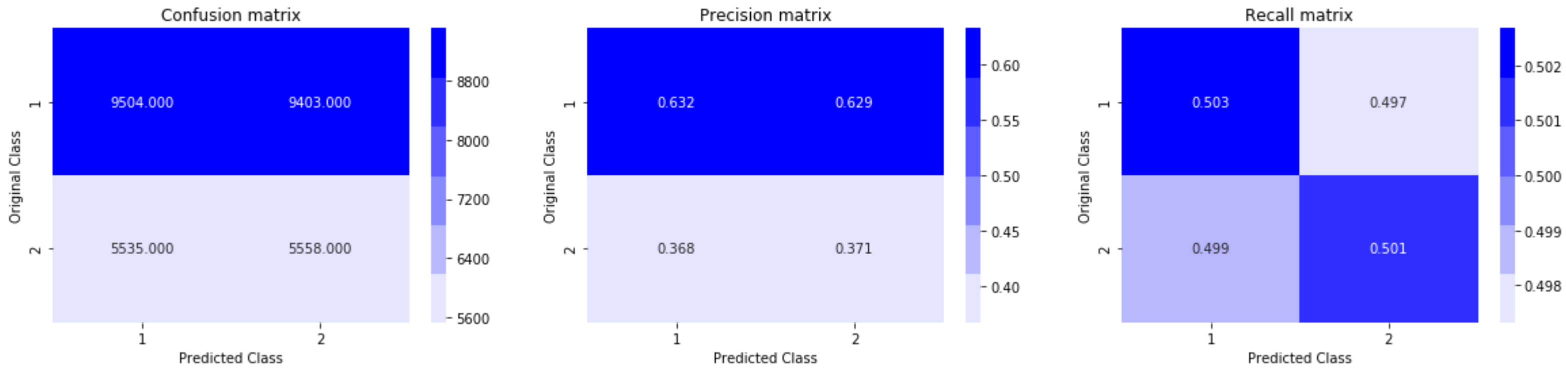
```
# This function plots the confusion matrices given y_i, y_i_hat.
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    # C = 9,9 matrix, each cell (i,j) represents number of points of class i are predicted class j

    A =(((C.T)/(C.sum(axis=1))).T)
    #divid each element of the confusion matrix with the sum of elements in that column

    # C = [[1, 2],
    #     [3, 4]]
    # C.T = [[1, 3],
    #        [2, 4]]
    # C.sum(axis = 1)  axis=0 corresonds to columns and axis=1 corresponds to rows in two diamensional
 array
    # C.sum(axix =1) = [[3, 7]]
    # ((C.T)/(C.sum(axis=1))) = [[1/3, 3/7]
    #                            [2/3, 4/7]]

    # ((C.T)/(C.sum(axis=1))).T = [[1/3, 2/3]
    #                            [3/7, 4/7]]
    # sum of row elements = 1

    B =(C/C.sum(axis=0))
    #divid each element of the confusion matrix with the sum of elements in that row
    # C = [[1, 2],
    #     [3, 4]]
    # C.sum(axis = 0)  axis=0 corresonds to columns and axis=1 corresponds to rows in two diamensional
 array
    # C.sum(axix =0) = [[4, 6]]
    # (C/C.sum(axis=0)) = [[1/4, 2/6],
    #                       [3/4, 4/6]]
    plt.figure(figsize=(20,4))

    labels = [1,2]
    # representing A in heatmap format
    cmap=sns.light_palette("blue")
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")

    plt.show()
```

4.4 Building a random model (Finding worst-case log-loss)

In [95]:

```
# we need to generate 9 numbers and the sum of numbers should be 1
# one solution is to genarate 9 numbers and divide each of the numbers by their sum
# ref: https://stackoverflow.com/a/18662466/4084039
# we create a output array that has exactly same size as the CV data
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.metrics import confusion_matrix
predicted_y = np.zeros((test_len,2))
for i in range(test_len):
    rand_probs = np.random.rand(1,2)
    predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Test Data using Random Model",log_loss(y_test, predicted_y, eps=1e-15))

predicted_y =np.argmax(predicted_y, axis=1)
plot_confusion_matrix(y_test, predicted_y)
```

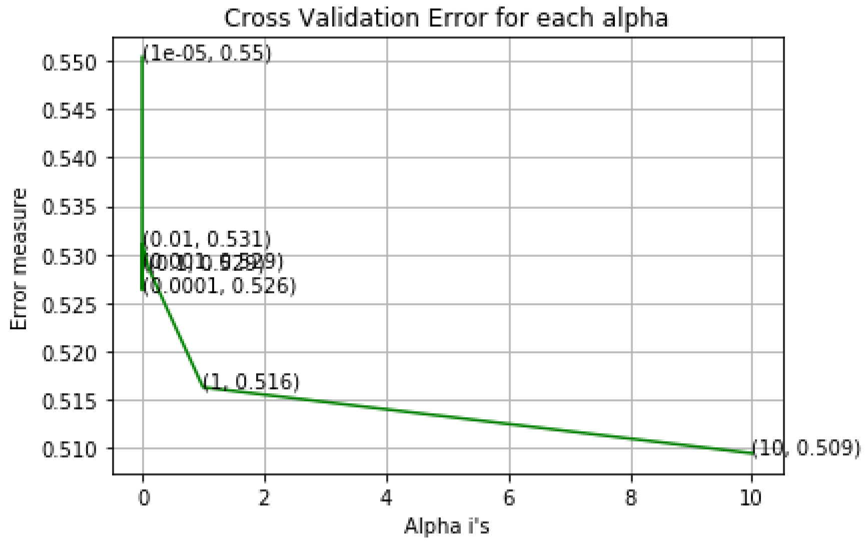
```
Log loss on Test Data using Random Model 0.8859844162334732
```



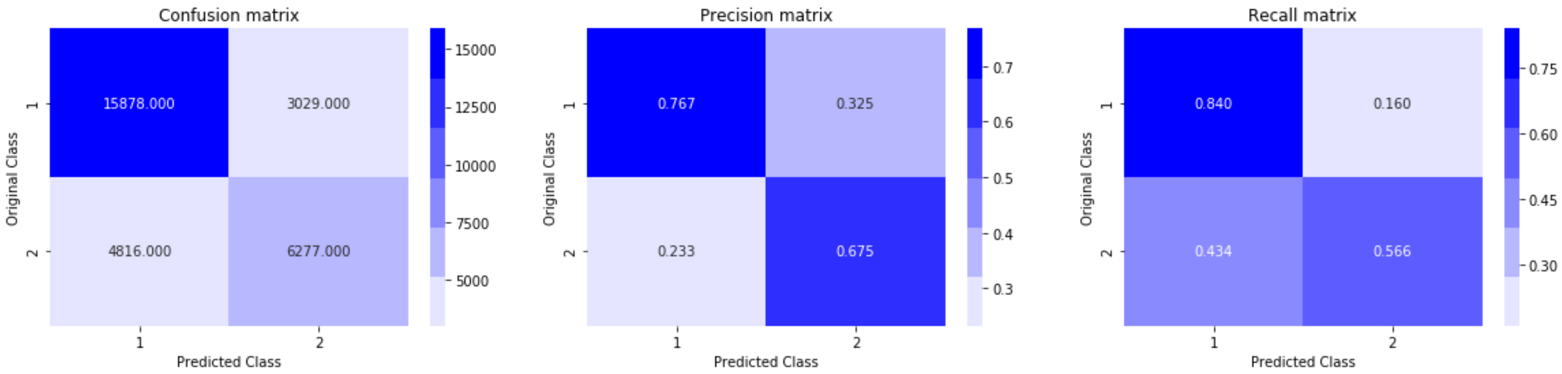
4.4 Logistic Regression with hyperparameter tuning

In [99]:
```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_m
odel.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=
None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.
0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, …])     Fit linear model with Stochastic Gradient Descent.
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link:
#------------------------------
from sklearn.calibration import CalibratedClassifierCV
from sklearn.linear_model import SGDClassifier
log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_test, predict_y, labels=clf.class
es_, eps=1e-15))

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pre
dict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predi
ct_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.550361958798781
For values of alpha =  0.0001 The log loss is: 0.5263632304462008
For values of alpha =  0.001 The log loss is: 0.5287429057021276
For values of alpha =  0.01 The log loss is: 0.5310350364575331
For values of alpha =  0.1 The log loss is: 0.528652877124861
For values of alpha =  1 The log loss is: 0.5162935110625999
For values of alpha =  10 The log loss is: 0.5094765307097092
```




```
For values of best alpha =  10 The train log loss is: 0.5049944197626971
For values of best alpha =  10 The test log loss is: 0.5094765307097092
Total number of data points : 30000
```

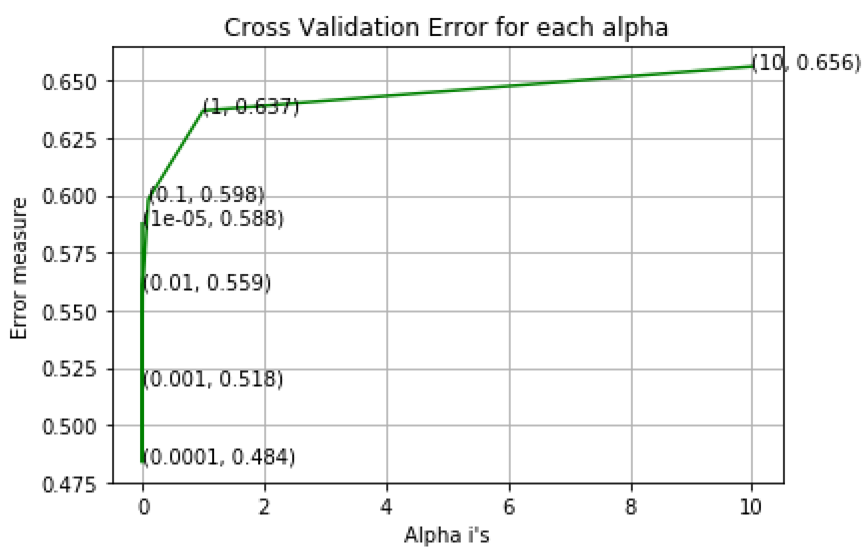


4.5 Linear SVM with hyperparameter tuning

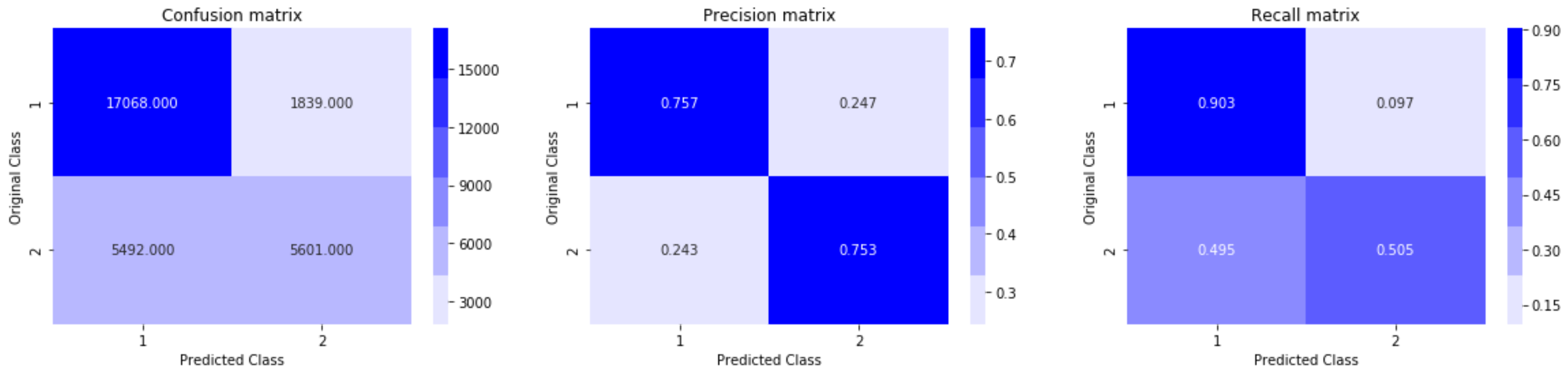
In [100]:

```python
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_m
odel.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=
None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.
0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, …])     Fit linear model with Stochastic Gradient Descent.
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link:
#------------------------------


log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l1', loss='hinge', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_test, predict_y, labels=clf.class
es_, eps=1e-15))

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l1', loss='hinge', random_state=42)
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pre
dict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predi
ct_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.5876837148203761
For values of alpha =  0.0001 The log loss is: 0.4837927769511895
For values of alpha =  0.001 The log loss is: 0.5181150652746598
For values of alpha =  0.01 The log loss is: 0.5594917068727758
For values of alpha =  0.1 The log loss is: 0.5981617907378611
For values of alpha =  1 The log loss is: 0.6368633892250878
For values of alpha =  10 The log loss is: 0.6561008491301867
```




```
For values of best alpha =  0.0001 The train log loss is: 0.4765354376982544
For values of best alpha =  0.0001 The test log loss is: 0.4837927769511895
Total number of data points : 30000
```



Observations

- The Test Log-loss of both the Logistic-regression and Linear-SVM after Hyper-parameter tuning is 0.509 and 0.4837.
- Since both the log-loss values are smaller than the simple random-model so both the model are sensible and doing a good-job.
- But still by studying the performance metrics the precision scores are good but the real problem is in the recall scores because it is quite low for the minority class-labels which decreases the log-loss values of the model.
- I think the model is facing with the bias problem since linear-model has some bias problem.
- By training a XG-Boost model the issue can be solved and let's try next.

4.6 XGBoost on the 794-Dim Data

In [101]:
```
import xgboost as xgb
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 4

d_train = xgb.DMatrix(X_train, label=y_train)
d_test = xgb.DMatrix(X_test, label=y_test)

watchlist = [(d_train, 'train'), (d_test, 'valid')]

bst = xgb.train(params, d_train, 400, watchlist, early_stopping_rounds=20, verbose_eval=10)

xgdmat = xgb.DMatrix(X_train,y_train)
predict_y = bst.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

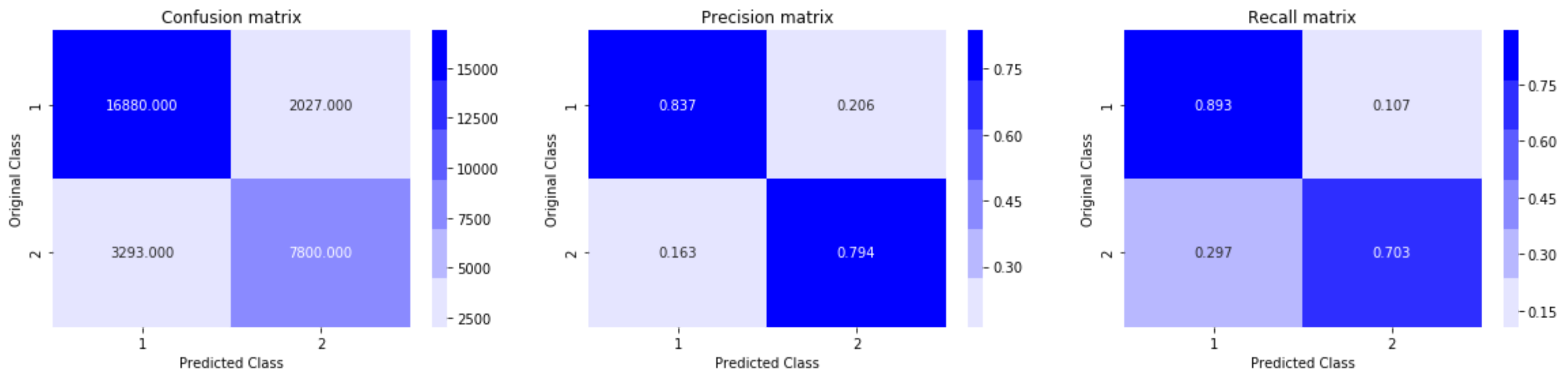
```
[22:02:21] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[0]	train-logloss:0.684828	valid-logloss:0.68489
Multiple eval metrics have been passed: 'valid-logloss' will be used for early stopping.

Will train until valid-logloss hasn't improved in 20 rounds.
[22:02:22] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:23] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:23] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:26] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:27] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:27] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:28] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[10]	train-logloss:0.615272	valid-logloss:0.615705
[22:02:29] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:30] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:30] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:31] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:32] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:32] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:33] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:34] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:34] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:35] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[20]	train-logloss:0.563801	valid-logloss:0.564516
[22:02:36] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:37] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:37] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:38] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:39] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:39] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:40] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:41] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:42] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:42] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[30]	train-logloss:0.524864	valid-logloss:0.525727
[22:02:43] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:44] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:44] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:45] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:46] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:47] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:47] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:48] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:02:49] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
```

```
[22:06:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:06:52] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 18 extra nodes, 0 pruned nodes, max_depth=4
[22:06:53] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:06:53] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:06:54] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:06:55] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 22 extra nodes, 0 pruned nodes, max_depth=4
[22:06:56] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:06:56] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:06:57] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 18 extra nodes, 0 pruned nodes, max_depth=4
[22:06:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[380]	train-logloss:0.344842	valid-logloss:0.354663
[22:06:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:06:59] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:00] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:01] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:02] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 20 extra nodes, 0 pruned nodes, max_depth=4
[22:07:02] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:03] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:04] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 18 extra nodes, 0 pruned nodes, max_depth=4
[22:07:05] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:06] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 22 extra nodes, 0 pruned nodes, max_depth=4
[390]	train-logloss:0.343843	valid-logloss:0.353988
[22:07:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 18 extra nodes, 0 pruned nodes, max_depth=4
[22:07:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:10] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 22 extra nodes, 0 pruned nodes, max_depth=4
[22:07:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22:07:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[399]	train-logloss:0.342916	valid-logloss:0.353375
The test log loss is: 0.35337548070589936
```

In [102]:

```python
predicted_y =np.array(predict_y>0.5,dtype=int)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
Total number of data points : 30000
```



Observation:-

- The Test log-loss of the XG-Boost model is 0.3533 which is much lower than the previous linear-model.
- By studying the above performance matrixes the precison and recall scores are much better than the previous linear model

and the bias problem is not there in this boosted classiffier technique.
- The performance of the above XGB-Classiffier can be improved by tuning the different hyper-parameters of the XGB-Classiffier.

Tuning the hyperparameters by using the random search technique

In [24]:
```
#UTILITY FUNCTION FOR TUNING THE Xgb-Classiffier
def Randomsearch_tuning(xgb_model,param,x_tr,y_tr):

    model = xgb_model

    random_param=param

    random_search=RandomizedSearchCV(model,param_distributions=random_param,verbose=2,n_jobs=-1,cv=5,s
coring='neg_log_loss')
    random_result = random_search.fit(x_tr,y_tr)
    # summarize results
    print("Best: %f using %s" % (random_result.best_score_, random_result.best_params_))
    means = random_result.cv_results_['mean_test_score']
    stds = random_result.cv_results_['std_test_score']
    params = random_result.cv_results_['params']
    for mean, stdev, param in zip(means, stds, params):
        print("%f (%f) with: %r" % (mean, stdev, param))
```

In [107]:
```
%%time
#Listing the paramaeters range that has to be tuned via random-search technique
prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,1000,1500],
    'colsample_bytree':[0.1,0.3,0.5,1],
    'subsample':[0.1,0.3,0.5,1]
}

#Specifing the model to be tuned
model=xgb.XGBClassifier(n_jobs=-1, random_state=15)
#Calling the Utility function
Randomsearch_tuning(model,prams,X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed: 66.4min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 78.5min finished
```

```
Best: -0.332875 using {'subsample': 1, 'n_estimators': 1500, 'learning_rate': 0.1, 'colsample_bytree':
0.1}
-0.386474 (0.003074) with: {'subsample': 0.1, 'n_estimators': 100, 'learning_rate': 0.1, 'colsample_by
tree': 0.1}
-0.353427 (0.002653) with: {'subsample': 0.5, 'n_estimators': 1500, 'learning_rate': 0.01, 'colsample_
bytree': 0.3}
-0.338878 (0.003623) with: {'subsample': 0.3, 'n_estimators': 1500, 'learning_rate': 0.03, 'colsample_
bytree': 1}
-0.343702 (0.004287) with: {'subsample': 0.3, 'n_estimators': 1000, 'learning_rate': 0.1, 'colsample_b
ytree': 0.5}
-0.412512 (0.001240) with: {'subsample': 0.5, 'n_estimators': 100, 'learning_rate': 0.03, 'colsample_b
ytree': 0.3}
-0.332875 (0.004308) with: {'subsample': 1, 'n_estimators': 1500, 'learning_rate': 0.1, 'colsample_byt
ree': 0.1}
-0.444550 (0.001337) with: {'subsample': 0.1, 'n_estimators': 200, 'learning_rate': 0.01, 'colsample_b
ytree': 0.3}
-0.341921 (0.004042) with: {'subsample': 1, 'n_estimators': 200, 'learning_rate': 0.2, 'colsample_bytr
ee': 0.3}
-0.384258 (0.005844) with: {'subsample': 0.1, 'n_estimators': 200, 'learning_rate': 0.2, 'colsample_by
tree': 0.5}
-0.380463 (0.002636) with: {'subsample': 0.1, 'n_estimators': 500, 'learning_rate': 0.01, 'colsample_b
ytree': 1}
Wall time: 1h 20min 42s
```

In [110]:

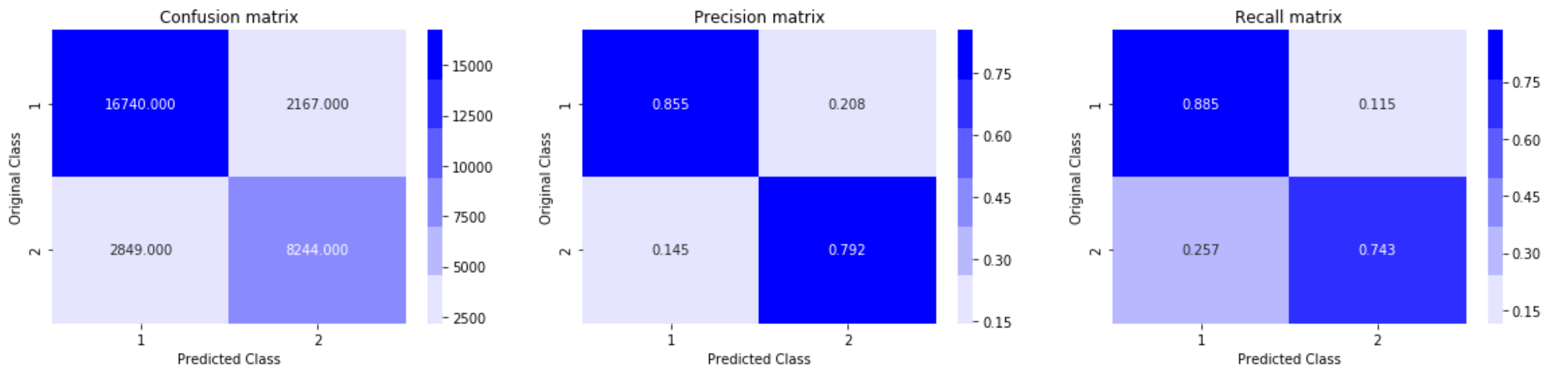
```
#Trying the Tuned parameters value
params1 = {}
params1['objective'] = 'binary:logistic'
params1['eval_metric'] = 'logloss'
params1['eta'] = 0.02
params1['max_depth'] = 3
params1["subsample"]=1
params1["n_estimators"]=1500
params1["learning_rate"]=0.1
params1["colsample_bytree"]=0.1
params1["n_jobs"]=-1

bst1 = xgb.train(params1, d_train, 400, watchlist, early_stopping_rounds=20, verbose_eval=2)

xgdmat1 = xgb.DMatrix(X_train,y_train)
predict_y1 = bst1.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y1, labels=clf.classes_, eps=1e-15))
```

```
[07:33:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[0]     train-logloss:0.660507  valid-logloss:0.660584
Multiple eval metrics have been passed: 'valid-logloss' will be used for early stopping.

Will train until valid-logloss hasn't improved in 20 rounds.
[07:33:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[2]     train-logloss:0.620908  valid-logloss:0.621258
[07:33:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[4]     train-logloss:0.601117  valid-logloss:0.601732
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[6]     train-logloss:0.577549  valid-logloss:0.578502
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[8]     train-logloss:0.54957   valid-logloss:0.550842
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[10]    train-logloss:0.532589  valid-logloss:0.534072
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:25] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[12]    train-logloss:0.524214  valid-logloss:0.525782
[07:33:26] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:27] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[14]    train-logloss:0.507299  valid-logloss:0.508993
[07:33:27] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:28] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[16]    train-logloss:0.500018  valid-logloss:0.501841
[07:33:28] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:29] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[18]    train-logloss:0.486402  valid-logloss:0.488116
[07:33:29] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:29] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[20]    train-logloss:0.477586  valid-logloss:0.479338
[07:33:30] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:30] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[22]    train-logloss:0.469546  valid-logloss:0.471355
[07:33:30] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:31] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[24]    train-logloss:0.46534   valid-logloss:0.467276
[07:33:31] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:31] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[26]    train-logloss:0.458479  valid-logloss:0.460536
[07:33:31] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:31] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[28]    train-logloss:0.453741  valid-logloss:0.455892
[07:33:32] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:32] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[30]    train-logloss:0.44825   valid-logloss:0.450753
[07:33:32] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:33:32] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[32]    train-logloss:0.446538  valid-logloss:0.449239
[07:33:32] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
```

```
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[382]	train-logloss:0.316461	valid-logloss:0.340449
[07:34:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[384]	train-logloss:0.31604	valid-logloss:0.340084
[07:34:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[386]	train-logloss:0.315862	valid-logloss:0.340047
[07:34:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[388]	train-logloss:0.315592	valid-logloss:0.339923
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[390]	train-logloss:0.31535	valid-logloss:0.339828
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[392]	train-logloss:0.315129	valid-logloss:0.339797
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[394]	train-logloss:0.31493	valid-logloss:0.339729
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[396]	train-logloss:0.314703	valid-logloss:0.339684
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[398]	train-logloss:0.314503	valid-logloss:0.339596
[07:34:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[399]	train-logloss:0.314364	valid-logloss:0.339525
The test log loss is: 0.3395247666180655
```

In [116]:
```
predicted_y1 =np.array(predict_y1>0.5,dtype=int)
print("Total number of data points :", len(predicted_y1))
plot_confusion_matrix(y_test, predicted_y1)
```

```
Total number of data points : 30000
```



Tuning the next set of Hyper-parameters of the classiffier

In [112]:
```
%%time
prams2={
 'max_depth':[1,2,3,4,5],
 'min_child_weight':[2,4,6,8],
 'gamma':[i/10.0 for i in range(0,5)],
 'reg_alpha':[1e-5, 1e-2, 0.1, 1, 100],
}

model_2=xgb.XGBClassifier(n_jobs=-1, random_state=15,subsample= 1,n_estimators= 1500,learning_rate= 0.
1,colsample_bytree= 0.1)

Randomsearch_tuning(model_2,prams2,X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed: 56.5min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 80.7min finished
```

```
Best: -0.327601 using {'reg_alpha': 1, 'min_child_weight': 8, 'max_depth': 4, 'gamma': 0.4}
-0.327601 (0.005709) with: {'reg_alpha': 1, 'min_child_weight': 8, 'max_depth': 4, 'gamma': 0.4}
-0.331124 (0.005430) with: {'reg_alpha': 0.1, 'min_child_weight': 4, 'max_depth': 5, 'gamma': 0.4}
-0.330916 (0.006394) with: {'reg_alpha': 0.1, 'min_child_weight': 4, 'max_depth': 5, 'gamma': 0.0}
-0.357525 (0.003574) with: {'reg_alpha': 100, 'min_child_weight': 8, 'max_depth': 2, 'gamma': 0.3}
-0.361588 (0.003387) with: {'reg_alpha': 0.01, 'min_child_weight': 8, 'max_depth': 1, 'gamma': 0.2}
-0.328009 (0.006139) with: {'reg_alpha': 1, 'min_child_weight': 8, 'max_depth': 5, 'gamma': 0.1}
-0.350658 (0.003792) with: {'reg_alpha': 100, 'min_child_weight': 8, 'max_depth': 5, 'gamma': 0.4}
-0.329129 (0.005924) with: {'reg_alpha': 1, 'min_child_weight': 4, 'max_depth': 5, 'gamma': 0.3}
-0.361673 (0.003370) with: {'reg_alpha': 1, 'min_child_weight': 4, 'max_depth': 1, 'gamma': 0.1}
-0.339037 (0.004080) with: {'reg_alpha': 0.1, 'min_child_weight': 8, 'max_depth': 2, 'gamma': 0.0}
Wall time: 1h 23min 33s
```

In [114]:

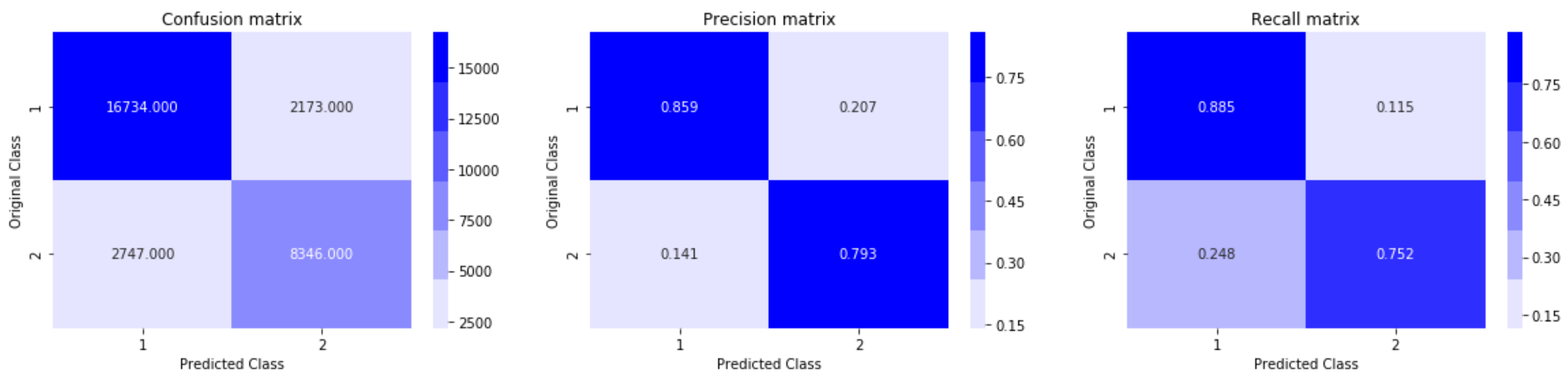
```
#Applying the Tuned parameters value and training the final XGB-Classiffier
params2 = {}
params2['objective'] = 'binary:logistic'
params2['eval_metric'] = 'logloss'
params2['eta'] = 0.02
params2['max_depth'] = 3
params2["subsample"]=1
params2["n_estimators"]=1500
params2["learning_rate"]=0.1
params2["colsample_bytree"]=0.1
params2["n_jobs"]=-1
params2['reg_alpha'] = 1
params2['min_child_weight'] = 8
params2['max_depth'] = 4
params2['gamma'] =  0.4

bst2 = xgb.train(params2, d_train, 400, watchlist, early_stopping_rounds=20, verbose_eval=2)

xgdmat2 = xgb.DMatrix(X_train,y_train)
predict_y2 = bst2.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y2, labels=clf.classes_, eps=1e-15))
```

```
[09:31:46] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[0]	train-logloss:0.658423	valid-logloss:0.658567
Multiple eval metrics have been passed: 'valid-logloss' will be used for early stopping.

Will train until valid-logloss hasn't improved in 20 rounds.
[09:31:49] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 0 pruned nodes, max_depth=4
[09:31:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 0 pruned nodes, max_depth=4
[2]	train-logloss:0.616772	valid-logloss:0.617403
[09:31:52] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:31:54] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[4]	train-logloss:0.596177	valid-logloss:0.597315
[09:31:55] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[09:31:56] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[6]	train-logloss:0.571956	valid-logloss:0.573676
[09:31:57] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[09:31:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[8]	train-logloss:0.541643	valid-logloss:0.543765
[09:31:59] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 0 pruned nodes, max_depth=4
[09:32:00] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[10]	train-logloss:0.523021	valid-logloss:0.525467
[09:32:01] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[09:32:02] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[12]	train-logloss:0.514388	valid-logloss:0.51709
[09:32:03] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:03] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[14]	train-logloss:0.496734	valid-logloss:0.499526
[09:32:04] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[09:32:04] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[16]	train-logloss:0.488668	valid-logloss:0.49161
[09:32:04] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[09:32:05] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[18]	train-logloss:0.475181	valid-logloss:0.478192
[09:32:05] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 2 pruned nodes, max_depth=4
[09:32:06] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[20]	train-logloss:0.466329	valid-logloss:0.469471
[09:32:06] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:06] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 26 extra nodes, 0 pruned nodes, max_depth=4
[22]	train-logloss:0.458077	valid-logloss:0.461481
[09:32:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 0 pruned nodes, max_depth=4
[09:32:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[24]	train-logloss:0.453541	valid-logloss:0.457168
[09:32:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 2 pruned nodes, max_depth=4
[26]	train-logloss:0.446119	valid-logloss:0.450041
[09:32:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 22 extra nodes, 2 pruned nodes, max_depth=4
[09:32:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[28]	train-logloss:0.441043	valid-logloss:0.445123
[09:32:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[30]	train-logloss:0.43549	valid-logloss:0.439903
[09:32:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[32]	train-logloss:0.433587	valid-logloss:0.438267
[09:32:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
```

```
ots, 22 extra nodes, 0 pruned nodes, max_depth=4
[09:32:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 0 pruned nodes, max_depth=4
[382]	train-logloss:0.28256	valid-logloss:0.332637
[09:32:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[384]	train-logloss:0.282087	valid-logloss:0.332397
[09:32:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 26 extra nodes, 0 pruned nodes, max_depth=4
[386]	train-logloss:0.281746	valid-logloss:0.332338
[09:32:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:50] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[388]	train-logloss:0.281368	valid-logloss:0.332254
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[390]	train-logloss:0.281042	valid-logloss:0.332265
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 20 extra nodes, 0 pruned nodes, max_depth=4
[392]	train-logloss:0.280764	valid-logloss:0.332264
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[394]	train-logloss:0.280423	valid-logloss:0.33228
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 26 extra nodes, 0 pruned nodes, max_depth=4
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 26 extra nodes, 0 pruned nodes, max_depth=4
[396]	train-logloss:0.280143	valid-logloss:0.332218
[09:32:51] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 20 extra nodes, 0 pruned nodes, max_depth=4
[09:32:52] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[398]	train-logloss:0.27986	valid-logloss:0.332146
[09:32:52] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[399]	train-logloss:0.279652	valid-logloss:0.332067
The test log loss is: 0.33206693717478364
```

In [115]:
```python
predicted_y2 =np.array(predict_y2>0.5,dtype=int)
print("Total number of data points :", len(predicted_y2))
plot_confusion_matrix(y_test, predicted_y2)
```

```
Total number of data points : 30000
```



Observations

- The final test log-loss of the Xgb-Classiffier is 0.3320 which is a significant improvement than the untuned Xgb-Classiffier.
- So here after tuning 8 parameters there is a decrease of 0.0213 log-loss is seen which is good for a classiffier.
- The Precsion and recall of the model is also improved which make the model even reliable than the simple linear models.

Implementing the Tfidf-Vectorization technique over the Dataset

In [2]:
```
#Reading the final_features data frame
df_nlp = pd.read_csv("nlp_features_train.csv",encoding='latin-1')
df_ppro = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')

data_1 = df_nlp.drop(['qid1','qid2'],axis=1)
data_2 = df_ppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)

merge_df  = data_1.merge(data_2, on='id',how='left')
```

In [3]:
```
merge_df.head(5)
```

Out[3]:

	id	question1	question2	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	...	freq_qid2	q1len
0	0	what is the step by step guide to invest in sh...	what is the step by step guide to invest in sh...	0	0.999980	0.833319	0.999983	0.999983	0.916659	0.785709	...	1	66
1	1	what is the story of kohinoor koh i noor dia...	what would happen if the indian government sto...	0	0.799984	0.399996	0.749981	0.599988	0.699993	0.466664	...	1	51
2	2	how can i increase the speed of my internet co...	how can internet speed be increased by hacking...	0	0.399992	0.333328	0.399992	0.249997	0.399996	0.285712	...	1	73
3	3	why am i mentally very lonely how can i solve...	find the remainder when math 23 24 math i...	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1	50
4	4	which one dissolve in water quikly sugar salt...	which fish would survive in salt water	0	0.399992	0.199998	0.999950	0.666644	0.571420	0.307690	...	1	76

5 rows × 30 columns

In [4]:
```
#Removing the first row
#data.drop(data.index[0], inplace=True)
labels = merge_df['is_duplicate']
merge_df.drop(['id','is_duplicate'], axis=1, inplace=True)
```

Splitting the train test split(70:30) randomly

In [5]:
```
X_train,X_test, y_Train, y_Test = train_test_split(merge_df, labels, stratify=labels, test_size=0.3)
```

In [6]:
```
print("Number of data points in train data :",X_train.shape)
print("Number of data points in test data :",X_test.shape)
```

```
Number of data points in train data : (283003, 28)
Number of data points in test data : (121287, 28)
```

In [8]:
```
from collections import Counter
print("-"*10, "Distribution of output variable in train data", "-"*10)
Train_distr = Counter(y_Train)
Train_len = len(y_Train)
print("Class 0: ",int(Train_distr[0])/Train_len,"Class 1: ", int(Train_distr[1])/Train_len)
print("-"*10, "Distribution of output variable in test data", "-"*10)
Test_distr = Counter(y_Test)
Test_len = len(y_Test)
print("Class 0: ",int(Test_distr[1])/Test_len, "Class 1: ",int(Test_distr[1])/Test_len)
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.6308025003268517 Class 1:  0.36919749967314835
---------- Distribution of output variable in test data ----------
Class 0:  0.3691986775169639 Class 1:  0.3691986775169639
```

Vectorizing the data by using TF-IDF vectorization technique

In [9]:
```
#Tfidf Vectorizer for Question-1
q1_tfidf = TfidfVectorizer()

train_q1 = q1_tfidf.fit_transform(X_train['question1'].values.astype('U'))
test_q1 = q1_tfidf.transform(X_test['question1'].values.astype('U'))

#Tfidf Vectorizer for Question-2
q2_tfidf = TfidfVectorizer()

train_q2 = q2_tfidf.fit_transform(X_train['question2'].values.astype('U'))
test_q2 = q2_tfidf.transform(X_test['question2'].values.astype('U'))
```

In [10]:
```
##PRINTING THE SIZES OF THE QUESTION_1 TF-IDF SET
print("The shape of the Question_1 train set is =",train_q1.shape)
print("The shape of the Question_1 test set is =",test_q1.shape)

print("*"*100)

##PRINTING THE SIZES OF THE QUESTION_2 TF-IDF SET
print("The shape of the Question_2 train set is =",train_q2.shape)
print("The shape of the Question_2 train set is =",test_q2.shape)
```

```
The shape of the Question_1 train set is = (283003, 58300)
The shape of the Question_1 test set is = (121287, 58300)
****************************************************************************************************
The shape of the Question_2 train set is = (283003, 53850)
The shape of the Question_2 train set is = (121287, 53850)
```

Preparing the data for further modelling.

In [11]:
```
#Combining Question-1 and Question-2 of train and test sets

#Since tfidf vectorization returns sparse matrix as output so when I tried with np.hstack the dimensio
n was only=2
#train_tfidf = np.hstack((train_q1,train_q2))
#test_tfidf = np.hstack((test_q1,test_q2))
#train_tfidf.shape =(2)

#So I used the scipy.sparse.hstack representation which can do horizontal stacking in a sparse matrix
 efficiently
#https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.hstack.html
#https://stackoverflow.com/questions/37716699/how-to-hstack-several-sparse-matrices-feature-matrices

train_tfidf = hstack((train_q1,train_q2))
test_tfidf = hstack((test_q1,test_q2))

#Dropping question-1 and question-2 and Replacing with tfidf values
X_train.drop(['question1','question2'], axis=1, inplace=True)
X_test.drop(['question1','question2'], axis=1, inplace=True)
```

In [12]:
```
#Combining all train nlp features, adv features and tfidf features
X_Train = hstack((X_train, train_tfidf)).tocsr()

#Combining all test nlp features, adv features and tfidf features
X_Test = hstack((X_test, test_tfidf)).tocsr()
```

In [16]:
```
print("The Shape of the train set is = ",X_Train.shape)
print("*"*100)
print("The shape of the test set is = ", X_Test.shape)
```

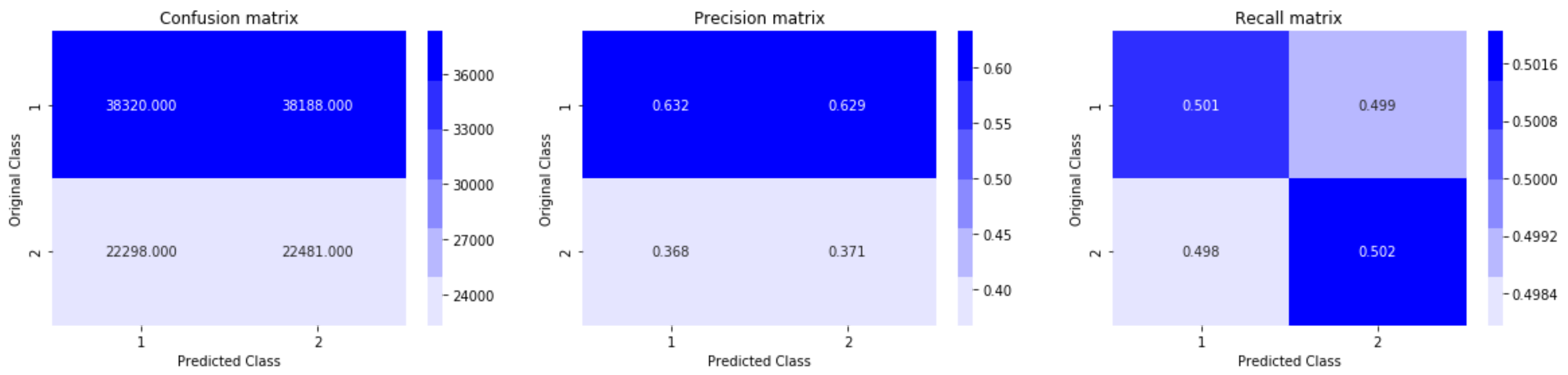
```
The Shape of the train set is =  (283003, 111824)
****************************************************************************************************
The shape of the test set is =  (121287, 111824)
```

Applying the Machine-Learning models over the Tf-idf Vectorized data

Implementing a simple Random-model

In [17]:
```
# we need to generate 9 numbers and the sum of numbers should be 1
# one solution is to genarate 9 numbers and divide each of the numbers by their sum
# ref: https://stackoverflow.com/a/18662466/4084039
# we create a output array that has exactly same size as the CV data
predicted_y = np.zeros((Test_len,2))
for i in range(Test_len):
    rand_probs = np.random.rand(1,2)
    predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Test Data using Random Model",log_loss(y_Test, predicted_y, eps=1e-15))

predicted_y =np.argmax(predicted_y, axis=1)
plot_confusion_matrix(y_Test, predicted_y)
```

```
Log loss on Test Data using Random Model 0.8818064860173283
```



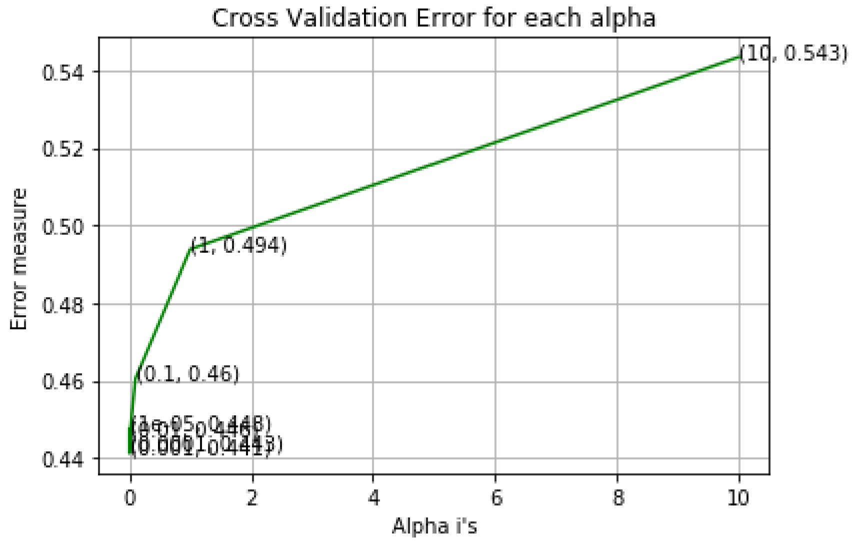
A random-model is implemnted to set an upper-limit for the log-loss values which can be used for the future modelling purpose.

Implementing and tuning the Logistic-Regression model

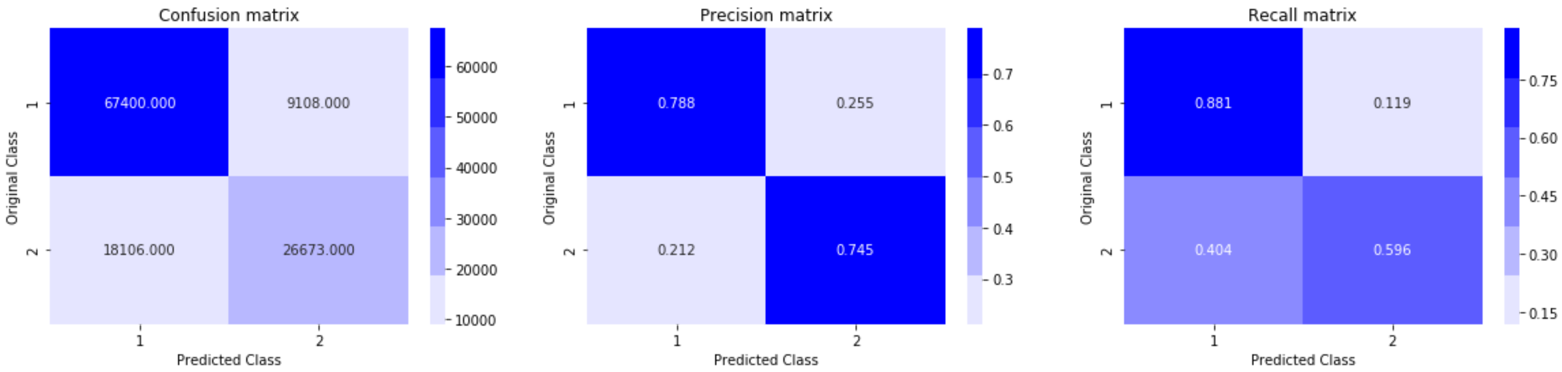
In [19]:

```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_m
odel.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=
None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.
0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, …])     Fit linear model with Stochastic Gradient Descent.
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link:
#------------------------------


log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(X_Train, y_Train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_Train, y_Train)
    predict_y = sig_clf.predict_proba(X_Test)
    log_error_array.append(log_loss(y_Test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_Test, predict_y, labels=clf.class
es_, eps=1e-15))

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)
clf.fit(X_Train, y_Train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_Train, y_Train)

predict_y = sig_clf.predict_proba(X_Train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_Train, pre
dict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_Test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_Test, predi
ct_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_Test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.44755124995361706
For values of alpha =  0.0001 The log loss is: 0.4425689125262311
For values of alpha =  0.001 The log loss is: 0.441261560321603
For values of alpha =  0.01 The log loss is: 0.4462572481911533
For values of alpha =  0.1 The log loss is: 0.46047094017633844
For values of alpha =  1 The log loss is: 0.49379407540868414
For values of alpha =  10 The log loss is: 0.5433843562419738
```




```
For values of best alpha =  0.001 The train log loss is: 0.4427183853539427
For values of best alpha =  0.001 The test log loss is: 0.441261560321603
Total number of data points : 121287
```

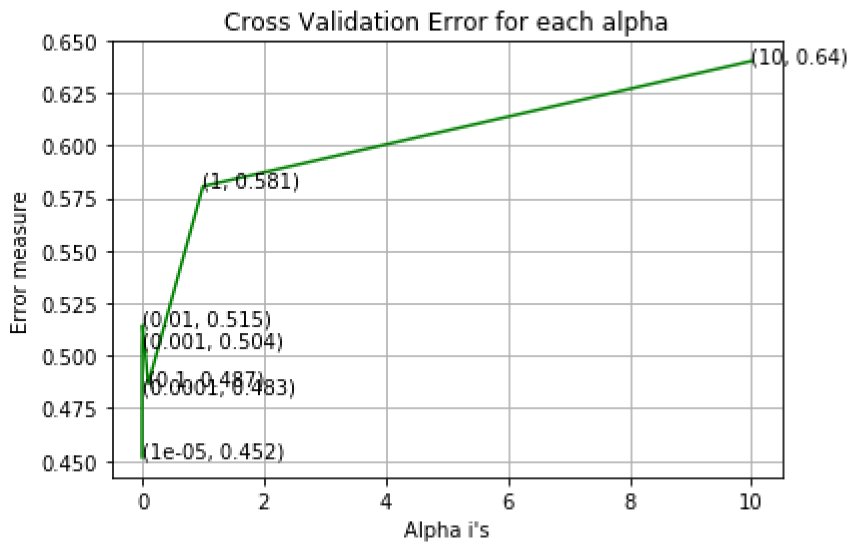


Implementing and Tuning the Linear-SVM model

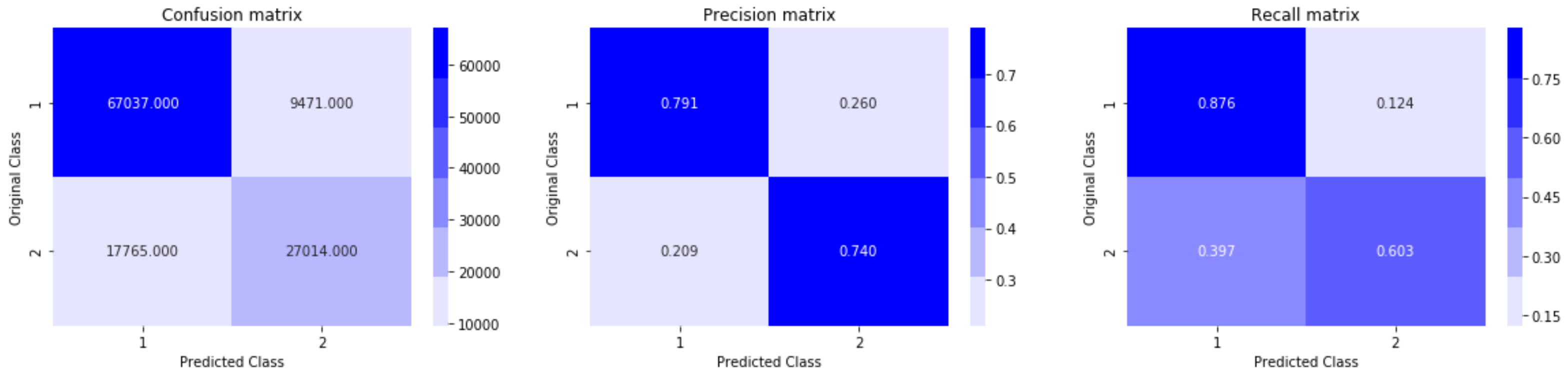
In [20]:

```python
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_m
odel.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=
None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.
0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, …])     Fit linear model with Stochastic Gradient Descent.
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link:
#------------------------------


log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l1', loss='hinge', random_state=42)
    clf.fit(X_Train, y_Train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_Train, y_Train)
    predict_y = sig_clf.predict_proba(X_Test)
    log_error_array.append(log_loss(y_Test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_Test, predict_y, labels=clf.class
es_, eps=1e-15))

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l1', loss='hinge', random_state=42)
clf.fit(X_Train, y_Train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_Train, y_Train)

predict_y = sig_clf.predict_proba(X_Train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_Train, pre
dict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_Test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_Test, predi
ct_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_Test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.4517101556926631
For values of alpha =  0.0001 The log loss is: 0.4828177138809125
For values of alpha =  0.001 The log loss is: 0.5040485523962721
For values of alpha =  0.01 The log loss is: 0.5145760955310129
For values of alpha =  0.1 The log loss is: 0.48651678732464176
For values of alpha =  1 The log loss is: 0.5806566434493734
For values of alpha =  10 The log loss is: 0.6402324511307722
```




```
For values of best alpha =  1e-05 The train log loss is: 0.45452816674610014
For values of best alpha =  1e-05 The test log loss is: 0.4517101556926631
Total number of data points : 121287
```



Observations

- The Test log-loss of both the linear models are (0.4412 and 0.4517) which is good than the Glove-vectorized models.
- The Linear models work pretty well on high-Dimensional text data so there is a quite decent amount of improvement is seen over the log-loss.
- But still the model is facing the bias problem because the recall score of the minority class is lower but still it is better than the Glove-vectorized data.
- The bias problem can be solved by using the XB-Boost classiffier which I tried next.

Implementing the XG-Boost classiffier over theTF-idf Vectorized data

In [29]:
```
import xgboost as xgb
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 4

D_train = xgb.DMatrix(X_Train, label=y_Train)
D_test = xgb.DMatrix(X_Test, label=y_Test)

Watchlist = [(D_train, 'train'), (D_test, 'valid')]

Bst = xgb.train(params, d_train, 400, Watchlist, early_stopping_rounds=20, verbose_eval=2)

xgdmat = xgb.DMatrix(X_Train,y_Train)
Predict_y = bst.predict(D_test)
print("The test log loss is:",log_loss(y_Test, Predict_y, labels=clf.classes_, eps=1e-15))
```

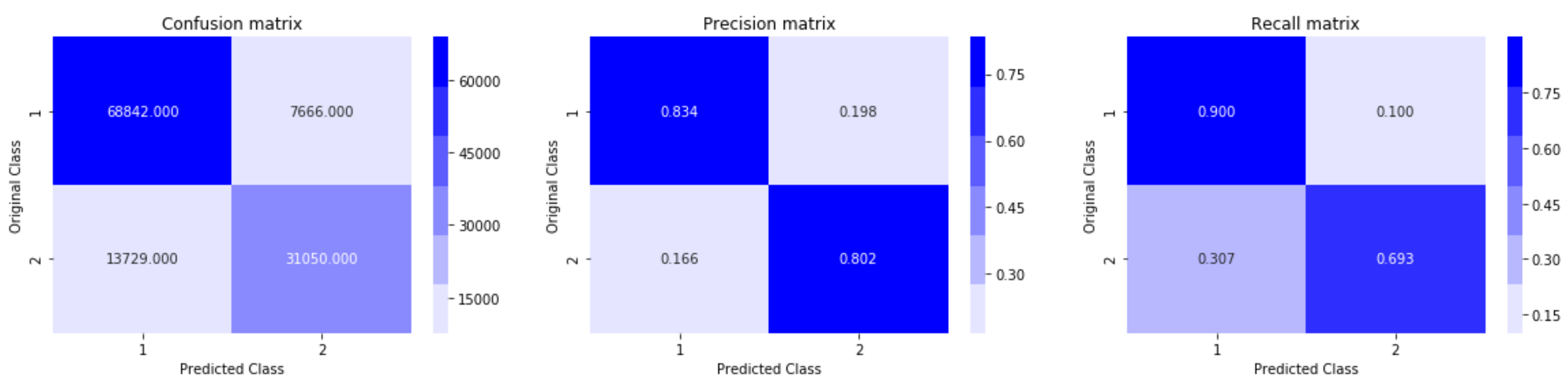
```
[19:01:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[0]	train-logloss:0.68487	valid-logloss:0.684852
Multiple eval metrics have been passed: 'valid-logloss' will be used for early stopping.

Will train until valid-logloss hasn't improved in 20 rounds.
[19:01:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[2]	train-logloss:0.668854	valid-logloss:0.668812
[19:01:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[4]	train-logloss:0.654026	valid-logloss:0.653972
[19:01:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[6]	train-logloss:0.640406	valid-logloss:0.640348
[19:01:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:14] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[8]	train-logloss:0.627344	valid-logloss:0.627279
[19:01:14] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:14] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[10]	train-logloss:0.615453	valid-logloss:0.615362
[19:01:14] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:15] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[12]	train-logloss:0.603715	valid-logloss:0.603655
[19:01:15] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:15] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[14]	train-logloss:0.593085	valid-logloss:0.592997
[19:01:15] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:16] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[16]	train-logloss:0.58304	valid-logloss:0.582981
[19:01:16] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:16] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[18]	train-logloss:0.573564	valid-logloss:0.573492
[19:01:16] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:17] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[20]	train-logloss:0.564568	valid-logloss:0.564505
[19:01:17] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:17] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[22]	train-logloss:0.555852	valid-logloss:0.555815
[19:01:18] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:18] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[24]	train-logloss:0.547939	valid-logloss:0.547903
[19:01:18] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:18] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[26]	train-logloss:0.540119	valid-logloss:0.540091
[19:01:19] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:19] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[28]	train-logloss:0.533168	valid-logloss:0.533157
[19:01:19] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:19] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[30]	train-logloss:0.526415	valid-logloss:0.526425
[19:01:20] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:01:20] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[32]	train-logloss:0.520141	valid-logloss:0.52014
[19:01:20] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
```

```
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:02:54] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 22 extra nodes, 0 pruned nodes, max_depth=4
[382]	train-logloss:0.354976	valid-logloss:0.357216
[19:02:55] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 20 extra nodes, 0 pruned nodes, max_depth=4
[19:02:55] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 22 extra nodes, 0 pruned nodes, max_depth=4
[384]	train-logloss:0.354896	valid-logloss:0.35714
[19:02:55] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:02:55] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 18 extra nodes, 0 pruned nodes, max_depth=4
[386]	train-logloss:0.354786	valid-logloss:0.357042
[19:02:56] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 26 extra nodes, 0 pruned nodes, max_depth=4
[19:02:56] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[388]	train-logloss:0.35465	valid-logloss:0.356918
[19:02:56] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:02:56] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 26 extra nodes, 0 pruned nodes, max_depth=4
[390]	train-logloss:0.354533	valid-logloss:0.356807
[19:02:57] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 20 extra nodes, 0 pruned nodes, max_depth=4
[19:02:57] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 0 pruned nodes, max_depth=4
[392]	train-logloss:0.35442	valid-logloss:0.356712
[19:02:57] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 26 extra nodes, 0 pruned nodes, max_depth=4
[19:02:57] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 28 extra nodes, 0 pruned nodes, max_depth=4
[394]	train-logloss:0.354275	valid-logloss:0.356576
[19:02:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 24 extra nodes, 0 pruned nodes, max_depth=4
[19:02:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[396]	train-logloss:0.354171	valid-logloss:0.356484
[19:02:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[19:02:58] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 30 extra nodes, 0 pruned nodes, max_depth=4
[398]	train-logloss:0.354051	valid-logloss:0.356373
[19:02:59] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 18 extra nodes, 0 pruned nodes, max_depth=4
[399]	train-logloss:0.353987	valid-logloss:0.35631
The test log loss is: 0.35630992698446756
```

In [30]:

```
Predicted_y =np.array(Predict_y>0.5,dtype=int)
print("Total number of data points :", len(Predicted_y))
plot_confusion_matrix(y_Test, Predicted_y)
```

```
Total number of data points : 121287
```



- The Test Log-loss of the XG-Boost classiffier is 0.3563 which is good than the linear model.
- The log-loss of this XG-boost classiffier is slightly higher than the Glove-Vectorized XG-Boost classiffier.
- This is Because of the increase in the dimensionality of the data and Xg-Boost tends to not perform well over high-dimensional data.
- The performance of the model can be increased by tuning the hyper-parameters properly.

Tuning the Hyper-parameters by using the Random-search technique

In [25]:
```
%%time
tuned_prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,1000,1500],
}

Model=xgb.XGBClassifier(n_jobs=-1, random_state=15)

Randomsearch_tuning(Model,tuned_prams,X_Train,y_Train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed: 43.0min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 89.7min finished
```

```
Best: -0.311020 using {'n_estimators': 1500, 'learning_rate': 0.15}
-0.315477 (0.000843) with: {'n_estimators': 1000, 'learning_rate': 0.15}
-0.338641 (0.000963) with: {'n_estimators': 200, 'learning_rate': 0.15}
-0.347058 (0.000673) with: {'n_estimators': 200, 'learning_rate': 0.1}
-0.334143 (0.000886) with: {'n_estimators': 1500, 'learning_rate': 0.03}
-0.324072 (0.000990) with: {'n_estimators': 500, 'learning_rate': 0.15}
-0.375315 (0.000680) with: {'n_estimators': 200, 'learning_rate': 0.03}
-0.311020 (0.000862) with: {'n_estimators': 1500, 'learning_rate': 0.15}
-0.362333 (0.000691) with: {'n_estimators': 100, 'learning_rate': 0.1}
-0.355176 (0.000753) with: {'n_estimators': 1500, 'learning_rate': 0.01}
-0.363612 (0.000742) with: {'n_estimators': 1000, 'learning_rate': 0.01}
Wall time: 1h 34min 1s
```

Trying the model with the above tuned parameter values

In [32]:
```
parameter1 = {}
parameter1['objective'] = 'binary:logistic'
parameter1['eval_metric'] = 'logloss'
parameter1['eta'] = 0.02
parameter1['max_depth'] = 3
parameter1["n_estimators"]=1500
parameter1["learning_rate"]=0.15
parameter1["n_jobs"]=-1

Bst1 = xgb.train(parameter1, d_train, 400, watchlist, early_stopping_rounds=20, verbose_eval=2)

xgdmat_1 = xgb.DMatrix(X_Train,y_Train)
Predict_y1 = Bst1.predict(D_test)
print("The test log loss is:",log_loss(y_Test, Predict_y1, labels=clf.classes_, eps=1e-15))
```

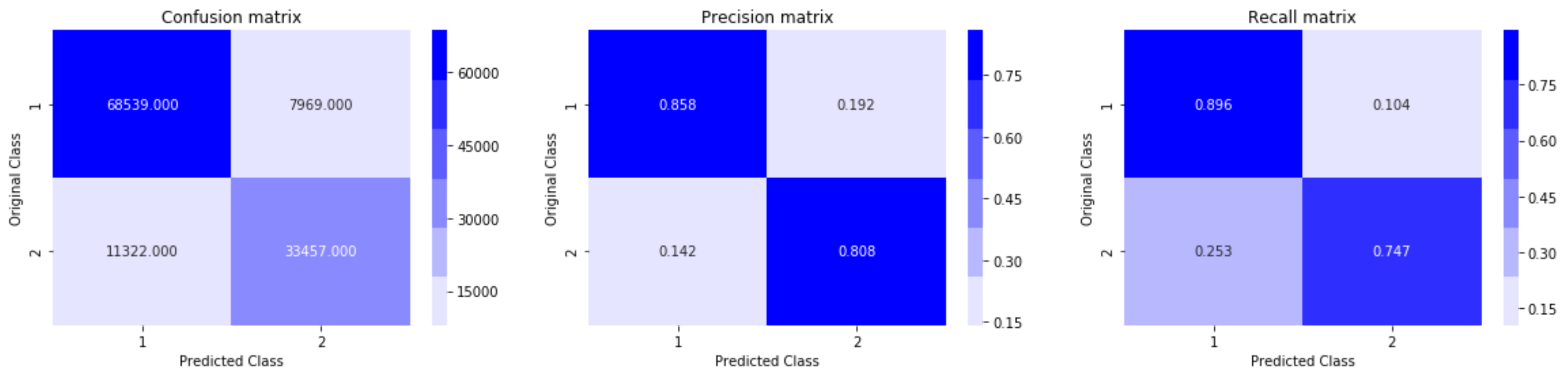
```
[19:09:06] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[0]     train-logloss:0.640966  valid-logloss:0.640763
Multiple eval metrics have been passed: 'valid-logloss' will be used for early stopping.

Will train until valid-logloss hasn't improved in 20 rounds.
[19:09:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[2]     train-logloss:0.564743  valid-logloss:0.564751
[19:09:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:07] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[4]     train-logloss:0.516939  valid-logloss:0.516919
[19:09:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[6]     train-logloss:0.483056  valid-logloss:0.483066
[19:09:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:08] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[8]     train-logloss:0.460127  valid-logloss:0.460177
[19:09:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[10]    train-logloss:0.443093  valid-logloss:0.443213
[19:09:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[12]    train-logloss:0.430094  valid-logloss:0.43032
[19:09:09] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:10] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[14]    train-logloss:0.419773  valid-logloss:0.420219
[19:09:10] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:10] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[16]    train-logloss:0.411929  valid-logloss:0.412422
[19:09:10] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:10] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[18]    train-logloss:0.40615   valid-logloss:0.40666
[19:09:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[20]    train-logloss:0.400844  valid-logloss:0.401379
[19:09:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[22]    train-logloss:0.39588   valid-logloss:0.39651
[19:09:11] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[24]    train-logloss:0.392798  valid-logloss:0.39353
[19:09:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[26]    train-logloss:0.390091  valid-logloss:0.390864
[19:09:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[28]    train-logloss:0.386813  valid-logloss:0.387604
[19:09:12] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[30]    train-logloss:0.384853  valid-logloss:0.385705
[19:09:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:09:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[32]    train-logloss:0.382029  valid-logloss:0.382979
[19:09:13] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
```

```
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:10:21] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[382]	train-logloss:0.319404	valid-logloss:0.327645
[19:10:21] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:10:22] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[384]	train-logloss:0.319316	valid-logloss:0.32761
[19:10:22] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:10:22] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[386]	train-logloss:0.319229	valid-logloss:0.327564
[19:10:22] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[19:10:22] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[388]	train-logloss:0.319151	valid-logloss:0.327521
[19:10:22] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:10:23] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[390]	train-logloss:0.319044	valid-logloss:0.327449
[19:10:23] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:10:23] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[392]	train-logloss:0.318944	valid-logloss:0.327418
[19:10:23] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[19:10:23] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[394]	train-logloss:0.318859	valid-logloss:0.327364
[19:10:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:10:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[396]	train-logloss:0.318771	valid-logloss:0.327323
[19:10:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[19:10:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[398]	train-logloss:0.318689	valid-logloss:0.327281
[19:10:24] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[399]	train-logloss:0.318597	valid-logloss:0.327221
The test log loss is: 0.3272213436181299
```

In [33]:

```
Predicted_Y =np.array(Predict_y1>0.5,dtype=int)
print("Total number of data points :", len(Predicted_Y))
plot_confusion_matrix(y_Test, Predicted_Y)
```

```
Total number of data points : 121287
```



Tuning the different set of hyper-parameter values

In [27]:

```
%%time
TUNED_prams2={
 'colsample_bytree':[0.1,0.3,0.5,1],
 'subsample':[0.1,0.3,0.5,1],
 'min_child_weight':[2,4,6,8],
 'gamma':[i/10.0 for i in range(0,5)],
 'reg_alpha':[1e-5, 1e-2, 0.1, 1, 100],
}

Model_2=xgb.XGBClassifier(n_jobs=-1, random_state=15,subsample= 1,n_estimators= 1500,learning_rate= 0.
15)

Randomsearch_tuning(Model_2,TUNED_prams2,X_Train,y_Train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed: 58.2min
[Parallel(n_jobs=-1)]: Done  50 out of  50 | elapsed: 95.0min finished
```

```
Best: -0.310347 using {'subsample': 1, 'reg_alpha': 1e-05, 'min_child_weight': 8, 'gamma': 0.0, 'colsa
mple_bytree': 1}
-0.311300 (0.000719) with: {'subsample': 1, 'reg_alpha': 1e-05, 'min_child_weight': 2, 'gamma': 0.0,
'colsample_bytree': 1}
-0.312276 (0.001107) with: {'subsample': 0.5, 'reg_alpha': 0.1, 'min_child_weight': 8, 'gamma': 0.4,
'colsample_bytree': 0.3}
-0.332015 (0.001213) with: {'subsample': 1, 'reg_alpha': 100, 'min_child_weight': 4, 'gamma': 0.0, 'co
lsample_bytree': 1}
-0.310347 (0.001190) with: {'subsample': 1, 'reg_alpha': 1e-05, 'min_child_weight': 8, 'gamma': 0.0,
'colsample_bytree': 1}
-0.327857 (0.001952) with: {'subsample': 0.1, 'reg_alpha': 1e-05, 'min_child_weight': 8, 'gamma': 0.0,
'colsample_bytree': 0.1}
-0.323843 (0.001518) with: {'subsample': 0.1, 'reg_alpha': 1e-05, 'min_child_weight': 8, 'gamma': 0.0,
'colsample_bytree': 0.5}
-0.312033 (0.000722) with: {'subsample': 1, 'reg_alpha': 1, 'min_child_weight': 2, 'gamma': 0.3, 'cols
ample_bytree': 0.3}
-0.333788 (0.001363) with: {'subsample': 1, 'reg_alpha': 100, 'min_child_weight': 2, 'gamma': 0.4, 'co
lsample_bytree': 0.1}
-0.310843 (0.001014) with: {'subsample': 0.3, 'reg_alpha': 0.1, 'min_child_weight': 4, 'gamma': 0.0,
'colsample_bytree': 1}
-0.329164 (0.001375) with: {'subsample': 0.5, 'reg_alpha': 100, 'min_child_weight': 2, 'gamma': 0.2,
'colsample_bytree': 1}
Wall time: 1h 39min 4s
```

Implementing the final XGB-Classiffier model with tuned Hyperparameter values

In [34]:

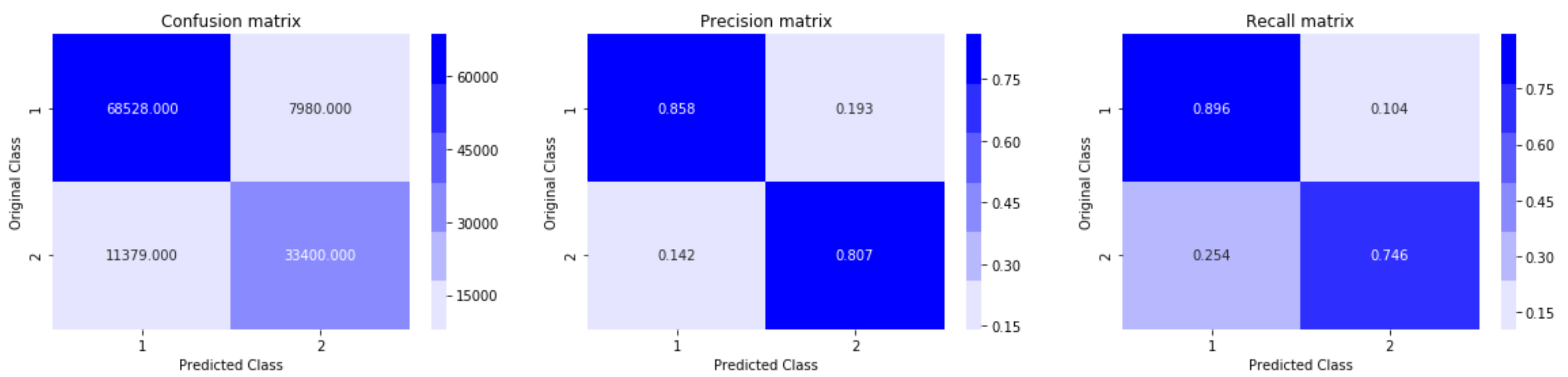
```
parameter2 = {}
parameter2['objective'] = 'binary:logistic'
parameter2['eval_metric'] = 'logloss'
parameter2['eta'] = 0.02
parameter2['max_depth'] = 3
parameter2["subsample"]=1
parameter2["n_estimators"]=1500
parameter2["learning_rate"]=0.15
parameter2["colsample_bytree"]=1
parameter2["n_jobs"]=-1
parameter2['reg_alpha'] = 1e-05
parameter2['min_child_weight'] = 8
parameter2['gamma'] =  0.0

Bst2 = xgb.train(parameter2, D_train, 400, Watchlist, early_stopping_rounds=20, verbose_eval=2)

xgdmat2 = xgb.DMatrix(X_Train,y_Train)
Predict_y2 = Bst2.predict(D_test)
print("The test log loss is:",log_loss(y_Test, Predict_y2, labels=clf.classes_, eps=1e-15))
```

```
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:18:35] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[382]	train-logloss:0.320762	valid-logloss:0.328432
[19:18:35] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 8 extra nodes, 0 pruned nodes, max_depth=3
[19:18:35] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 8 extra nodes, 0 pruned nodes, max_depth=3
[384]	train-logloss:0.320653	valid-logloss:0.328373
[19:18:36] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 10 extra nodes, 0 pruned nodes, max_depth=3
[19:18:36] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[386]	train-logloss:0.320556	valid-logloss:0.328305
[19:18:36] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[19:18:36] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 8 extra nodes, 0 pruned nodes, max_depth=3
[388]	train-logloss:0.320394	valid-logloss:0.328201
[19:18:36] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[19:18:36] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 10 extra nodes, 0 pruned nodes, max_depth=3
[390]	train-logloss:0.320253	valid-logloss:0.328077
[19:18:37] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[19:18:37] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 14 extra nodes, 0 pruned nodes, max_depth=3
[392]	train-logloss:0.320047	valid-logloss:0.327924
[19:18:37] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 8 extra nodes, 0 pruned nodes, max_depth=3
[19:18:37] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 10 extra nodes, 0 pruned nodes, max_depth=3
[394]	train-logloss:0.319954	valid-logloss:0.32786
[19:18:37] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 8 extra nodes, 0 pruned nodes, max_depth=3
[19:18:38] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 8 extra nodes, 0 pruned nodes, max_depth=3
[396]	train-logloss:0.319883	valid-logloss:0.327818
[19:18:38] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[19:18:38] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 12 extra nodes, 0 pruned nodes, max_depth=3
[398]	train-logloss:0.319769	valid-logloss:0.327702
[19:18:38] C:\Users\Administrator\Desktop\xgboost\src\tree\updater_prune.cc:74: tree pruning end, 1 ro
ots, 8 extra nodes, 0 pruned nodes, max_depth=3
[399]	train-logloss:0.31972	valid-logloss:0.327681
The test log loss is: 0.3276809720369367
```

In [44]:

```
Predicted_Y2 =np.array(Predict_y2>0.5,dtype=int)
print("Total number of data points :", len(Predicted_Y2))
plot_confusion_matrix(y_Test, Predicted_Y2)
```

```
Total number of data points : 121287
```



Observations

- The Test log-loss of the previous tuned parameter set was better than the final parameter set so previous model is considered.
- The Test log-loss with optimal hyper-parameters is 0.3272 which is better than all the previous models including the Glove Vectorized models.

Function for printing the final conclusion-table

In [45]:
```
def conclusion_table():
    print()
    ptable=PrettyTable()
    ptable.title="The comparisons of all the algorithms and Vectorizers are as follows: "
    ptable.field_names=["Vectorizer_1","Algorithm_1","Log_loss_1","Vectorizer_2","Algorithm_2","Log_lo
ss_2","Difference"]
    ptable.add_row(["Glove","Random_1",0.8859,"Tf-idf","Random_2",0.8818,0.0041])
    ptable.add_row(["Glove","Logistic_1",0.5094 ,"Tf-idf","Logistic_2",0.4412,0.0682])
    ptable.add_row(["Glove","Linear_SVM_1",0.4837 ,"Tf-idf","Linear_SVM_2",0.4517,0.0320])
    ptable.add_row(["Glove","XG-Boost_1",0.3533 ,"Tf-idf","XG-Boost_2",0.3563,0.0030])
    ptable.add_row(["Glove","Tuned_XG-Boost_1",0.3320 ,"Tf-idf","Tuned_XG-Boost_2",0.3272,0.0048])
    print(ptable)
```

Flow & Conclusion of the case-study

The whole above case-study can be easily broken down into final 4 parts and they are as follows:

1. EDA
2. Feature-Engineering.
3. Data-Featurization.
    - GLOVE.
    - TF-IDF Vectorization.
4. Modelling.

There are total of 404290 number of datapoints & 6 columns present in the above dataset at start.The distribution of the class-labels was not very skewed in nature and it is a binary classiffication problem with log-loss as evalution metric.

The initial features was not sufficient and informative so I had tried 2-types of feature engineering techniques in this case study and they are as follows:

1. Basic Features (11-features).
2. Advanced-NLP & Fuzzy features(15-features).

This phase of the case-study is very important as it provided very fruitful and important insights of the data and helped in tackling the problem.

T-sne is used for visualizing the data in both 2d and 3d formats which gave a very intutive understanding of the data.

The text featurization techniques used here is GLOVE and TFIDF Vectorization techniques.The dimensions of the data afer GLOVE vectorization was 797 and the dimensions of the data after TF-IDF Vectorization is around 111824.

So clearly the glove vectorization has less dimensions and the data is dense in nature whereas the TF-idf Vectorization technique creates a high dimensional data with sparse in nature.

After all this steps finally the data is taken for modelling and following supervised machine learning models are apllied:

- Logistic-Regression.
- Linear_SVM.
- XG-BOOST.

All the above models are tuned by Hyper-parameter tuning for finding the best values of the hyper-parameters and finding the best log-loss value.The performance of the model can be seen on the below table as follows:-

In [46]:
```
conclusion_table()
```

The comparisons of all the algorithms and Vectorizers are as follows:

Vectorizer_1	Algorithm_1	Log_loss_1	Vectorizer_2	Algorithm_2	Log_loss_2	Difference
Glove	Random_1	0.8859	Tf-idf	Random_2	0.8818	0.0041
Glove	Logistic_1	0.5094	Tf-idf	Logistic_2	0.4412	0.0682
Glove	Linear_SVM_1	0.4837	Tf-idf	Linear_SVM_2	0.4517	0.032
Glove	XG-Boost_1	0.3533	Tf-idf	XG-Boost_2	0.3563	0.003
Glove	Tuned_XG-Boost_1	0.332	Tf-idf	Tuned_XG-Boost_2	0.3272	0.0048

1. After doing all the analysis I can conclude that the Tuned_XG-Boost model is better than the simple Linear model like {Logistic and Linear-SVM} for solving the Qoura-Question-pair similarity problem.
2. The Linear models tend to perform better over High-dimensional data as compared to low dimensional data
3. The Tfidf vectorization strategy is better than the Glove strategy as it produced better results which can be observed on the above table.

```
In [ ]:
```