In [139]:
```
import pandas as pd
import numpy as np
import re
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import TfidfVectorizer
from tqdm import tqdm
import matplotlib.pyplot as plt
import os
from scipy.sparse import hstack
import warnings
import seaborn as sns
```

In [102]:
```

dfnlp = pd.read_csv('NLP_features_train.csv',encoding='latin-1')
dfppro = pd.read_csv('df_fe_without_preprocessing_train.csv',encoding='latin-1')
```

In [103]:
```
df1 = dfnlp.drop(['qid1','qid2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
```

In [104]:
```
df1.head()
```

Out[104]:

	id	question1	question2	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_
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	0.916667	0.78
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	0.700000	0.46
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	0.400000	0.28
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	0.00
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	0.571429	0.30

In [105]:
```
df2.head()
```

Out[105]:

	id	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_common	word_Total	W
0	0	1	1	66	57	14	12	10.0	23.0	
1	1	4	1	51	88	8	13	4.0	20.0	
2	2	1	1	73	59	14	10	4.0	24.0	
3	3	1	1	50	65	11	9	0.0	19.0	
4	4	3	1	76	39	13	7	2.0	20.0	

In [106]:
```
df1 = df1.merge(df2,on='id',how='left')
```

In [107]:
```
df1.head(2)
```

Out[107]:

	id	question1	question2	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_
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	0.916667	0.78
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	0.700000	0.46

2 rows × 30 columns

In [108]:
```
df1 = df1[df1['question1'].notnull()]
df1 = df1[df1['question2'].notnull()]
```

In [110]:
```
Y = df1['is_duplicate']
Y.shape
```

Out[110]:
```
(404269,)
```

In [112]:
```
df1 = df1.drop(['id','is_duplicate'],axis=1)
```

In [113]:
```
tfidfq1 = TfidfVectorizer()
q1_tfidf = tfidfq1.fit_transform(df1['question1'].values.astype('U'))
```

In [114]:
```
type(q1_tfidf)
```

Out[114]:
```
scipy.sparse.csr.csr_matrix
```

In [115]:
```
q1_tfidf.shape
```

Out[115]:
```
(404269, 67909)
```

In [116]:
```
tfidfq2 = TfidfVectorizer()
q2_tfidf = tfidfq2.fit_transform(df1['question2'].values.astype('U'))
```

In [117]:
```
q2_tfidf.shape
```

Out[117]:
```
(404269, 62704)
```

In [118]:
```
## Combining the two questions together
q_tfidf = hstack((q1_tfidf,q2_tfidf))
```

In [119]:
```
df1 = df1.drop(['question1','question2'],axis=1,inplace=True)
```

In [120]:
```
q_tfidf.shape
```

Out[120]:
```
(404269, 130613)
```

In [121]:
```
type(q_tfidf)
```

Out[121]:
```
scipy.sparse.coo.coo_matrix
```

In [155]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier



from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
##from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
```

In [156]:
```
X_train,X_test,Y_train,Y_test = train_test_split(q_tfidf,Y,stratify=Y,test_size=0
```

In [124]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(282988, 130613)
(121281, 130613)
```

In [125]:
```
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 

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

    # 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 row
    # 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 
    # C = [[1, 2],
    #     [3, 4]]
    # C.sum(axis = 0)  axis=0 corresonds to columns and axis=1 corresponds to row
    # 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, yticklab
    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, yticklab
    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, yticklab
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")

    plt.show()
```

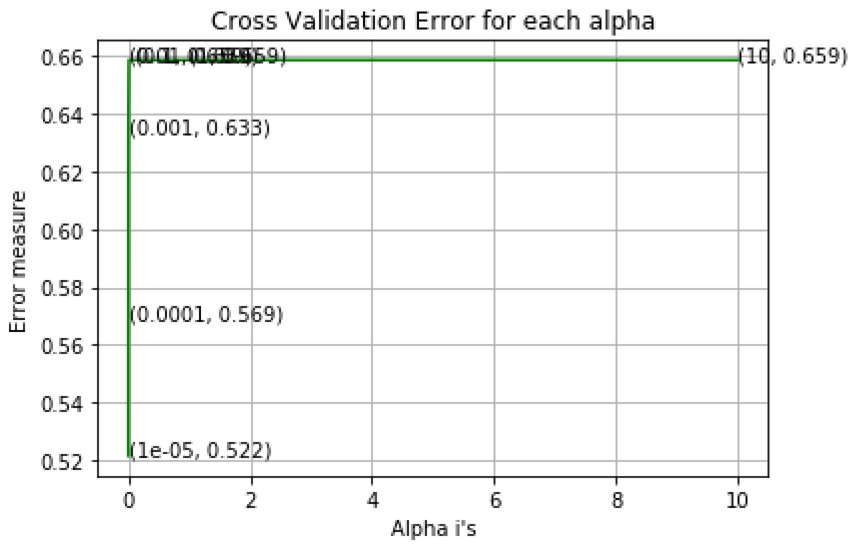
Logistic Regression(Hyper-parameter tuned)

In [131]:
```
alpha = [10**x for x in range(-5,2)]

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-1
    print('For the values of alpha = ',i,"The log loss is:",log_loss(Y_test,predi
```

```
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:128: FutureWarning: max_iter and tol parameters have been added
in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19.
If both are left unset, they default to max_iter=5 and tol=None. If tol is no
t None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:128: FutureWarning: max_iter and tol parameters have been added
in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19.
If both are left unset, they default to max_iter=5 and tol=None. If tol is no
t None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:128: FutureWarning: max_iter and tol parameters have been added
in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19.
If both are left unset, they default to max_iter=5 and tol=None. If tol is no
t None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
```

In [133]:
```
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()
```



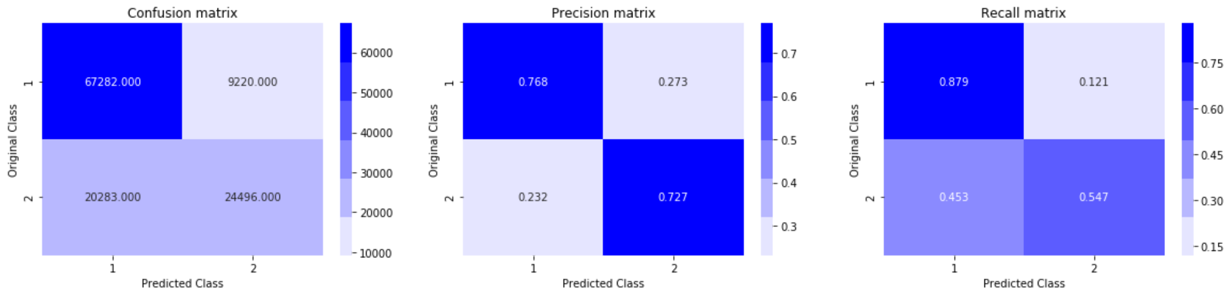
In [135]:
```
best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha],penalty='l2',loss='log',random_state=
clf.fit(X_train,Y_train)
sig_clf = CalibratedClassifierCV(clf,method="sigmoid")
sig_clf.fit(X_train,Y_train)
```

```
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

Out[135]:
```
CalibratedClassifierCV(base_estimator=SGDClassifier(alpha=1e-05, average=False,
class_weight=None, epsilon=0.1,
       eta0=0.0, fit_intercept=True, l1_ratio=0.15,
       learning_rate='optimal', loss='log', max_iter=None, n_iter=None,
       n_jobs=1, penalty='l2', power_t=0.5, random_state=42, shuffle=True,
       tol=None, verbose=0, warm_start=False),
            cv=3, method='sigmoid')
```

In [140]:
```
predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",l
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 best alpha =  1e-05 The train log loss is: 0.4720810857646023
For values of best alpha =  1e-05 The test log loss is: 0.5082508140455909
Total number of data points : 121281
```

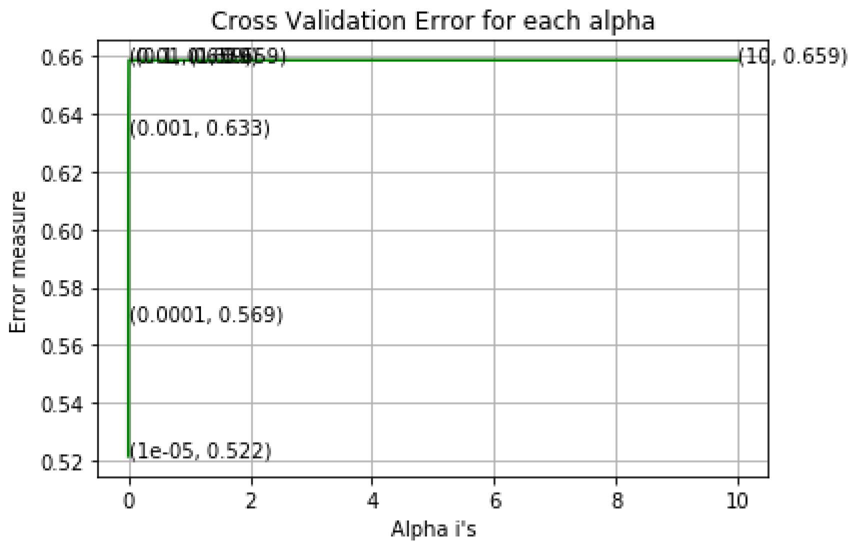


Linear SVM with hyperparameter tuning

In [142]:
```
alpha = [10 ** x for x in range(-5, 2)]
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=1
    print('For values of alpha = ', i, "The log loss is:",log_loss(Y_test, predic
```

```
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:128: FutureWarning: max_iter and tol parameters have been added
in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19.
If both are left unset, they default to max_iter=5 and tol=None. If tol is no
t None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:128: FutureWarning: max_iter and tol parameters have been added
in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19.
If both are left unset, they default to max_iter=5 and tol=None. If tol is no
t None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:128: FutureWarning: max_iter and tol parameters have been added
in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19.
If both are left unset, they default to max_iter=5 and tol=None. If tol is no
t None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
```

In [143]:
```
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()
```



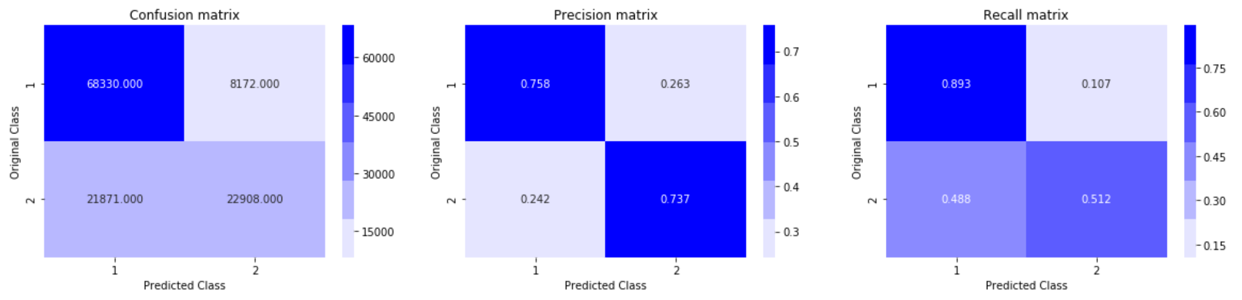
In [145]:
```
best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l1', loss='hinge', random_s
clf.fit(X_train, Y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, Y_train)
```

```
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
D:\Anaconda\envs\tensorflow\lib\site-packages\sklearn\linear_model\stochastic_g
radient.py:128: FutureWarning: max_iter and tol parameters have been added in <
class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If bot
h are left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, a
nd default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

Out[145]:
```
CalibratedClassifierCV(base_estimator=SGDClassifier(alpha=1e-05, average=False,
class_weight=None, epsilon=0.1,
       eta0=0.0, fit_intercept=True, l1_ratio=0.15,
       learning_rate='optimal', loss='hinge', max_iter=None, n_iter=None,
       n_jobs=1, penalty='l1', power_t=0.5, random_state=42, shuffle=True,
       tol=None, verbose=0, warm_start=False),
            cv=3, method='sigmoid')
```

In [146]:
```
predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",l
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 best alpha =  1e-05 The train log loss is: 0.49949532720743306
For values of best alpha =  1e-05 The test log loss is: 0.521583251161899
Total number of data points : 121281
```



XGBOOST

In [159]:
```
import xgboost as xgb
from xgboost import XGBClassifier
params = {
    'max_depth':[3,5,6,7,8],
    'learning_rate' :[0.01,0.02,0.03,0.1,0.2,0.3],
    'n_estimators':[100,200,300,400,500],
    'gamma':[0,0.5,1,1.5,2,5]
    }
model = XGBClassifier(nthread=-1)
kfold = StratifiedKFold(n_splits=4,shuffle=True)
random_search  = RandomizedSearchCV(model,param_distributions=params,scoring="neg
random_result  = random_search.fit(X_train,Y_train)
```

In [160]:
```
print(random_search.best_estimator_)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=2, learning_rate=0.3, max_delta_step=0,
       max_depth=6, min_child_weight=1, missing=None, n_estimators=200,
       n_jobs=1, nthread=-1, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

In [161]:
```
print(random_search.best_params_)
```

```
{'n_estimators': 200, 'learning_rate': 0.3, 'max_depth': 6, 'gamma': 2}
```

In [162]:
```
import xgboost as xgb
params = {}
params['objective'] ='binary:logistic'
params['eval_metric']  = 'logloss'
params['n_estimators'] = 200
params['learning_rate'] = 0.3
params['max_depth'] = 6
params['gamma']  = 2

d_train = xgb.DMatrix(X_train,label=Y_train)
d_test  = xgb.DMatrix(X_test,label=Y_test)

watch_list = [(d_train,'train'),(d_test,'valid')]
bst = xgb.train(params,d_train,400,watch_list,early_stopping_rounds=20,verbose_ev
xgdmat = xgb.DMatrix(X_train,Y_train)
```

```
[0]     train-logloss:0.652804  valid-logloss:0.653483
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.
[10]    train-logloss:0.564021  valid-logloss:0.56849
[20]    train-logloss:0.540598  valid-logloss:0.547044
[30]    train-logloss:0.526073  valid-logloss:0.534579
[40]    train-logloss:0.515753  valid-logloss:0.525502
[50]    train-logloss:0.507798  valid-logloss:0.519003
[60]    train-logloss:0.500988  valid-logloss:0.513297
[70]    train-logloss:0.494719  valid-logloss:0.508467
[80]    train-logloss:0.490062  valid-logloss:0.505168
[90]    train-logloss:0.484666  valid-logloss:0.50088
[100]   train-logloss:0.47955   valid-logloss:0.497495
[110]   train-logloss:0.475843  valid-logloss:0.494796
[120]   train-logloss:0.47244   valid-logloss:0.492377
[130]   train-logloss:0.468747  valid-logloss:0.489726
[140]   train-logloss:0.465923  valid-logloss:0.487911
[150]   train-logloss:0.463325  valid-logloss:0.486273
[160]   train-logloss:0.460787  valid-logloss:0.484735
[170]   train-logloss:0.457638  valid-logloss:0.482706
[180]   train-logloss:0.455027  valid-logloss:0.481064
[190]   train-logloss:0.451385  valid-logloss:0.478542
[200]   train-logloss:0.449522  valid-logloss:0.477311
[210]   train-logloss:0.44727   valid-logloss:0.475891
[220]   train-logloss:0.445042  valid-logloss:0.474544
[230]   train-logloss:0.442694  valid-logloss:0.472868
[240]   train-logloss:0.440596  valid-logloss:0.471629
[250]   train-logloss:0.438466  valid-logloss:0.470357
[260]   train-logloss:0.434884  valid-logloss:0.467995
[270]   train-logloss:0.431886  valid-logloss:0.465711
[280]   train-logloss:0.430291  valid-logloss:0.464759
[290]   train-logloss:0.428458  valid-logloss:0.46379
[300]   train-logloss:0.427097  valid-logloss:0.463067
[310]   train-logloss:0.42553   valid-logloss:0.462268
[320]   train-logloss:0.423604  valid-logloss:0.461015
[330]   train-logloss:0.420737  valid-logloss:0.459352
[340]   train-logloss:0.419403  valid-logloss:0.458584
```

```
[350]	train-logloss:0.41737	valid-logloss:0.457503
[360]	train-logloss:0.415963	valid-logloss:0.456825
[370]	train-logloss:0.414507	valid-logloss:0.456152
[380]	train-logloss:0.413223	valid-logloss:0.455474
[390]	train-logloss:0.411768	valid-logloss:0.454846
[399]	train-logloss:0.410378	valid-logloss:0.453996
```

In [163]:

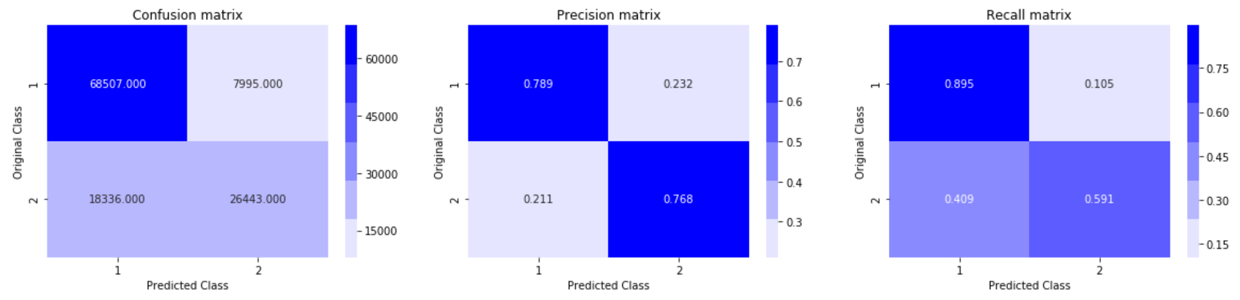
```
predict_y1 = bst.predict(d_test)
print("The test log Loss is:",log_loss(Y_test,predict_y1,labels=clf.classes_,eps=
```

```
The test log Loss is: 0.45399573660014325
```

In [164]:

```
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 : 121281
```



In [168]:

```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Model","Train_log_loss","Test_log_loss"]
x.add_row(["Logistic Regression",0.4720810857646023,0.5082508140455909])
x.add_row(["Linear SVM",0.4994953272074330,0.521583251161899])
x.add_row(["XGBoost",0.410378,0.45399573660014325])
print(x)
```

```
+---------------------+--------------------+---------------------+
|        Model        |   Train_log_loss   |    Test_log_loss    |
+---------------------+--------------------+---------------------+
| Logistic Regression | 0.4720810857646023 |  0.5082508140455909 |
|      Linear SVM     | 0.499495327207433  |  0.521583251161899  |
|       XGBoost       |      0.410378      | 0.45399573660014325 |
+---------------------+--------------------+---------------------+
```

In [ ]: