In [0]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
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 GridSearchCV
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

import pickle
```

4. Machine Learning Models

4.1 Reading data from file and storing into sql table

In [0]:
```
#Creating db file from csv
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('Quora/final_features.csv', names=['Unnamed:
 0','id','is_duplicate','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','freq_qid1','freq_qid2','q1len','q2len','q1_n_wo
rds','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','11
7_x','118_x','119_x','120_x','121_x','122_x','123_x','124_x','125_x','1
26_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','17
0_x','171_x','172_x','173_x','174_x','175_x','176_x','177_x','178_x','1
79_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','22
3_x','224_x','225_x','226_x','227_x','228_x','229_x','230_x','231_x','2
32_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','27
6_x','277_x','278_x','279_x','280_x','281_x','282_x','283_x','284_x','2
85_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','32
9_x','330_x','331_x','332_x','333_x','334_x','335_x','336_x','337_x','3
38_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','38
2_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','11
7_y','118_y','119_y','120_y','121_y','122_y','123_y','124_y','125_y','1
26_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','17
0_y','171_y','172_y','173_y','174_y','175_y','176_y','177_y','178_y','1
79_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','22
3_y','224_y','225_y','226_y','227_y','228_y','229_y','230_y','231_y','2
32_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','27
6_y','277_y','278_y','279_y','280_y','281_y','282_y','283_y','284_y','2
85_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','32
9_y','330_y','331_y','332_y','333_y','334_y','335_y','336_y','337_y','3
38_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','38
```

```
2_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
```

```
180000 rows
```

In [0]:
```
#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 [0]:
```
read_db = 'train.db'
conn_r = create_connection(read_db)
checkTableExists(conn_r)
conn_r.close()
```

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

In [0]:
```
# 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 10000
1;""", 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 [0]:
```
# 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 [0]:
```
data.head()
```

Out[0]:

	cwc_min	cwc_max	csc_min	csc_max	
1	0.199996000079998	0.166663888935184	0.0	0.0	0.142
2	0.399992000159997	0.399992000159997	0.499987500312492	0.499987500312492	0.444
3	0.833319444675922	0.714275510349852	0.999983333611106	0.857130612419823	0.687
4	0.0	0.0	0.599988000239995	0.499991666805553	0.249
5	0.749981250468738	0.749981250468738	0.499987500312492	0.499987500312492	0.624

5 rows × 794 columns

4.2 Converting strings to numerics

In [0]:
```
# 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)
```

In [0]:
```
# 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 [0]:
```
X_train,X_test, y_train, y_test = train_test_split(data, y_true, stratify=y_true, test_size=0.3)
```

In [0]:
```
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 [0]:
```
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_dis
tr[1])/test_len)
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.6324857142857143 Class 1:  0.36751428571428574
---------- Distribution of output variable in train data ----------
Class 0:  0.3675 Class 1:  0.3675
```

In [0]:

```
# 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 cl
ass i are predicted class j

    A =(((C.T)/(C.sum(axis=1))).T)
    #divid each element of the confusion matrix with the sum of element
s 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 correspo
nds 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 element
s in that row
    # C = [[1, 2],
    #     [3, 4]]
    # C.sum(axis = 0)  axis=0 corresonds to columns and axis=1 correspo
nds to rows in two diamensional array
```

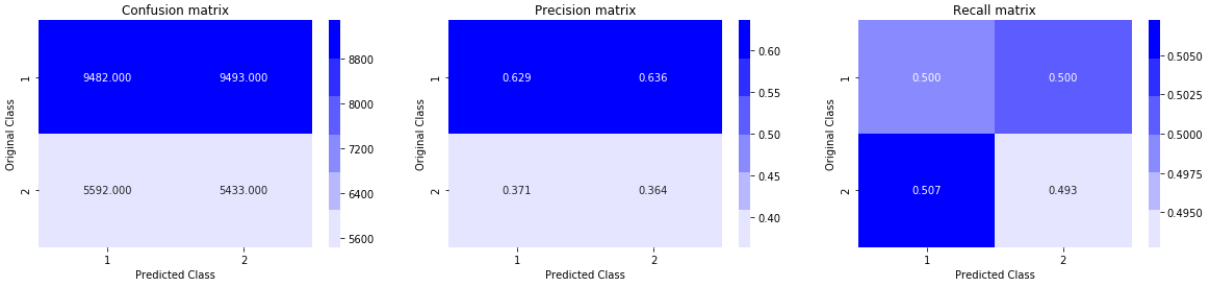
```
    # C.sum(axix =0) = [[4, 6]]
    # (C/C.sum(axis=0)) = [[1/4, 2/6],
    #                      [3/4, 4/6]]
    plt.figure(figsize=(20,4))

    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 [0]:
```
# 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, predi
cted_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 0.887242646958
```



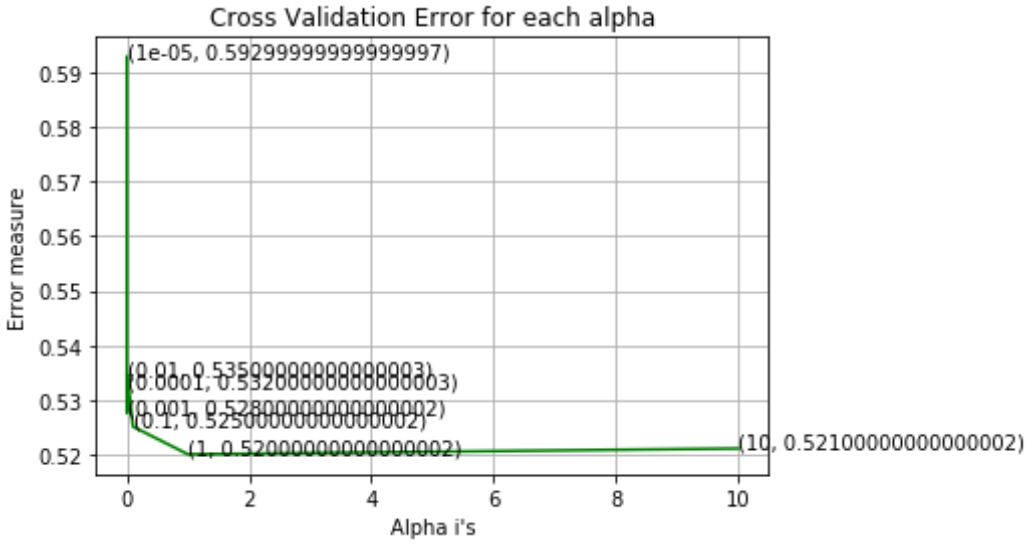
4.4 Logistic Regression with hyperparameter tuning

In [0]:
```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifie
r.

# read more about SGDClassifier() at http://scikit-learn.org/stable/mod
ules/generated/sklearn.linear_model.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
5, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
arning_rate='optimal', eta0=0.0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)
```

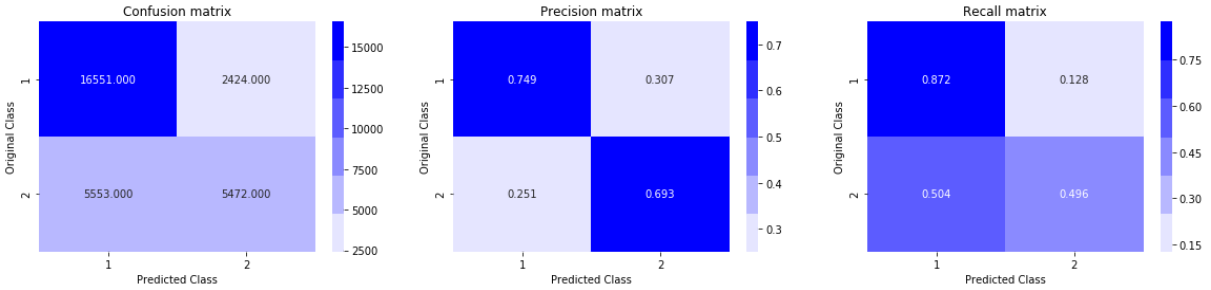
```
# some of methods
# fit(X, y[, coef_init, intercept_init, …])     Fit linear model with S
tochastic Gradient Descent.
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# 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.class
es_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_te
st, predict_y, labels=clf.classes_, 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, predict_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 l
oss is:",log_loss(y_test, predict_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: 0.592800211149
For values of alpha =  0.0001 The log loss is: 0.532351700629
For values of alpha =  0.001 The log loss is: 0.527562275995
For values of alpha =  0.01 The log loss is: 0.534535408885
For values of alpha =  0.1 The log loss is: 0.525117052926
For values of alpha =  1 The log loss is: 0.520035530431
For values of alpha =  10 The log loss is: 0.521097925307
```



```
For values of best alpha =  1 The train log loss is: 0.513842874233
For values of best alpha =  1 The test log loss is: 0.520035530431
Total number of data points : 30000
```



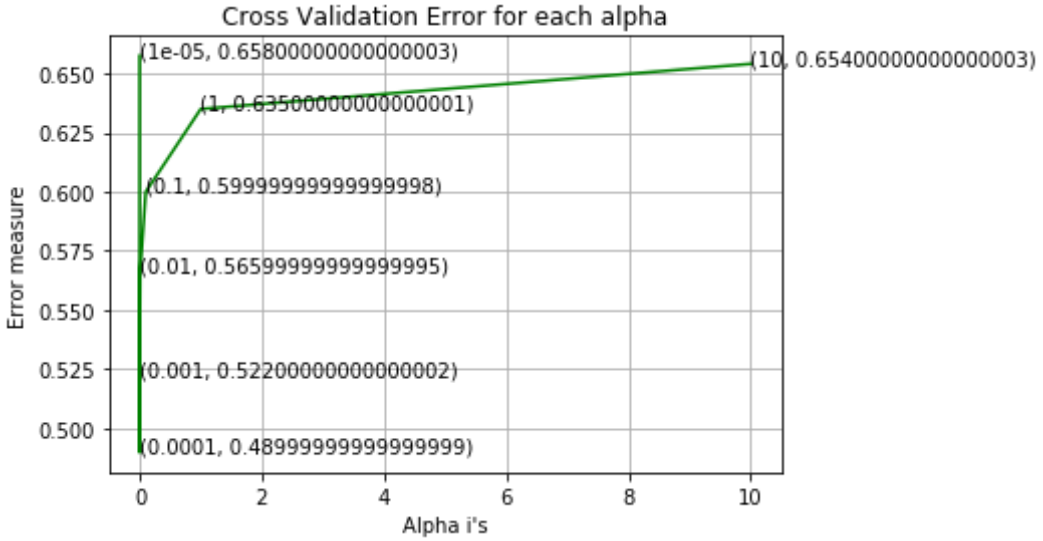
4.5 Linear SVM with hyperparameter tuning

In [0]:
```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifie
r.

# read more about SGDClassifier() at http://scikit-learn.org/stable/mod
ules/generated/sklearn.linear_model.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
5, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
arning_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 S
tochastic Gradient Descent.
# predict(X)    Predict class labels for samples in X.

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

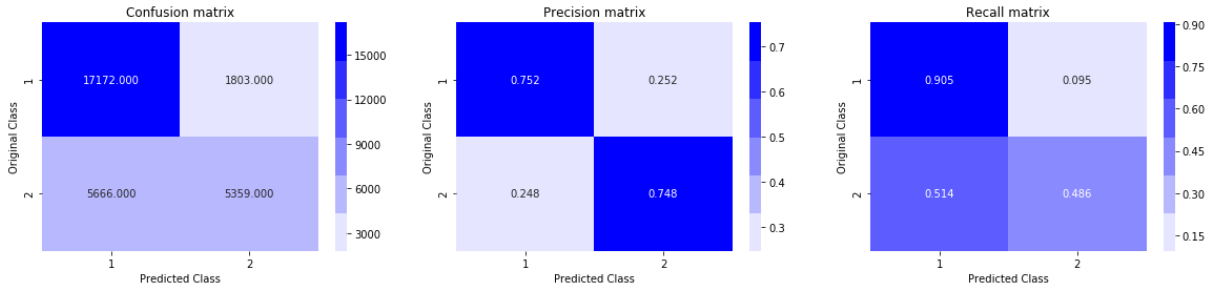
```
log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l1', loss='hinge', random_sta
te=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test,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

```
oss is:",log_loss(y_test, predict_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: 0.657611721261
For values of alpha =  0.0001 The log loss is: 0.489669093534
For values of alpha =  0.001 The log loss is: 0.521829068562
For values of alpha =  0.01 The log loss is: 0.566295616914
For values of alpha =  0.1 The log loss is: 0.599957866217
For values of alpha =  1 The log loss is: 0.635059427016
For values of alpha =  10 The log loss is: 0.654159467907
```




```
For values of best alpha =  0.0001 The train log loss is: 0.47805467728
5
For values of best alpha =  0.0001 The test log loss is: 0.489669093534
Total number of data points : 30000
```



4.6 XGBoost

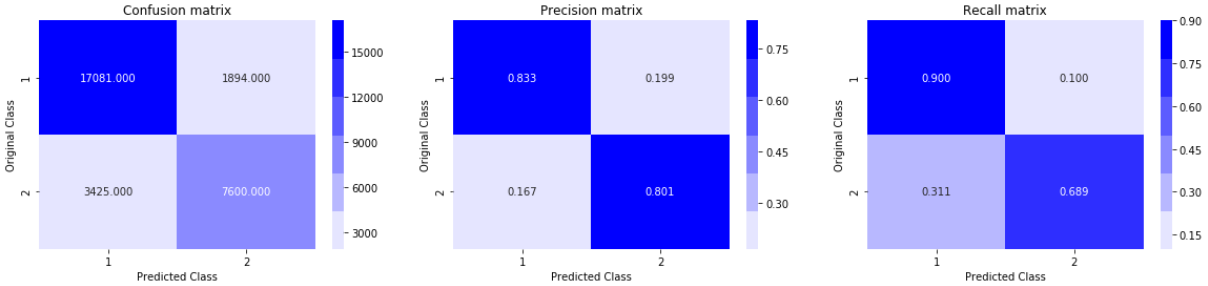
In [0]:
```
import xgboost as xgb
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 4

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.cl
asses_, eps=1e-15))
```

```
[0]     train-logloss:0.684819  valid-logloss:0.684845
Multiple eval metrics have been passed: 'valid-logloss' will be used fo
r early stopping.

Will train until valid-logloss hasn't improved in 20 rounds.
[10]    train-logloss:0.61583   valid-logloss:0.616104
```

```
[20]    train-logloss:0.564616  valid-logloss:0.565273
[30]    train-logloss:0.525758  valid-logloss:0.52679
[40]    train-logloss:0.496661  valid-logloss:0.498021
[50]    train-logloss:0.473563  valid-logloss:0.475182
[60]    train-logloss:0.455315  valid-logloss:0.457186
[70]    train-logloss:0.440442  valid-logloss:0.442482
[80]    train-logloss:0.428424  valid-logloss:0.430795
[90]    train-logloss:0.418803  valid-logloss:0.421447
[100]   train-logloss:0.41069   valid-logloss:0.413583
[110]   train-logloss:0.403831  valid-logloss:0.40693
[120]   train-logloss:0.398076  valid-logloss:0.401402
[130]   train-logloss:0.393305  valid-logloss:0.396851
[140]   train-logloss:0.38913   valid-logloss:0.392952
[150]   train-logloss:0.385469  valid-logloss:0.389521
[160]   train-logloss:0.382327  valid-logloss:0.386667
[170]   train-logloss:0.379541  valid-logloss:0.384148
[180]   train-logloss:0.377014  valid-logloss:0.381932
[190]   train-logloss:0.374687  valid-logloss:0.379883
[200]   train-logloss:0.372585  valid-logloss:0.378068
[210]   train-logloss:0.370615  valid-logloss:0.376367
[220]   train-logloss:0.368559  valid-logloss:0.374595
[230]   train-logloss:0.366545  valid-logloss:0.372847
[240]   train-logloss:0.364708  valid-logloss:0.371311
[250]   train-logloss:0.363021  valid-logloss:0.369886
[260]   train-logloss:0.36144   valid-logloss:0.368673
[270]   train-logloss:0.359899  valid-logloss:0.367421
[280]   train-logloss:0.358465  valid-logloss:0.366395
[290]   train-logloss:0.357128  valid-logloss:0.365361
[300]   train-logloss:0.355716  valid-logloss:0.364315
[310]   train-logloss:0.354425  valid-logloss:0.363403
[320]   train-logloss:0.353276  valid-logloss:0.362595
[330]   train-logloss:0.352084  valid-logloss:0.361823
[340]   train-logloss:0.351051  valid-logloss:0.361167
[350]   train-logloss:0.349867  valid-logloss:0.36043
[360]   train-logloss:0.348829  valid-logloss:0.359773
[370]   train-logloss:0.347689  valid-logloss:0.359019
[380]   train-logloss:0.346607  valid-logloss:0.358311
[390]   train-logloss:0.345568  valid-logloss:0.357674
The test log loss is: 0.357054433715
```

In [0]:
```
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
```



5. Assignments

1. Try out models (Logistic regression, Linear-SVM) with simple TF-IDF vectors instead of TD_IDF weighted word2Vec.
2. Hyperparameter tune XgBoost using RandomSearch to reduce the log-loss.

Note : Here in this assigment considering 100k data points gives memory error. So here in this assignment I have considered 10000 data points

In [0]:
```
final_data = pd.read_csv('Quora/nlp_features_train.csv',encoding = 'latin-1').astype('U')
```

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

Out[0]:
```
Index(['id', 'qid1', 'qid2', 'question1', 'question2', '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'],
      dtype='object')
```

In [0]:
```
final_data.head()
```

Out[0]:

	id	qid1	qid2	question1	question2	is_duplicate	cwc_min	cv
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	0.999980000399992	0.83331944467
1	1	3	4	what is the story of kohinoor koh i noor dia...	what would happen if the indian government sto...	0	0.7999840003199936	0.39999600003
2	2	5	6	how can i increase the speed of my internet co...	how can internet speed be increased by hacking...	0	0.3999920001599968	0.33332777787
3	3	7	8	why am i mentally very lonely how can i solve...	find the remainder when math 23 24 math i...	0	0.0	0.0

	id	qid1	qid2	question1	question2	is_duplicate	cwc_min	cv
4	4	9	10	which one dissolve in water quikly sugar salt...	which fish would survive in salt water	0	0.3999920001599968	0.19999800001

In [0]:
```
sample_df = final_data.sample(n=50000)
sample_df.shape
```

Out[0]: (50000, 21)

In [0]:
```
sample_df.to_csv('Colab Notebooks/sample_features.csv')
```

In [0]:
```
sample_df = pd.read_csv('Colab Notebooks/sample_features.csv').astype(
'U')
```

In [0]:
```
sample_df.shape
```

Out[0]: (50000, 21)

In [0]:
```
sample_df.fillna('')
```

Out[0]:

	id	qid1	qid2	question1	question2	is_duplicate	
59455	59455	78603	104125	how many keywords are there in the s q l prog...	how many keywords are there in p h p programm...	0	0.7499906251

	id	qid1	qid2	question1	question2	is_duplicate	
331173	331173	123992	13239	can humans be immortal	how do you become immortal	1	0.4999750012
186226	186226	142312	284111	how do you stay home and make money in your pa...	what are some good ways to make money sitting ...	1	0.5999880002
269650	269650	387382	75256	how long does it take to become a chartered ac...	how can i become a chartered accountant in india	0	0.7499812504
185402	185402	157661	283049	what is the difference between the centre of g...	what is the difference between centre of mass ...	0	0.7499812504
364857	364857	494939	191351	what is smart home platform	what is a smart home	0	0.9999500024
255458	255458	370360	370361	what is the most useful minor available at a c...	what are the most versatile college majors	0	0.3333222225
267082	267082	374475	182436	how to find lost iphone through imei number	how do i recover my lost iphone 4s using the ...	1	0.7999840003

	id	qid1	qid2	question1	question2	is_duplicate	
189591	189591	288385	288386	oppression which u s citizens are not allowe...	what are some examples of oppression today	0	0.333322222
335827	335827	352540	463113	who is the worst actor in bollywood besides sa...	what is the worst salman khan film	0	0.749981250
183788	183788	280952	280953	what are the steps of becoming a doctor in india	how difficult is it to become a doctor in india	0	0.499987500
251755	251755	365941	365942	why the rainbows are curved in shape	what is the shape of rainbow	0	0.499975001
13311	13311	25574	25575	who is the best orthopedic surgeon in nyc	who own the best orthopedic surgeon in pune	0	0.749981250
76497	76497	130764	130765	who is the best international bulk sms service...	which are the bulk sms service provider compan...	0	0.666655555

	id	qid1	qid2	question1	question2	is_duplicate	
313748	313748	316047	7974	how do i prepare gate exam	what are some tips to prepare for the gate	1	0.6666444451
136917	136917	218333	218334	how do i develop android app using python i a...	what are the constraints in using python to wr...	0	0.4999916668
363760	363760	493773	493774	what are the best one liners that describe the...	if you had to describe your state with a one l...	0	0.3999920001
19347	19347	36564	36565	what is the significance of the cherubim and s...	what are the roles of the cherubim and seraphim	1	0.6666444451
241520	241520	353537	353538	what are the structural advantages of an arcua...	can i construct a partition wall over mid span...	0	0.0
205570	205570	289967	35225	what was the main cause of the chernobyl disas...	what was the main reason behind chernobyl nucl...	1	0.7499812504

	id	qid1	qid2	question1	question2	is_duplicate	
186119	186119	283976	158204	how do i tip for domino own delivery online	i am a pizza hut delivery driver what are som...	0	0.4999875003
158623	158623	128346	51110	why can not a pilot pull back the yoke all the...	why can not an airplane just fly into space	0	0.0
357059	357059	122829	4784	is ice more or less dense than water	is ice lighter than water	1	0.6666444451
325365	325365	55324	451602	what is it like to be a southerner in the nort...	in the united states how much of a difference...	0	0.5999880002
400467	400467	53623	96762	what are the things that makes indians happy	what are the things that make indians happy	1	0.7499812504
310419	310419	434531	434532	i dont know how to talk to people	how can i learn how to talk to people	1	0.6666444451
402073	402073	535517	535518	what happens to cub linux website	what happens if microsoft copies source code f...	0	0.4999875003

	id	qid1	qid2	question1	question2	is_duplicate	
258222	258222	373721	273914	why do people love breaking bad	what is the meaning of breaking bad	0	0.6666444451
216951	216951	16625	323205	how do i kiss for the first time	what are some kissing tips	0	0.0
52177	52177	92431	92432	how do i convince a famous dj to accept my boo...	how should i make a girl accept my friend requ...	0	0.3333277778
...	...	...	...	...	...	...	...
135846	135846	216868	216869	where can i find a list of all of the business...	how do i find all businesses in a given city	0	0.7499812504
240431	240431	275739	352241	how can someone with a mechanical engineering ...	how can engineering technology based on a star...	0	0.3749953125
50222	50222	49078	15606	what is the best tv series and why	which are the best tv series to watch	1	0.9999666677
130138	130138	208912	208913	my device is facing the error internal server...	how do i fix 0xc0k7b error	0	0.6666444451

	id	qid1	qid2	question1	question2	is_duplicate	
68144	68144	117816	117817	what can i do against a ceo	what does a ceo do	0	0.999900009
121948	121948	109308	197495	who is the all time best footballer	what would be the all time best world 11 footb...	0	0.666644445
69907	69907	120578	120579	is it possible to change iso level of photos s...	what is a good and reliable entry level nas w...	0	0.33332962
9977	9977	19372	19373	what is the use of public activity on github ...	is there anyway to hide the public activity on...	1	0.99998333
147551	147551	232845	232846	brazzerss teens in the backseat angel wicky s...	teens in the backseat angel wicky sam bourne ...	1	0.899991000
87240	87240	146966	15432	is 175cm 5 9 tall for a 14 years and 5 months ...	how well can you predict a child own adult h...	0	0.0
318754	318754	444145	121892	how can i make a good career in the data analy...	how do i make career in data analytics	1	0.99997500

	id	qid1	qid2	question1	question2	is_duplicate	
222139	222139	329621	329622	what is the best ide for python programming on...	what is the best free mac python ide for a beg...	0	0.5999880002
141143	141143	110760	224145	what is the difference between keynesian and c...	what is the difference between the classical a...	1	0.7499812504
314555	314555	297676	439295	is magic real how do great magicians perform ...	what are the tricks used by the magicians whil...	0	0.7499812504
366768	366768	171	5314	how can i increase my height after 21 also	how girls can increase their height after 18 y...	1	0.4999875003
115562	115562	188418	188419	why did not sherlock recognize his own sister ...	why did not sherlock recognize his sister eur...	1	0.5999880002
88552	88552	148923	148924	what are some very common mistakes people make...	I am color blind so i am able for applying lie...	0	0.0

	id	qid1	qid2	question1	question2	is_duplicate	
334139	334139	30249	34509	how should californians vote on 2016 own propo...	how should californians vote on 2016 own propo...	0	0.7999840003
195985	195985	296583	296584	what are the best taglines slogans for electro...	does iocl recruit electronics and communicatio...	0	0.1999960000
265049	265049	381951	27912	why did so many americans vote for donald trump	will the americans actually vote donald trump	0	0.7999840003
359924	359924	489626	489627	what are the most interesting products and inn...	what are the most interesting products and inn...	0	0.8749890626
225229	225229	121599	333515	how do i see old snapchat conversations	how do i view my snapchat history	0	0.3333222225
177013	177013	272217	272218	any good songs to workout to	what are some great songs to workout to	1	0.6666444451
217062	217062	323339	19068	does smoking weed cause psychosis	has anyone ever died from smoking marijuana	0	0.2499937501

	id	qid1	qid2	question1	question2	is_duplicate	
288083	288083	408974	408975	what is yoplait yogurt good for	why is yoplait yogurt good for you	1	0.9999666677
264426	264426	381183	381184	what would you think of someone that is suppos...	if you had a chance to interview steve wozniak...	0	0.3333296296
209855	209855	314252	314253	can the relationship between china and japan b...	what is the difference between the sworn enemy...	0	0.4999916668
314131	314131	438824	438825	which are the advantages to use facebook login...	what are the advantages of launching a product...	0	0.5999880002
44250	44250	28716	34406	is quora supporting hillary clinton	why is quora so completely biased towards hill...	1	0.7499812504
330046	330046	23706	76871	why is donald trump not racist	why do some people call donald trump racist	1	0.9999666677

50000 rows × 21 columns

In [0]:
```
# merge texts
```

```
questions = list(sample_df['question1']) + list(sample_df['question2'])
```

In [0]:
```
y = sample_df['is_duplicate']
sample_df.drop(['is_duplicate'],axis =1, inplace=True)
```

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

Out[0]:
```
Index(['id', 'qid1', 'qid2', 'question1', 'question2', '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'],
      dtype='object')
```

In [0]:
```
from sklearn.preprocessing import StandardScaler
sample_df[['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']] = StandardScaler().fit_transform(sample
_df[['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']])
```

In [0]:
```
X_train,X_test, y_train, y_test = train_test_split(sample_df, y, strati
fy=y, test_size=0.3)
```

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer


tfidf = TfidfVectorizer(lowercase=False, max_features= 2000)
vec_tr = tfidf.fit_transform(list(X_train['question1'])+list(X_train['q
uestion2']))
```

```
vec_te = tfidf.transform(list(X_test['question1'])+list(X_test['questio
n2']))
```

In [0]:
```
def to_df(data):
  len_ = int(data.shape[0]/2)
  q1 = data[:len_].todense()
  q2 = data[len_:].todense()

  idx_x = ['x_' + str(i) for i in range(0,q1.shape[1])]
  idx_y = ['y_' + str(i) for i in range(0,q2.shape[1])]

  q1_df = pd.DataFrame(q1,columns= idx_x)
  q2_df = pd.DataFrame(q2,columns= idx_y)

  tfidf_features = pd.concat([q1_df,q2_df],axis = 1)

  return tfidf_features
```

In [0]:
```
X_tr = to_df(vec_tr)
```

In [0]:
```
X_te = to_df(vec_te)
```

In [0]:
```
import pickle
with open('Colab Notebooks/x_tr.pkl','wb') as f:
  pickle.dump((X_tr,y_train),f)

with open('Colab Notebooks/x_te.pkl','wb') as f:
  pickle.dump((X_te,y_test),f)
```

In [0]:
```
y_train = list(map(int,y_train))
y_test = list(map(int,y_test))
```

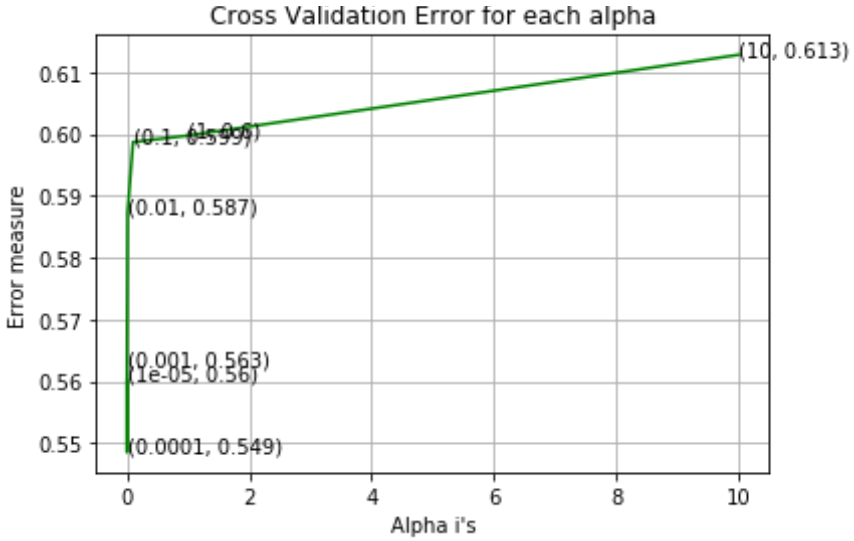
Logistic Regression with simple tf-idf

In [0]:
```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifie
```

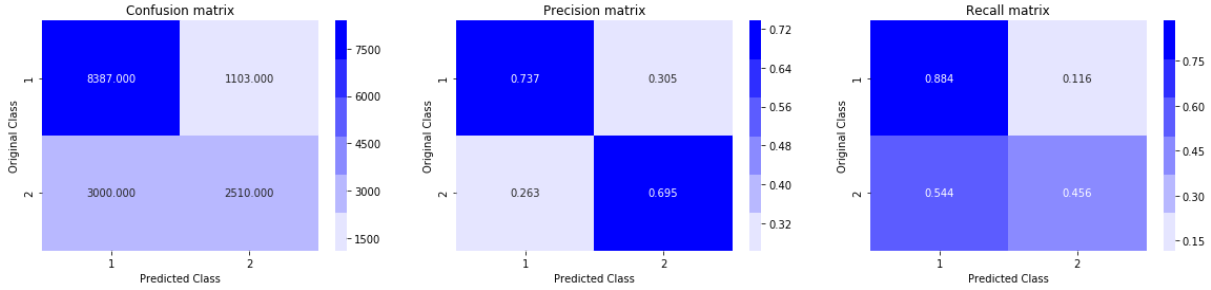
```
r.

# read more about SGDClassifier() at http://scikit-learn.org/stable/mod
ules/generated/sklearn.linear_model.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
5, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
arning_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 S
tochastic 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', class_weight
 = 'balanced',random_state=42)
    clf.fit(X_tr, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_tr, y_train)
    predict_y = sig_clf.predict_proba(X_te)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.class
es_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_te
st, predict_y, labels=clf.classes_, 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',
class_weight = 'balanced',random_state=42)
clf.fit(X_tr, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_tr, y_train)

predict_y = sig_clf.predict_proba(X_tr)
print('For values of best alpha = ', alpha[best_alpha], "The train log
 loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(X_te)
print('For values of best alpha = ', alpha[best_alpha], "The test log l
oss is:",log_loss(y_test, predict_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: 0.5601654064024454
For values of alpha =  0.0001 The log loss is: 0.5485267388634054
For values of alpha =  0.001 The log loss is: 0.5626700277585442
For values of alpha =  0.01 The log loss is: 0.5872810036061259
For values of alpha =  0.1 The log loss is: 0.5986925516027887
For values of alpha =  1 The log loss is: 0.5997493314728551
For values of alpha =  10 The log loss is: 0.6128037545846011
```




```
For values of best alpha =  0.0001 The train log loss is: 0.51640997883
0354
For values of best alpha =  0.0001 The test log loss is: 0.548526738863
4054
Total number of data points : 15000
```



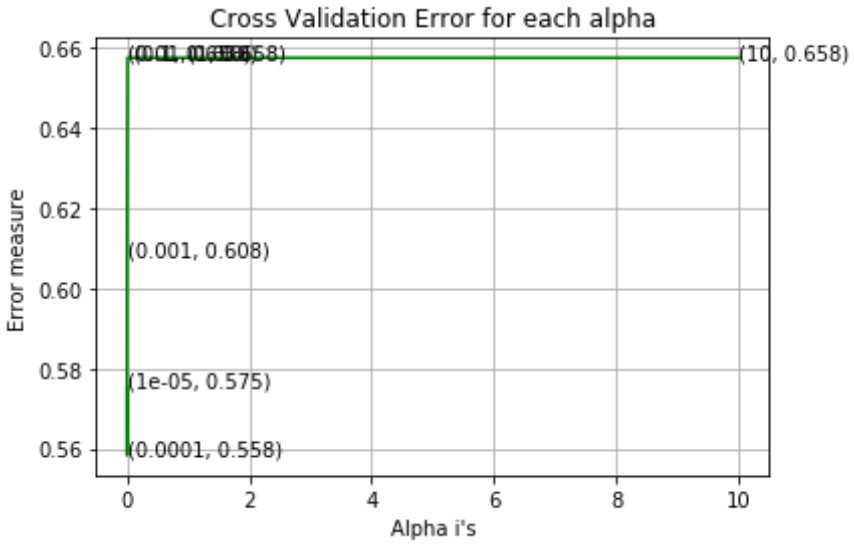
Linear SVM with simple tf-idf

In [0]:
```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifie
r.
```

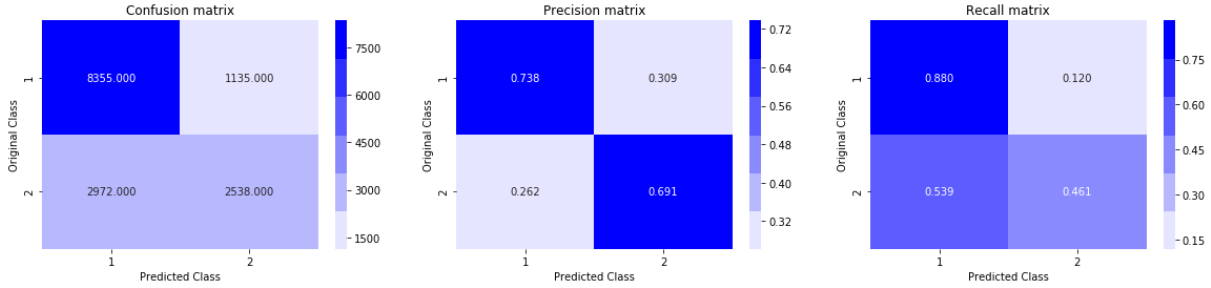
```
# read more about SGDClassifier() at http://scikit-learn.org/stable/mod
ules/generated/sklearn.linear_model.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
5, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
arning_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 S
tochastic 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',class_weigh
t ='balanced', random_state=42)
    clf.fit(X_tr, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_tr, y_train)
    predict_y = sig_clf.predict_proba(X_te)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.class
es_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_te
st, predict_y, labels=clf.classes_, 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'
, class_weight = 'balanced', random_state=42)
clf.fit(X_tr, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_tr, y_train)

predict_y = sig_clf.predict_proba(X_tr)
print('For values of best alpha = ', alpha[best_alpha], "The train log
 loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(X_te)
print('For values of best alpha = ', alpha[best_alpha], "The test log l
oss is:",log_loss(y_test, predict_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: 0.575299672471695
For values of alpha =  0.0001 The log loss is: 0.5582927884622025
For values of alpha =  0.001 The log loss is: 0.6080293190705774
For values of alpha =  0.01 The log loss is: 0.657521167470038
For values of alpha =  0.1 The log loss is: 0.6575211674700377
For values of alpha =  1 The log loss is: 0.657521167470038
For values of alpha =  10 The log loss is: 0.657521167470038
```




```
For values of best alpha =  0.0001 The train log loss is: 0.5323848515475809
For values of best alpha =  0.0001 The test log loss is: 0.5582927884622025
Total number of data points : 15000
```



Hyperparameter tune XgBoost using RandomSearch

In [0]:
```
df = pd.read_csv("Quora/train.csv")
df = df.sample(n=50000)
# 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 [0]:
```
y = df['is_duplicate']
```

In [0]:
```
df.shape , len(y)
```

Out[0]:
```
((50000, 6), 50000)
```

In [0]:
```
X_tr,X_te,y_tr,y_te = train_test_split(df,y,stratify=y, test_size=0.3)
```

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

tfidf = TfidfVectorizer(lowercase=False, max_features=2000)
tfidf.fit_transform(X_tr)



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

In [0]:
```
import spacy
from tqdm import tqdm

nlp = spacy.load('en_core_web_sm')
def Vectorize(file):
```

```
# merge texts
questions = list(file['question1']) + list(file['question2'])
vecs1 = []
vecs2 = []
# https://github.com/noamraph/tqdm
# tqdm is used to print the progress bar
for qu1 in tqdm(list(file['question1'])):
    doc1 = nlp(qu1)
    # 384 is the number of dimensions of vectors
    mean_vec1 = np.zeros([len(doc1), len(doc1[0].vector)])
    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)

for qu1 in tqdm(list(file['question2'])):
    doc1 = nlp(qu1)
    # 384 is the number of dimensions of vectors
    mean_vec1 = np.zeros([len(doc1), len(doc1[0].vector)])
    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)
    vecs2.append(mean_vec1)
```

```
  file['q1_feats_m'] = list(vecs1)
  file['q2_feats_m'] = list(vecs1)
  return file
```

In [0]:
```
X_tr_tfidf = Vectorize(X_tr)
```

```
100%|██████████| 35000/35000 [07:31<00:00, 77.54it/s]
100%|██████████| 35000/35000 [07:12<00:00, 80.88it/s]
```

In [0]:
```
if os.path.isfile('Quora/nlp_features_train.csv'):
    dfnlp = pd.read_csv("Quora/nlp_features_train.csv",encoding='latin-
1')
else:
    print("download nlp_features_train.csv from drive or run previous n
otebook")

if os.path.isfile('Quora/df_fe_without_preprocessing_train.csv'):
    dfppro = pd.read_csv("Quora/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 [0]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'
],axis=1)
df3 = X_tr_tfidf.drop(['qid1','qid2','question1','question2','is_duplic
ate'],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 [0]:
```
df3_q1['id']=df1['id']
df3_q2['id']=df1['id']
df2  = df3_q1.merge(df3_q2, on='id',how='left')
df1  = df1.merge(df2, on='id',how='left')
result_tr  = df2.merge(df1, on='id',how='left')
```

In [0]:
```
result_tr.shape
```

Out[0]:
```
(35000, 1553)
```

In [0]:
```
import pickle

with open('result_tr.pkl','wb') as f:
  pickle.dump((result_tr,y_tr),f)
```

In [0]:
```
result_tr,y_tr = pickle.load(open('result_tr.pkl','rb'))
```

In [0]:
```
X_te_tfidf = Vectorize(X_te)
```

```
100%|██████████| 15000/15000 [03:15<00:00, 76.87it/s]
100%|██████████| 15000/15000 [03:15<00:00, 82.78it/s]
```

In [0]:
```
X_te_tfidf.shape
```

Out[0]:
```
(15000, 8)
```

In [0]:
```
df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'
],axis=1)
df3 = X_te_tfidf.drop(['qid1','qid2','question1','question2','is_duplic
ate'],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 [0]:
```
df3_q1['id']=df1['id']
df3_q2['id']=df1['id']
df2  = df3_q1.merge(df3_q2, on='id',how='left')
df1  = df1.merge(df2, on='id',how='left')
result_te  = df2.merge(df1, on='id',how='left')
```

In [0]:
```
import pickle
```

```
with open('result_te.pkl','wb') as f:
  pickle.dump((result_te,y_te),f)
```

In [0]:
```
with open('result_te.pkl','rb') as f:
  result_te,y_te = pickle.load(f)
```

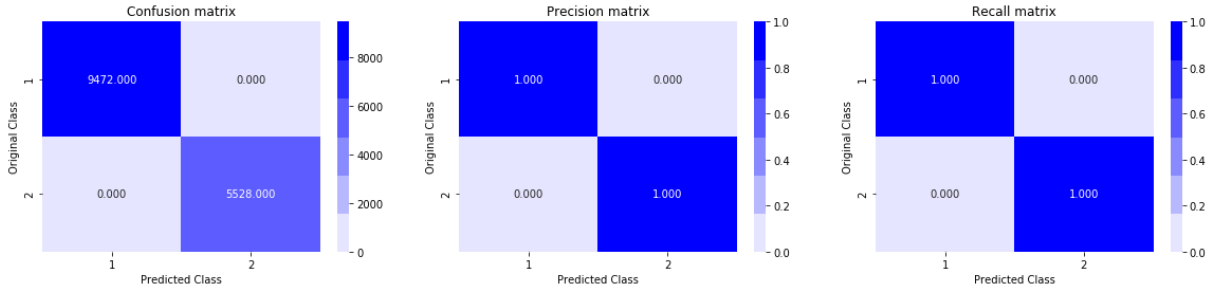
In [11]:
```
result_te.shape
```

Out[11]:
```
(15000, 1553)
```

In [0]:
```
y_tr = list(map(int,y_tr.values))
y_te = list(map(int,y_te.values))
```

In [0]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier

def XGBoost(x_train,y_train,x_test,y_test):
  param = {'n_estimators': [1, 4, 16,32],
          'max_depth': [1,2,3,4,5],
          'learning_rate':[0.1,0.2,0.3]}
  clf = XGBClassifier(objective = 'binary:logistic',eval_metric = 'logl
oss')
  grid_Search = GridSearchCV(clf,param_grid  = param, cv = 3, n_jobs =
-1,return_train_score = True)
  grid_Search.fit(x_train,y_train)

  y_pred = grid_Search.predict(x_test)
  log_loss_value = log_loss(y_test, y_pred,  eps=1e-15)
  best_params = grid_Search.best_params_

  plot_confusion_matrix(y_test, y_pred)
  return print('best_params: ',best_params, '\nlog loss: ', log_loss_va
lue)
```

In [24]:
```
XGBoost(result_tr,y_tr,result_te,y_te)
```




```
best_params:  {'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 1}

log loss:  9.992007221626413e-16
```

Here above in XGBoost log loss is 9.992007221626413e-16 at value of larning rate 0.1 ,max depth 1 and n_estimators 1.

In [0]:
```
df = pd.DataFrame({"Model":["Logistic Regression with weighted Tf-idf "
,"Linear SVM with weighted Tf-idf","XGBoost without hyperparam tuning",
"Logistic Regression with simple tf-idf","Linear SVM with simple tf-id
f"],
                   "Train log loss":[0.513842874233,0.478054677285,0.34
5568,0.516409978830354,0.5323848515475809],
                                      "Test log loss":[0.520035530431
,0.489669093534,0.357054433715,0.5485267388634054,0.5582927884622025]}
                  ,columns=["Model","Train log loss","Test log loss"])
df.sort_values(by="Test log loss",ascending=False)
```

Out[0]:

	Model	Train log loss	Test log loss
4	Linear SVM with simple tf-idf	0.532385	0.558293
3	Logistic Regression with simple tf-idf	0.516410	0.548527
0	Logistic Regression with weighted Tf-idf	0.513843	0.520036
1	Linear SVM with weighted Tf-idf	0.478055	0.489669
2	XGBoost without hyperparam tuning	0.345568	0.357054

Above results show that weighted tf-idf works better than simple tf-idf.