In [1]:
```
import warnings
warnings.filterwarnings("ignore")
import shutil
import os
import pandas as pd
import matplotlib
matplotlib.use(u'nbAgg')
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pickle
from sklearn.manifold import TSNE
from csv import writer
from sklearn import preprocessing
import pandas as pd
from multiprocessing import Process# this is used for multithreading
import multiprocessing
import codecs# this is used for file operations
import scipy
import random as r
import array
from xgboost import XGBClassifier
import pickle as pkl
from sklearn.model_selection import RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import log_loss
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    print("Number of misclassified points ",(len(test_y)-np.trace(C))/len(test_y)*100)
    # 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]]

    labels = [1,2,3,4,5,6,7,8,9]
    cmap=sns.light_palette("green")
    # representing A in heatmap format
    print("-"*50, "Confusion matrix", "-"*50)
    plt.figure(figsize=(10,5))
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    print("-"*50, "Precision matrix", "-"*50)
    plt.figure(figsize=(10,5))
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("Sum of columns in precision matrix",B.sum(axis=0))

    # representing B in heatmap format
    print("-"*50, "Recall matrix"    , "-"*50)
    plt.figure(figsize=(10,5))
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("Sum of rows in precision matrix",A.sum(axis=1))
```

1.Image Extraction from ASM -files

Image Extraction using : https://github.com/dchad/malware-detection/blob/master/mmcc/feature-extraction.ipynb (https://github.com/dchad/malware-detection/blob/master/mmcc/feature-extraction.ipynb)

In [3]:
```
def entropy(p,n):
    '''Returns entrophy'''
    p_ratio = float(p)/(p+n)
    n_ratio = float(n)/(p+n)
    return -p_ratio*math.log(p_ratio) - n_ratio * math.log(n_ratio)
```

In [4]:
```
def info_gain(p0,n0,p1,n1,p,n):
    '''Return iformation gain'''
    return entropy(p,n) - float(p0+n0)/(p+n)*entropy(p0,n0) - float(p1+n1)/(p+n)*entropy(p1,n1)
```

In [5]:
```
def read_image(filename):
    '''
    Read the image data
    '''
    f = open(filename,'rb')
    ln = os.path.getsize(filename) # length of file in bytes
    width = 256
    rem = ln%width
    a = array.array("B") # uint8 array
    a.fromfile(f,ln-rem)
    f.close()
    g = np.reshape(a,(int(len(a)/width), width))
    g = np.uint8(g)
    g = np.resize(g, (1000,))
    return list(g)
```

In [6]:
```
# Do asm image extraction
def extract_asm_image_features(tfiles):
    asm_files = [i for i in tfiles if '.asm' in i]
    ftot = len(asm_files)

    pid = os.getpid()
    print('Process id:', pid)
    feature_file = str(pid) + '-image-features-asm.csv'
    print('feature file:', feature_file)

    outrows = []
    with open(feature_file,'w') as f:
        fw = writer(f)
        column_names = ['filename'] + [("ASM_{:s}".format(str(x))) for x in range(1000)]
        fw.writerow(column_names)
        for idx, fname in enumerate(asm_files):
            file_id = fname.split('.')[0]
            image_data = read_image('asmFiles/' + fname)
            outrows.append([file_id] + image_data)

            # Print progress
            if (idx+1) % 10 == 0:
              print(pid, idx + 1, 'of', ftot, 'files processed.')
              fw.writerows(outrows)
              outrows = []

        # Write remaining files
        if len(outrows) > 0:
            fw.writerows(outrows)
            outrows = []
```

In [7]:

```
# TRAIN FILES ASM
# Now divide the train files into four groups for multiprocessing
import time
from multiprocessing import Pool
start_time = time.time()
tfiles =  os.listdir('asmFiles')
quart = len(tfiles)//4
train1 = tfiles[:quart]
train2 = tfiles[quart:(2*quart)]
train3 = tfiles[(2*quart):(3*quart)]
train4 = tfiles[(3*quart):]
print (len(tfiles), quart, (len(train1)+len(train2)+len(train3)+len(train4)))
trains = [train1, train2, train3, train4]
p = Pool(4)
p.map(extract_asm_image_features, trains)
print("Elapsed time: {:.2f} hours.".format((time.time() - start_time)//3600.0))
```

```
10868 2717 10868
Process id:Process id:Process id:Process id:    1139114011381137



feature file:feature file:feature file:feature file:    1139-image-features-asm.csv1140-image-features-asm.csv1138-image-features-asm.csv1137-image-features-asm.csv



1138 10 of 2717 files processed.
1137 10 of 2717 files processed.
1140 10 of 2717 files processed.
1138 20 of 2717 files processed.
1139 10 of 2717 files processed.
1137 20 of 2717 files processed.
1138 30 of 2717 files processed.
1140 20 of 2717 files processed.
1139 20 of 2717 files processed.
1137 30 of 2717 files processed.
1140 30 of 2717 files processed.
1138 40 of 2717 files processed.
1138 50 of 2717 files processed.
1139 30 of 2717 files processed.
1140 40 of 2717 files processed.
1137 40 of 2717 files processed.
1139 40 of 2717 files processed.
1138 60 of 2717 files processed.
1137 50 of 2717 files processed.
1138 70 of 2717 files processed.
1139 50 of 2717 files processed.
1140 50 of 2717 files processed.
1140 60 of 2717 files processed.
1138 80 of 2717 files processed.
1139 60 of 2717 files processed.
1137 60 of 2717 files processed.
1140 70 of 2717 files processed.
1139 70 of 2717 files processed.
1137 70 of 2717 files processed.
1138 90 of 2717 files processed.
1139 80 of 2717 files processed.
1138 100 of 2717 files processed.
1140 80 of 2717 files processed.
1137 80 of 2717 files processed.
1140 90 of 2717 files processed.
1138 110 of 2717 files processed.
1137 901139 90  ofof 2717  2717files processed.
 files processed.
1140 100 of 2717 files processed.
1138 120 of 2717 files processed.
1139 100 of 2717 files processed.
1137 100 of 2717 files processed.
1140 110 of 2717 files processed.
1139 110 of 2717 files processed.
1138 130 of 2717 files processed.
1137 110 of 2717 files processed.
1139 120 of 2717 files processed.
1140 120 of 2717 files processed.
1137 120 of 2717 files processed.
1139 130 of 2717 files processed.
1140 130 of 2717 files processed.
1138 140 of 2717 files processed.
1138 150 of 2717 files processed.
1140 140 of 2717 files processed.
1138 160 of 2717 files processed.
1137 130 of 2717 files processed.
1139 140 of 2717 files processed.
1138 170 of 2717 files processed.
1137 140 of 2717 files processed.
1140 150 of 2717 files processed.
1139 150 of 2717 files processed.
1138 180 of 2717 files processed.
1139 160 of 2717 files processed.
1138 190 of 2717 files processed.
1137 150 of 2717 files processed.
1139 170 of 2717 files processed.
1140 160 of 2717 files processed.
1139 180 of 2717 files processed.1140 170
 of 2717 files processed.
1137 160 of 2717 files processed.
1139 190 of 27171138  files processed.200
 of 2717 files processed.
1137 170 of 2717 files processed.
1140 180 of 2717 files processed.
1140 190 of 2717 files processed.
1139 200 of 2717 files processed.
1137 180 of 2717 files processed.
1138 210 of 2717 files processed.
1140 200 of 2717 files processed.
1138 220 of 2717 files processed.
1139 210 of 2717 files processed.
1137 190 of 2717 files processed.
1138 230 of 2717 files processed.
1140 210 of 2717 files processed.
1137 2001138  240 ofof 2717  files processed.2717
files processed.
1139 220 of 2717 files processed.
1140 220 of 2717 files processed.
1137 210 of 2717 files processed.
1139 230 1140of 2717 files processed.
 230 of 2717 files processed.
1138 250 of 2717 files processed.
1139 240 of 2717 files processed.
1137 220 of 2717 files processed.
1138 260 of 2717 files processed.
1140 240 of 2717 files processed.
1139 250 of 2717 files processed.
1137 230 of 2717 files processed.
1140 250 of 2717 files processed.
1138 270 of 2717 files processed.
1137 240 of 2717 files processed.
1139 260 of 2717 files processed.
1139 270 of 2717 files processed.
1140 260 of 2717 files processed.
1138 280 of 2717 files processed.
1139 280 of 2717 files processed.
1139 290 of 2717 files processed.
1138 290 of 2717 files processed.
1137 250 of 2717 files processed.
1140 270 of 2717 files processed.
1137 260 of 2717 files processed.
1139 300 of 2717 files processed.
1138 300 of 2717 files processed.
1137 270 of 2717 files processed.
1139 310 of 2717 files processed.
1138 310 of 2717 files processed.
1140 280 of 2717 files processed.
1137 280 of 2717 files processed.
1140290  of 2717 files processed.
1138 320 of 2717 files processed.
1137 290 of 2717 files processed.
1140 300 of 2717 files processed.
1139 320 of 2717 files processed.
1137 300 of 2717 files processed.
1138 330 of 2717 files processed.
1139 330 of 2717 files processed.
1138 340 of 2717 files processed.
1137 310 of 2717 files processed.
1139 340 of 2717 files processed.
1140 310 of 2717 files processed.
1138 350 of 2717 files processed.
1137 320 of 2717 files processed.
1140 320 of 2717 files processed.
1138 360 of 2717 files processed.
1140 330 of 2717 files processed.
1139 350 of 2717 files processed.
1137 330 of 2717 files processed.
1138 370 of 2717 files processed.
1140 340 of 2717 files processed.
1137 340 of 2717 files processed.
1139 360 of 2717 files processed.
1139 370 of 2717 files processed.
1138 380 of 2717 files processed.
1140 350 of 2717 files processed.
1137 350 of 2717 1139files processed.
 380 of 2717 files processed.
11391138  390 390of  of 27172717 files processed. files processed.
```

```
1138 400 of 2717 files processed.
1140 360 of 2717 files processed.
1139 400 of 2717 files processed.
1137 360 of 2717 files processed.
1138 410 of 2717 files processed.
1140 370 of 2717 files processed.
1139 410 of 2717 files processed.
1138 420 of 2717 files processed.
1139 420 of 2717 files processed.
1140 380 of 2717 files processed.
1137 370 of 2717 files processed.
1139 430 of 2717 files processed.
1138 430 of 2717 files processed.
1139 440 of 2717 files processed.
1140 390 of 2717 files processed.
1140 400 of 2717 files processed.
1139 450 of 2717 files processed.
1138 440 of 2717 files processed.
1137 380 of 2717 files processed.
1138 450 of 2717 files processed.
1140 410 of 2717 files processed.
1139 460 of 2717 files processed.
1139 470 of 2717 files processed.
1137 390 of 2717 files processed.
1138 460 of 2717 files processed.
1139 480 of 2717 files processed.
1137 400 of 2717 files processed.
1139 490 of 2717 files processed.
1140 420 of 2717 files processed.
1138 470 of 2717 files processed.
1139 500 of 2717 files processed.
1137 410 of 2717 files processed.
1140 430 of 2717 files processed.
1138 480 of 2717 files processed.
1140 440 of 2717 files processed.
1139 510 of 2717 files processed.
1140 450 of 2717 files processed.
1137 420 of 2717 files processed.
1138 490 of 2717 files processed.
1139 520 of 2717 files processed.
1137 430 of 2717 files processed.
1139 530 of 2717 files processed.
1138 500 of 2717 files processed.
1137 440 of 2717 files processed.
1140 460 of 2717 files processed.
1138 510 of 2717 files processed.
1137 450 of 2717 files processed.
1138 520 of 2717 files processed.
1139 540 of 2717 files processed.
1137 460 of 2717 files processed.
1140 470 of 2717 files processed.
1138 530 of 2717 files processed.
1139 550 of 2717 files processed.
1137 470 of 2717 files processed.
1140 480 of 2717 files processed.
1138 540 of 2717 files processed.
1140 490 of 2717 files processed.
1139 560 of 2717 files processed.
1137 480 of 2717 files processed.
1140 500 of 2717 files processed.
1138 550 of 2717 files processed.
1138 560 of 2717 files processed.
1137 490 of 2717 files processed.
1139 570 of 2717 files processed.
1140 510 of 2717 files processed.
1137 500 of 2717 files processed.
1140 520 of 2717 files processed.
1138 570 of 2717 files processed.
1139 580 of 2717 files processed.
1139 590 of 2717 files processed.
1138 580 of 2717 files processed.
1140 530 of 2717 files processed.
1137 510 of 2717 files processed.
1138 590 of 2717 files processed.
1140 540 of 2717 files processed.
1137 520 of 2717 files processed.
1140 550 of 2717 files processed.
1139 600 of 2717 files processed.
1137 530 of 2717 files processed.
1138 600 of 2717 files processed.
1139 610 of 2717 files processed.
1140 560 of 2717 files processed.
1137 540 of 2717 files processed.
1138 610 of 2717 files processed.
1139 620 of 2717 files processed.
1140 570 of 2717 files processed.
1137 550 of 2717 files processed.
1139 630 of 2717 files processed.
1137 560 of 2717 files processed.
1140 580 of 2717 files processed.
1137 570 of 2717 files processed.
1139 640 of 2717 files processed.
1137 580 of 2717 files processed.
1138 620 of 2717 files processed.
1140 590 of 2717 files processed.
1137 590 of 2717 files processed.
1139 650 of 2717 files processed.
1140 600 of 2717 files processed.
1138 630 of 2717 files processed.
1140 610 of 2717 files processed.
1137 600 of 2717 files processed.
1138 640 of 2717 files processed.
1139 660 of 2717 files processed.
1137 610 of 2717 files processed.
1138 650 of 2717 files processed.
1139 670 of 2717 files processed.
1140 620 of 2717 files processed.
1138 660 of 2717 files processed.
1137 620 of 2717 files processed.
1139 680 of 2717 files processed.
1140 630 of1138 670  2717of  files processed.2717
 files processed.
1137 630 of 2717 files processed.
1140 640 of 2717 files processed.
1139 690 of 2717 files processed.
1139 700 of 2717 files processed.
1138 680 of 2717 files processed.
1137 640 of 2717 files processed.
1137 650 of 2717 files processed.
1139 710 of 2717 files processed.
1138 690 of 2717 files processed.
1140 650 of 2717 files processed.
1137 660 of 2717 files processed.
1139 720 of 2717 files processed.
1138 700 of 2717 files processed.
1139 730 of 2717 files processed.
1140 660 of 2717 files processed.
1137 670 of 2717 files processed.
1139 740 of 2717 files processed.
1137 680 of 2717 files processed.
1138 710 of 2717 files processed.
1139 750 of 2717 files processed.
1137 690 of 2717 files processed.
1139 760 of 2717 files processed.
1138 720 of 2717 files processed.
1140 670 of 2717 files processed.
1137 700 of 2717 files processed.
1139 770 of 2717 files processed.
1137 710 of 2717 files processed.
1140 680 of 2717 files processed.
1138 730 of 2717 files processed.
1137 720 of 2717 files processed.
1140 690 of 2717 files processed.
1138 740 of 2717 files processed.
1139 780 of 2717 files processed.
1137 730 of 2717 files processed.
1140 700 of 2717 files processed.
1138 750 of 2717 files processed.
1137 740 of 2717 files processed.
1138 760 of 2717 files processed.
1138 770 of 2717 files processed.
1140 710 of 27171137  750files processed.
 of 2717 files processed.
1139 790 of 2717 files processed.
1140 720 of 2717 files processed.
1137 760 of 2717 files processed.
1140 730 of 2717 files processed.
1138 780 of 2717 files processed.1140 740
 of 2717 files processed.
1139 800 of 2717 files processed.
1137 770 of 2717 files processed.
1139 810 of 2717 files processed.
1140 750 of 2717 files processed.
1137 780 of 2717 files processed.
1138 790 of 2717 files processed.
1137 790 of 2717 files processed.
1139 820 of 2717 files processed.
1138 800 of 2717 files processed.
1140 760 of 2717 files processed.
1138 810 of 2717 files processed.
1137 800 of 2717 files processed.
1139 830 of 2717 files processed.
1140 770 of 2717 files processed.
1137 810 of 2717 files processed.
1137 820 of 2717 files processed.
```

```
1138 820 of 2717 files processed.
1140 780 of 2717 files processed.
1137 830 of 2717 files processed.
1138 830 of 2717 files processed.
1140 790 of 2717 files processed.
1137 840 of 2717 files processed.
1139 840 of 2717 files processed.
1137 850 of 2717 files processed.
1138 840 of 2717 files processed.
1140 800 of 2717 files processed.
1139 850 of 2717 files processed.
1140 810 of 2717 files processed.
1137 860 of 2717 files processed.
1138 850 of 2717 files processed.
1140 820 of 2717 files processed.
1139 860 of 2717 files processed.
1138 860 of 2717 files processed.
1137 870 of 2717 files processed.
1138 870 of 2717 files processed.
1139 870 of 2717 files processed.
1140 830 of 2717 files processed.
1137 880 of 2717 files processed.
1139 880 of 2717 files processed.
1139 890 of 2717 files processed.
1138 880 of 2717 files processed.
1140 840 of 2717 files processed.
1138 890 of 2717 files processed.
1137 890 of 2717 files processed.
1140 850 of 2717 files processed.
1138 900 of 2717 files processed.
1140 860 of 2717 files processed.
1137 900 of 2717 files processed.
1139 900 of 2717 files processed.
1137 910 of 2717 files processed.
1140 870 of 2717 files processed.
1139 910 of 2717 files processed.
1138 910 of 2717 files processed.
1139 920 of 2717 files processed.
1140 880 of 2717 files processed.
1137 920 of 2717 files processed.
1139 930 of 2717 files processed.
1138 920 of 2717 files processed.
1137 930 of 2717 files processed.
1140 890 of 2717 files processed.
1138 930 of 2717 files processed.
1139 940 of 2717 files processed.
1137 940 of 2717 files processed.
1137 950 of 2717 files processed.
1139 950 of 2717 files processed.
1140 900 of 2717 files processed.
1138 940 of 2717 files processed.
1137 960 of 2717 files processed.
1139 960 of 2717 files processed.
1137 970 of 2717 files processed.
1139 970 of 2717 files processed.
1139 980 of 2717 files processed.
1137 980 of 2717 files processed.
1140 910 of 2717 files processed.
1140 920 of 2717 files processed.
1138 950 of 2717 files processed.
1137 990 of 2717 files processed.
1139 990 of 2717 files processed.
1138 960 of 2717 files processed.
1140 930 of 2717 files processed.
1139 1000 of 2717 files processed.
1138 970 of 2717 files processed.
1139 1010 of 2717 files processed.
1137 1000 of 2717 files processed.
1140 940 of 2717 files processed.
1138 980 of 2717 files processed.
1139 1020 of 2717 files processed.
1140 950 of 2717 files processed.
1139 1030 of 2717 files processed.
1140 960 of 2717 files processed.
1137 1010 of 2717 files processed.
1139 1040 of 2717 files processed.
1137 1020 of 2717 files processed.
1138 990 of 2717 files processed.
1140 970 of 2717 files processed.
1139 1050 of 2717 files processed.
1138 1000 of 2717 files processed.
1137 1030 of 2717 files processed.
1140 980 of 27171138  files processed.
1010 of 2717 files processed.
1137 1040 of 2717 files processed.
1138 1020 of 2717 files processed.
1139 1060 of 2717 files processed.
1138 1030 of 2717 files processed.
1140 990 of 2717 files processed.
1139 1070 of 2717 files processed.
1138 1040 of 2717 files processed.
1138 1050 of 2717 files processed.
1137 1050 of 2717 files processed.
1139 1080 of 2717 files processed.
1140 1000 of 2717 files processed.
1140 1010 of 2717 files processed.
1137 1060 of 2717 files processed.
1139 1090 of 2717 files processed.
1139 1100 of 2717 files processed.
1138 1060 of 2717 files processed.
1140 1020 of 2717 files processed.
1139 1110 of 2717 files processed.
1137 1070 of 2717 files processed.
1137 1080 of 2717 files processed.
1139 1120 of 2717 files processed.
1140 1030 of 2717 files processed.
1137 1090 of 2717 files processed.
1138 1070 of 2717 files processed.
1138 1080 of 2717 files processed.
1138 1090 of 2717 files processed.
1140 1040 of 2717 files processed.
1139 1130 of 2717 files processed.
1137 1100 of 2717 files processed.
1139 1140 of 2717 files processed.
1138 1100 of 2717 files processed.
1137 1110 of 2717 files processed.
1138 1110 of 2717 files processed.
1140 1050 of 2717 files processed.
1139 1150 of 2717 files processed.
1137 1120 of 2717 files processed.
1138 1120 of 2717 files processed.
1140 1060 of 2717 files processed.
1138 1130 of 2717 files processed.
1139 1160 of 2717 files processed.
1137 1130 of 2717 files processed.
1140 1070 of 2717 files processed.
1138 1140 of 2717 files processed.
1140 1080 of 2717 files processed.
1138 1150 of 2717 files processed.
1137 1140 of 2717 files processed.
1140 1090 of 2717 files processed.
1139 1170 of 2717 files processed.
1139 1180 of 2717 files processed.
1140 1100 of 2717 files processed.
1137 1150 of 2717 files processed.
1138 1160 of 2717 files processed.
1139 1190 of 2717 files processed.
1138 1170 of 2717 files processed.
1140 1110 of 27171139 1200  of files processed.2717
 files processed.
1137 1160 of 2717 files processed.
1139 1210 of 2717 files processed.
1137 1170 of 2717 files processed.
1139 1220 of 2717 files processed.
1140 1120 of 2717 files processed.
1140 1130 of 2717 files processed.
1138 1180 of 2717 files processed.
1137 1180 of 2717 files processed.
1139 1230 of 2717 files processed.
1140 1140 of 2717 files processed.
1137 1190 of 2717 files processed.
1138 1190 of 2717 files processed.
1139 1240 of 2717 files processed.
1137 1200 of 2717 files processed.
1138 1200 of 2717 files processed.
1140 1150 of 2717 files processed.
1139 1250 of 2717 files processed.
1137 1210 of 2717 files processed.
1138 1210 of 2717 files processed.
1140 1160 of 2717 files processed.
1137 1220 of 2717 files processed.
1138 1220 of 2717 files processed.
1140 1170 of 2717 files processed.
1140 1180 of 2717 files processed.
1139 1260 of 2717 files processed.
1137 1230 of 2717 files processed.
1140 1190 of 2717 files processed.
1138 1230 of 2717 files processed.
1139 1270 of 2717 files processed.
1137 1240 of 2717 files processed.
1140 1200 of 2717 files processed.
1139 1280 of 2717 files processed.
1138 1240 of 2717 files processed.
1137 1250 of 2717 files processed.
1139 1290 of 2717 files processed.
```

```
1138 1250 of 2717 files processed.
1140 1210 of 2717 files processed.
1137 1260 of 2717 files processed.
1139 1300 of 2717 files processed.
1138 1260 of 2717 files processed.
1139 1310 of 2717 files processed.
1139 1320 of 2717 files processed.
1140 1220 of 2717 files processed.
1137 1270 of 2717 files processed.
1137 1280 of 2717 files processed.
1139 1330 of 2717 files processed.
1138 1270 of 2717 files processed.
1140 1230 of 2717 files processed.
1138 1280 of 2717 files processed.
1137 1290 of 2717 files processed.
1140 1240 of 2717 files processed.
1140 1250 of 2717 files processed.
1137 1300 of 2717 files processed.
1139 1340 of 2717 files processed.
1140 1260 of 2717 files processed.
1137 1310 of 2717 files processed.
1138 1290 of 2717 files processed.
1139 1350 of 2717 files processed.
1139 1360 of 2717 files processed.
1140 1270 of 2717 files processed.
1138 1300 of 2717 files processed.
1137 1320 of 2717 files processed.
1139 1370 of 2717 files processed.
1140 1280 of 2717 files processed.
1137 1330 of 2717 files processed.
1138 1310 of 2717 files processed.
1140 1290 of 2717 files processed.
1137 1340 of 2717 files processed.
1138 1320 of 2717 files processed.
1139 1380 of 2717 files processed.
1140 1300 of 2717 files processed.
1138 1330 of 2717 files processed.
1137 1350 of 2717 files processed.
1138 1340 of 2717 files processed.
1140 1310 of 2717 files processed.
1138 1350 of 2717 files processed.
1139 1390 of 2717 files processed.
1140 1320 of 2717 files processed.
1137 1360 of 2717 files processed.
1138 1360 of 2717 files processed.
1138 1370 of 2717 files processed.
1137 1370 of 2717 files processed.
1139 1400 of 2717 files processed.
1140 1330 of 2717 files processed.
1138 1380 of 2717 files processed.
1139 1410 of 2717 files processed.
1138 1390 of 2717 files processed.
1137 1380 of 2717 files processed.
1140 1340 of 2717 files processed.
1138 1400 of 2717 files processed.
1139 1420 of 2717 files processed.
1140 1350 of 2717 files processed.
1139 1430 of 2717 files processed.
1137 1390 of 2717 files processed.
1140 1360 of 2717 files processed.
1139 1440 of 2717 files processed.
1139 1450 of 2717 files processed.
1138 1410 of 2717 files processed.
1137 1400 of 2717 files processed.
1140 1370 of 2717 files processed.
1139 1460 of 2717 files processed.
1140 1380 of 2717 files processed.
1137 1410 1140 1139of1390 of  1470  of 2717 27172717 files processed. files processed.files processed.


1138 1420 of 2717 files processed.
1137 1420 of 2717 files processed.
1139 1480 of 2717 files processed.
1140 1400 of 2717 files processed.
1138 1430 of 2717 files processed.
1137 1430 of 2717 files processed.
1139 1490 of 2717 files processed.
1140 1410 of 2717 files processed.
1138 1440 of 2717 files processed.
1139 1500 of 2717 files processed.
1140 1420 of 2717 files processed.
1139 1510 of 2717 files processed.
1137 1440 of 2717 files processed.
1140 1430 of 2717 files processed.
1138 1450 of 2717 files processed.
1137 1450 of 2717 files processed.
1140 1440 of 2717 files processed.
1140 1450 of 2717 files processed.
1137 1460 of 2717 files processed.
1138 1460 of 2717 files processed.
1137 1470 of 2717 files processed.
1139 1520 of 2717 files processed.
1137 1480 of 2717 files processed.
1139 1530 of 2717 files processed.
1137 1490  of2717 files processed.
1138 1470 of 2717 files processed.
1140 1460 of 2717 files processed.
1137 1500 of 2717 files processed.
1138 1480 of 2717 files processed.
1139 1540 of 2717 files processed.
1140 1470 of 2717 files processed.
1137 1510 of 2717 files processed.
1139 1550 of 2717 files processed.
1138 1490 of 2717 files processed.
1137 1520 of 2717 files processed.
1139 1560 of 2717 files processed.
1140 1480 of 2717 files processed.
1138 1500 of 2717 files processed.
1138 1510 of 2717 files processed.
1137 1530 of 2717 files processed.
1139 1570 of 2717 files processed.
1138 1520 of 2717 files processed.
1140 1490 of 2717 files processed.
1138 1530 of 2717 files processed.
1140 1500 of 2717 files processed.
1139 1580 of 2717 files processed.
1139 1590 of 2717 files processed.
1138 1540 of 2717 files processed.
1140 1510 of 2717 files processed.
1137 1540 of 2717 files processed.
1138 1550 of 2717 files processed.
1139 1600 of 2717 files processed.
1140 1520 of 2717 files processed.
1138 1560 of 2717 files processed.
1139 1610 of 2717 files processed.
1137 1550 of 2717 files processed.
1138 1570 of 2717 files processed.
1139 1620 of 2717 files processed.
1137 1560 of 2717 files processed.
1140 1530 of 2717 files processed.
1138 1580 of 2717 files processed.
1137 1570 of 2717 files processed.
1138 1590 of 27171139 files processed.
 1630 of 2717 files processed.
1140 1540 of 2717 files processed.
1140 1550 of 2717 files processed.
1137 1580 of 2717 files processed.
1139 1640 of 2717 files processed.
1138 1600 of 2717 files processed.
1137 1590 of 2717 files processed.
1140 1560 of 2717 files processed.
1139 1650 of 2717 files processed.
1139 1660 of 2717 files processed.
1137 1600 of 2717 files processed.
1140 1570 of 2717 files processed.
1138 1610 of 2717 files processed.
1138 1620 of 2717 files processed.
1138 1630 of 2717 files processed.
1139 1670 of 2717 files processed.
1137 1610 of 2717 files processed.
1139 1680 of 2717 files processed.
1138 1640 of 2717 files processed.
1140 1580 of 2717 files processed.
1137 1620 of 2717 files processed.
1140 1590 of 2717 files processed.
1138 1650 of 2717 files processed.
1139 1690 of 2717 files processed.
1137 1630 of 2717 files processed.
1140 1600 of 2717 files processed.
1139 1700 of 2717 files processed.
1138 1660 of 2717 files processed.
1140 1610 of 2717 files processed.
1137 1640 of 2717 files processed.
1138 1670 of 2717 files processed.
1138 1680 of 2717 files processed.
1139 1710 of 2717 files processed.
1140 1620 of 2717 files processed.
1137 1650 of 2717 files processed.
1140 1630 of 2717 files processed.
1139 1720 of 2717 files processed.
1138 1690 of 2717 files processed.
1140 1640 of 2717 files processed.
11381137  17001660 of of 2717  2717files processed. files processed.

1139 1730 of 2717 files processed.
```

```
1138 1710 of 2717 files processed.
1139 1740 of 2717 files processed.
1138 1720 of 2717 files processed.
1137 1670 of 2717 files processed.
1139 1750 of 2717 files processed.
1140 1650 of 2717 files processed.
1138 1730 of 2717 files processed.
1140 1660 of 2717 files processed.
1139 1760 of 2717 files processed.
1137 1680 of 2717 files processed.
1140 1670 of 2717 files processed.
1139 1770 of 2717 files processed.
1140 1680 of 2717 files processed.
1137 1690 of 2717 files processed.
1138 1740 of 2717 files processed.
1139 1780 of 2717 files processed.
1137 1700 of 2717 files processed.
1140 1690 of 2717 files processed.
1140 1700 of 2717 files processed.
1137 1710 of 2717 files processed.
1139 1790 of 2717 files processed.
1140 1710 of 2717 files processed.
1138 1750 of 2717 files processed.
1137 1720 of 2717 files processed.
1140 1720 of 2717 files processed.
1140 1730 of 2717 files processed.
1139 1800 of 2717 files processed.
1138 1760 of 2717 files processed.
1139 1810 of 2717 files processed.
1140 1740 of 2717 files processed.
1137 1730 of 2717 files processed.
1138 1770 of 2717 files processed.
1138 1780 of 2717 files processed.
1139 1820 of 2717 files processed.
1137 1740 of 2717 files processed.
1140 1750 of 2717 files processed.
1138 1790 of 2717 files processed.
1139 1830 of 2717 files processed.
1138 1800 of 2717 files processed.
1140 1760 of 2717 files processed.
1140 1770 of 2717 files processed.
1139 1840 of 2717 files processed.
1137 1750 of 2717 files processed.
1140 1780 of 2717 files processed.
1137 1760 of 2717 files processed.
1138 1810 of 2717 files processed.
1137 1770 of 2717 files processed.
1137 1780 of 2717 files processed.
1138 1820 of 2717 files processed.
1140 1790 of 2717 files processed.
1138 1830 of 2717 files processed.
1137 1790 of 2717 files processed.
1140 1800 of 2717 files processed.
1138 1840 of 2717 files processed.
1139 1850 of 2717 files processed.
1140 1810 of 2717 files processed.
1138 1850 of 2717 files processed.
1140 1820 of 2717 files processed.
1137 1800 of 2717 files processed.
1140 1830 of 2717 files processed.
1140 1840 of 2717 files processed.
1139 1860 of 2717 files processed.
1137 1810 of 2717 files processed.
1138 1860 of 2717 files processed.
1137 1820 of 2717 files processed.
1140 1850 of 2717 files processed.
1137 1830 of 2717 files processed.
1140 1860 of 2717 files processed.
1138 1870 of 2717 files processed.
1137 1840 1139of  18702717  files processed.
of 2717 files processed.
1137 1850 of 2717 files processed.
1140 1870 of 2717 files processed.
1139 1880 of 2717 files processed.
1138 1880 of 2717 files processed.
1140 1880 of 2717 files processed.
1138 1890 of 2717 files processed.
1137 1860 of 2717 files processed.
1138 1900 of 2717 files processed.
1139 1890 of 2717 files processed.
1137 1870 of 2717 files processed.
1139 1900 of 2717 files processed.
1137 1880 of 2717 files processed.
1138 1910 of 2717 files processed.
1140 1890 of 2717 files processed.
1139 1910 of 2717 files processed.
1140 1900 of 2717 files processed.
1137 1890 of 2717 files processed.
1137 1900 of 2717 files processed.
1140 1910 of 2717 files processed.
1137 1910 of 2717 files processed.
1138 1920 of 2717 files processed.
1139 1920 of 2717 files processed.
1137 1920 of 2717 files processed.
1140 1920 of 2717 files processed.
1138 1930 of 2717 1139files processed.
1930 of 2717 files processed.
1137 1930 of 2717 files processed.
1137 1940 of 2717 files processed.
1139 1940 of 2717 files processed.
1140 1930 of 2717 files processed.
1138 1940 of 2717 files processed.
1139 1950 of 2717 files processed.
1137 1950 of 2717 files processed.
1138 1950 of 2717 files processed.
1140 1940 of 2717 files processed.
1138 1960 of 2717 files processed.
1139 1960 of 2717 files processed.
1138 1970 of 2717 files processed.
1139 1970 of 2717 files processed.
1137 1960 of 2717 files processed.
1138 1980 of 2717 files processed.
1140 1950 of 2717 files processed.
1139 1980 of 2717 files processed.
1138 1990 of 2717 files processed.
1137 1970 of 2717 files processed.
1140 1960 of 2717 files processed.
1139 1990 of 2717 files processed.
1137 1980 of 2717 files processed.
1140 1970 of 2717 files processed.
1137 1990 of 2717 files processed.
1139 2000 of 2717 files processed.
1140 1980 of 2717 files processed.
1138 2000 of 2717 files processed.
1140 1990 of 2717 files processed.
1137 2000 of 2717 files processed.
1139 2010 of 2717 files processed.
1137 2010 of 2717 files processed.
1139 2020 of 2717 files processed.
1138 2010 of 2717 files processed.
1139 2030 of 2717 files processed.
1138 2020 of 2717 files processed.
1140 2000 of 2717 files processed.
1138 2030 of 2717 files processed.
1137 2020 of 2717 files processed.
1139 2040 of 2717 files processed.
1140 2010 of 2717 files processed.
1137 2030 of 2717 files processed.
1140 2020 of 2717 files processed.
1138 2040 of 2717 files processed.
1139 2050 of 2717 files processed.
1137 2040 of 2717 files processed.
1138 2050 of 2717 files processed.
1140 2030 of 2717 files processed.
1139 2060 of 2717 files processed.
1137 2050 of 2717 files processed.
1140 2040 of 2717 files processed.
1137 2060 of 2717 files processed.1139
 2070 of 2717 files processed.
1139 2080 of 2717 files processed.
1138 2060 of 2717 files processed.
1137 2070 of 2717 files processed.
1138 2070 of 2717 files processed.
1137 2080 of 2717 files processed.
1140 2050 of 2717 files processed.
1139 2090 of 2717 files processed.
1140 2060 of 2717 files processed.
1137 2090 of 2717 files processed.
1139 2100 of 2717 files processed.
1140 2070 of 2717 files processed.
1138 2080 of 2717 files processed.
1139 2110 of 2717 files processed.
1140 2080 of 2717 files processed.
1137 2100 of 2717 files processed.
1139 2120 of 2717 files processed.
1138 2090 of 2717 files processed.
1140 2090 of 2717 files processed.
1137 2110 of 2717 files processed.
1138 2100 of 2717 files processed.
1139 2130 of 2717 files processed.
1140 2100 of 2717 files processed.
1137 2120 of 2717 files processed.
1138 2110 of 2717 files processed.
1139 2140 of 2717 files processed.
1140 2110 of 2717 files processed.
```

```
1137 2130 of 2717 files processed.
1138 2120 of 2717 files processed.
1139 2150 of 2717 files processed.
1139 2160 of 2717 files processed.
1137 2140 of 2717 files processed.
1138 2130 of 2717 files processed.
1140 2120 of 2717 files processed.
1139 2170 of 2717 files processed.
1137 2150 of 2717 files processed.
1138 2140 of 2717 files processed.
1140 2130 of 2717 files processed.
1139 2180 of 2717 files processed.
1137 2160 of 2717 files processed.
1137 2170 of 2717 files processed.
1138 2150 of 2717 files processed.
1140 2140 of 2717 files processed.
1139 2190 of 2717 files processed.
1137 2180 of 2717 files processed.
1138 2160 of 2717 files processed.
1137 2190 of 2717 files processed.
1139 2200 of 2717 files processed.
1140 2150 of 2717 files processed.
1138 2170 of 2717 files processed.
1137 2200 of 2717 files processed.
1139 2210 of 2717 files processed.
1140 2160 of 2717 files processed.
1139 2220 of 2717 files processed.
1140 2170 of 2717 files processed.
1137 2210 of 2717 files processed.
1138 2180 of 2717 files processed.
1140 2180 of 2717 files processed.
1139 2230 of 2717 files processed.
1138 2190 of 2717 files processed.
1140 2190 of 2717 files processed.
1140 2200 of 2717 files processed.
1137 2220 of 2717 files processed.
1139 2240 of 2717 files processed.
1138 2200 of 2717 files processed.
1139 2250 of 2717 files processed.
1140 2210 of 2717 files processed.
1140 2220 of 2717 files processed.
1139 2260 of 2717 files processed.
1140 2230 of 2717 files processed.
1137 2230 of 2717 files processed.
1139 2270 of 2717 files processed.
1140 2240 of 2717 files processed.
1138 2210 of 2717 files processed.
1139 2280 of 2717 files processed.
1140 2250 of 2717 files processed.
1137 2240 of 2717 files processed.
1140 2260 of 2717 files processed.
1139 2290 of 2717 files processed.
1138 2220 of 2717 files processed.
1140 2270 of 2717 files processed.
1137 2250 of 2717 files processed.
1139 2300 of 2717 files processed.
1138 2230 of 2717 files processed.
1137 2260 of 2717 files processed.
1140 2280 of 2717 files processed.
1139 2310 of 2717 files processed.
1137 2270 of 2717 files processed.
1138 2240 of 2717 files processed.
1140 2290 of 2717 files processed.
1137 2280 of 2717 files processed.
1139 2320 of 2717 files processed.
1140 2300 of 2717 files processed.
1138 2250 of 2717 files processed.
1140 2310 of 2717 files processed.
1137 2290 of 2717 files processed.
1138 2260 of 2717 files processed.
1140 2320 of 2717 files processed.
1139 2330 of 2717 files processed.
1138 2270 of 2717 files processed.
1140 2330 of 2717 files processed.
1138 2280 of 2717 files processed.
1139 2340 of 2717 files processed.
1140 2340 of 2717 files processed.
1137 2300 of 2717 files processed.
1137 2310 of 2717 files processed.
1138 2290 of 2717 files processed.
1139 2350 of 2717 files processed.
1140 2350 of 2717 files processed.
1138 2300 of 2717 files processed.
1137 2320 of 2717 files processed.
1138 2310 of 2717 files processed.
1139 2360 of 2717 files processed.
1140 2360 of 2717 files processed.
1138 2320 of 2717 files processed.
1140 2370 of 2717 files processed.
1139 2370 of 2717 files processed.
1139 2380 of 2717 files processed.
1137 2330 of 2717 files processed.
1140 2380 of 2717 files processed.
1138 2330 of 2717 files processed.
1137 2340 of 2717 files processed.
1138 2340 of 2717 files processed.
1139 2390 of 2717 files processed.
1140 2390 of 2717 files processed.
1137 2350 of 2717 files processed.
1138 2350 of 2717 files processed.
1139 2400 of 2717 files processed.
1140 2400 of 2717 files processed.
1138 2360 of 1137 27172360  offiles processed.
 2717 files processed.
1139 2410 of 2717 files processed.
1138 2370 of 2717 files processed.
1137 2370 of 11392717  files processed.2420
 of 2717 files processed.
1137 2380 of 2717 files processed.
1138 2380 of 2717 files processed.
1139 2430 of 2717 files processed.
1140 2410 of 2717 files processed.
1139 2440 of 2717 files processed.
1137 2390 of 2717 files processed.
1139 2450 of 2717 files processed.
1140 2420 of 2717 files processed.
1138 2390 of 2717 files processed.
1137 2400 of 2717 files processed.
1139 2460 of 2717 files processed.
1138 2400 of 2717 files processed.
1140 2430 of 2717 files processed.
1137 2410 of 2717 files processed.
1137 2420 of 2717 files processed.
1139 2470 of 2717 files processed.
1137 2430 of 2717 files processed.
1140 2440 of 2717 files processed.
1138 2410 of 2717 files processed.
1137 2440 of 2717 files processed.
1140 2450 of 2717 files processed.
1139 2480 of 2717 files processed.
1138 2420 of 2717 files processed.
1137 2450 of 2717 files processed.
1139 2490 of 2717 files processed.
1140 2460 of 2717 files processed.
1139 2500 of 2717 files processed.
1137 2460 of 2717 files processed.
1140 2470 of 2717 files processed.
1139 2510 of 2717 files processed.
1138 2430 of 2717 files processed.
1138 2440 of 2717 files processed.
1139 2520 of 2717 files processed.
1140 2480 of 2717 files processed.
1138 2450 of 2717 files processed.
1137 2470 of 2717 files processed.
1140 2490 of 2717 files processed.
1137 2480 of 2717 files processed.
1139 2530 of 2717 files processed.
1140 2500 of 2717 files processed.
1139 2540 of 2717 files processed.
1140 2510 of 2717 files processed.
1137 2490 of 2717 files processed.
1138 2460 of 2717 files processed.
1139 2550 of 2717 files processed.
1140 2520 of 2717 files processed.
1138 2470 of 2717 files processed.
1139 2560 of 2717 files processed.
1140 2530 of 2717 files processed.
1139 2570 of 2717 files processed.
1138 2480 of 2717 files processed.
1137 2500 of 2717 files processed.
1140 2540 of 2717 files processed.
1139 2580 of 2717 files processed.
1138 2490 of 2717 files processed.
1140 2550 of 2717 files processed.
1139 2590 of 2717 files processed.
1138 2500 of 2717 files processed.
1140 2560 of 2717 files processed.
1137 2510 of 2717 files processed.
1138 2510 of 2717 files processed.
1138 2520 of 2717 files processed.
1140 2570 of 2717 files processed.
1139 2600 of 2717 files processed.
1138 2530 of 2717 files processed.
1137 2520 of 2717 files processed.
1140 2580 of 2717 files processed.
```

```
1137 2530 of 2717 files processed.
1139 2610 of 2717 files processed.
1138 2540 of 2717 files processed.
1137 2540 of 2717 files processed.
1138 2550 of1140  27172590  offiles processed.
 2717 files processed.
1139 2620 of 2717 files processed.
1137 2550 of 2717 files processed.
1138 2560 of 2717 files processed.
1137 2560 of 2717 files processed.
1139 2630 of 2717 files processed.
1138 2570 of 2717 files processed.
1137 2570 of 2717 files processed.
1140 2600 of 2717 files processed.
1138 2580 of 2717 files processed.
1137 2580 of 2717 files processed.
1139 2640 of 2717 files processed.
1140 2610 of 2717 files processed.
1138 2590 of 2717 files processed.
1139 2650 of 2717 files processed.
1140 2620 of 2717 files processed.
1139 2660 of 2717 files processed.
1137 2590 of 2717 files processed.
1138 2600 of 2717 files processed.
1140 2630 of 2717 files processed.
1139 2670 of 2717 files processed.
1140 2640 of 2717 files processed.
1139 2680 of 2717 files processed.
1138 2610 of 2717 files processed.
1137 2600 of 2717 files processed.
1139 2690 of 2717 files processed.
1138 2620 of 2717 files processed.
1140 2650 of 2717 files processed.
1138 2630 of 2717 files processed.
1140 2660 of 2717 files processed.
1139 2700 of 2717 files processed.
1138 2640 of 2717 files processed.
1137 2610 of 2717 files processed.
1139 2710 of 2717 files processed.
1137 2620 of 2717 files processed.
1137 2630 of 2717 files processed.
1138 2650 of 2717 files processed.
1140 2670 of 2717 files processed.
1137 2640 of 2717 files processed.
1138 2660 of 2717 files processed.
1140 2680 of 2717 files processed.
1137 2650 of 2717 files processed.
1140 2690 of 2717 files processed.
1138 2670 of 2717 files processed.
1137 2660 of 2717 files processed.
1137 2670 of 2717 files processed.
1138 2680 of 2717 files processed.
1140 2700 of 2717 files processed.
1137 2680 of 2717 files processed.
1138 2690 of 2717 files processed.
1137 2690 of 2717 files processed.
1140 2710 of 2717 files processed.
1138 2700 of 2717 files processed.
1138 2710 of 2717 files processed.
1137 2700 of 2717 files processed.
1137 2710 of 2717 files processed.
Elapsed time: 0.00 hours.
```

Merging the image dataset

In [6]:
```python
file1 = pd.read_csv('1137-image-features-asm.csv')
file2 = pd.read_csv('1138-image-features-asm.csv')
file3 = pd.read_csv('1139-image-features-asm.csv')
file4 = pd.read_csv('1140-image-features-asm.csv')

image_data = np.concatenate([file1,file2,file3,file4],axis=0)
```

In [7]:
```python
file1.columns
```

Out[7]:
```
Index(['filename', 'ASM_0', 'ASM_1', 'ASM_2', 'ASM_3', 'ASM_4', 'ASM_5',
       'ASM_6', 'ASM_7', 'ASM_8',
       ...
       'ASM_990', 'ASM_991', 'ASM_992', 'ASM_993', 'ASM_994', 'ASM_995',
       'ASM_996', 'ASM_997', 'ASM_998', 'ASM_999'],
      dtype='object', length=1001)
```

In [8]:
```python
labels = pd.read_csv('trainLabels.csv')
```

In [9]:
```python
labels.head(5)
```

Out[9]:

	Id	Class
0	01kcPWA9K2BOxQeS5Rju	1
1	04EjIdbPV5e1XroFOpiN	1
2	05EeG39MTRrI6VY21DPd	1
3	05rJTUWYAKNegBk2wE8X	1
4	0AnoOZDNbPXIr2MRBSCJ	1

In [10]:
```python
image_data = pd.DataFrame(data=image_data,columns=file1.columns)
```

In [11]:
```python
image_data.head()
```

Out[11]:

	filename	ASM_0	ASM_1	ASM_2	ASM_3	ASM_4	ASM_5	ASM_6	ASM_7	ASM_8	...	ASM_990	ASM_991	ASM_992	ASM_993	ASM_994	ASM_995	ASM_996	ASM_997	ASM_998	ASM_999
0	JKbYXxOv2Wa8eMNlFq04	46	116	101	120	116	58	48	48	52	...	10	46	116	101	120	116	58	48	48	52
1	1ZYtDkrdm9yVLpeU5cSJ	72	69	65	68	69	82	58	48	48	...	59	32	70	111	114	109	97	116	9	32
2	Hl3iXz1IhS69DCANfRVT	72	69	65	68	69	82	58	48	48	...	116	101	120	116	58	48	48	52	48	49
3	HSyZm4zlbXMT5R37F1wK	46	116	101	120	116	58	48	48	52	...	10	46	116	101	120	116	58	48	48	52
4	ESoHvWkbZ4JgwzlFK6uA	72	69	65	68	69	82	58	48	48	...	59	32	70	111	114	109	97	116	9	32

5 rows × 1001 columns

In [13]:
```python
image_data_sorted = image_data.sort_values(by='filename')
labels_sorted = labels.sort_values(by='Id')

## removing the column file name from the dataset

X_ = image_data.iloc[:,1:]
Y_ = labels.iloc[:,1:]
```

In [14]:
```python
print('Shape of the train data:',X_.shape,' shape of labels:',Y_.shape)
```

```
Shape of the train data: (10868, 1000)  shape of labels: (10868, 1)
```

2.Feature Selection using Chi2- selecting top 50%

refer: [https://github.com/dchad/malware-detection/blob/master/mmcc/feature-reduction.ipynb (https://github.com/dchad/malware-detection/blob/master/mmcc/feature-reduction.ipynb)](https://github.com/dchad/malware-detection/blob/master/mmcc/feature-reduction.ipynb)

In [15]:
```python
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn.feature_selection import chi2
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [17]:
```python
# find the top 50 percent variance features, from 1006 -> 503 features
fsp = SelectPercentile(chi2, 50)
X_new_50 = fsp.fit_transform(X_,Y_)
X_new_50.shape
```

Out[17]:
```
(10868, 500)
```

In [18]:
```
selected_names = fsp.get_support(indices=True)
selected_names = selected_names + 1
selected_names
```

Out[18]:
```
array([  15,   21,   22,   29,   30,   32,   33,   34,   35,   41,   42,
         43,   44,   48,   50,  125,  126,  135,  136,  138,  139,  140,
        141,  142,  143,  144,  145,  146,  147,  148,  151,  155,  156,
        157,  158,  160,  161,  162,  163,  164,  165,  167,  169,  173,
        174,  179,  186,  188,  190,  198,  201,  202,  205,  215,  216,
        220,  221,  222,  223,  224,  226,  227,  236,  240,  242,  243,
        244,  245,  246,  247,  248,  249,  252,  253,  260,  261,  262,
        263,  264,  265,  266,  267,  268,  271,  272,  282,  287,  291,
        292,  293,  294,  295,  296,  297,  307,  310,  311,  312,  313,
        314,  315,  317,  318,  323,  326,  327,  328,  330,  334,  337,
        338,  339,  340,  341,  343,  344,  345,  346,  349,  351,  352,
        353,  354,  356,  357,  358,  359,  366,  370,  371,  372,  373,
        374,  375,  376,  378,  379,  380,  381,  384,  399,  400,  401,
        405,  408,  409,  410,  412,  413,  414,  415,  422,  423,  424,
        425,  426,  427,  428,  436,  437,  439,  440,  441,  446,  447,
        448,  449,  450,  451,  452,  453,  457,  460,  461,  464,  465,
        466,  467,  477,  478,  479,  480,  481,  482,  538,  539,  555,
        556,  557,  558,  559,  560,  561,  563,  564,  567,  568,  572,
        573,  580,  581,  582,  583,  584,  585,  586,  587,  588,  589,
        590,  597,  598,  600,  601,  602,  603,  606,  607,  613,  614,
        615,  616,  617,  618,  619,  620,  621,  622,  623,  624,  627,
        628,  629,  630,  631,  632,  633,  634,  635,  636,  637,  638,
        641,  642,  643,  644,  645,  646,  647,  648,  649,  650,  651,
        652,  653,  654,  655,  656,  657,  658,  659,  660,  663,  664,
        665,  667,  668,  669,  670,  671,  672,  673,  674,  675,  676,
        677,  678,  679,  680,  681,  682,  683,  684,  685,  686,  687,
        688,  691,  692,  693,  694,  695,  696,  697,  701,  702,  705,
        706,  707,  708,  709,  710,  711,  712,  713,  714,  715,  717,
        718,  719,  720,  722,  723,  724,  725,  726,  727,  728,  729,
        730,  731,  732,  733,  734,  735,  736,  737,  738,  739,  744,
        746,  747,  748,  749,  751,  752,  753,  754,  755,  756,  758,
        759,  760,  761,  762,  763,  764,  765,  774,  775,  777,  778,
        779,  780,  781,  782,  783,  784,  785,  786,  788,  790,  791,
        792,  793,  794,  795,  797,  798,  799,  800,  801,  802,  813,
        814,  815,  816,  818,  819,  828,  829,  830,  831,  832,  833,
        834,  835,  836,  837,  838,  841,  843,  844,  845,  847,  848,
        849,  850,  851,  852,  853,  854,  855,  856,  857,  865,  866,
        867,  868,  869,  870,  873,  874,  875,  876,  877,  878,  879,
        882,  885,  887,  888,  889,  890,  892,  893,  894,  895,  896,
        898,  900,  901,  902,  903,  904,  905,  906,  916,  917,  918,
        919,  920,  923,  924,  930,  931,  932,  934,  935,  936,  937,
        938,  943,  947,  948,  949,  950,  951,  952,  953,  954,  955,
        956,  957,  958,  959,  960,  961,  962,  963,  964,  965,  966,
        967,  968,  969,  970,  973,  974,  975,  976,  977,  978,  979,
        980,  981,  982,  983,  989,  990,  991,  992,  993,  994,  995,
        996,  997,  998,  999, 1000])
```

In [21]:
```
data_trimmed = image_data_sorted.iloc[:,selected_names]
data_fnames = pd.DataFrame(image_data_sorted['filename'])
data_50 = data_fnames.join(data_trimmed)
data_50.head()
```

Out[21]:

	filename	ASM_14	ASM_20	ASM_21	ASM_28	ASM_29	ASM_31	ASM_32	ASM_33	ASM_34	...	ASM_990	ASM_991	ASM_992	ASM_993	ASM_994	ASM_995	ASM_996	ASM_997	ASM_998	ASM_999
3677	01IsoiSMh5gxyDYTl4CB	9	32	32	13	10	116	101	120	116	...	10	46	116	101	120	116	58	48	48	52
5749	01SuzwMJEIXsK7A8dQbl	48	9	9	68	69	58	48	48	52	...	116	101	120	116	58	48	48	52	48	49
3852	01azqd4InC7m9JpocGv5	48	9	9	68	69	58	48	48	52	...	116	101	120	116	58	48	48	52	48	49
9748	01jsnpXSAlgw6aPeDxrU	48	9	9	68	69	58	48	48	52	...	116	101	120	116	58	48	48	52	48	49
6133	01kcPWA9K2BOxQeS5Rju	48	9	9	68	69	58	49	48	48	...	71	77	69	78	84	32	72	69	65	68

5 rows × 501 columns

In [23]:
```
print('Shape of the data after data reduction with 50% variance features:',data_50.shape)
```

```
Shape of the data after data reduction with 50% variance features: (10868, 501)
```

In [34]:
```
### save the file after performing feature selection
data_50.rename(columns={'filename':'ID'},inplace=True)
data_50.to_csv('asm_50%_variance_data.csv')
```

3. Implemation of different Feature Combination with ASM extracted image

3.1 Asm-uni grams + ASM-Extracted image features

In [127]:
```
labels =pd.read_csv('trainLabels.csv')
```

In [128]:
```
dfasm=pd.read_csv("asmoutputfile.csv")
labels.columns = ['ID', 'Class']
result_asm = pd.merge(dfasm, labels,on='ID', how='left')
result_asm.head()
```

Out[128]:

	ID	HEADER:	.text:	.Pav:	.idata:	.data:	.bss:	.rdata:	.edata:	.rsrc:	...	edx	esi	eax	ebx	ecx	edi	ebp	esp	eip	Class
0	01kcPWA9K2BOxQeS5Rju	19	744	0	127	57	0	323	0	3	...	18	66	15	43	83	0	17	48	29	1
1	1E93CpP60RHFNiT5Qfvn	17	838	0	103	49	0	0	0	3	...	18	29	48	82	12	0	14	0	20	1
2	3ekVow2ajZHbTnBcsDfX	17	427	0	50	43	0	145	0	3	...	13	42	10	67	14	0	11	0	9	1
3	3X2nY7iQaPBIWDrAZqJe	17	227	0	43	19	0	0	0	3	...	6	8	14	7	2	0	8	0	6	1
4	46OZzdsSKDCFV8h7XWxf	17	402	0	59	170	0	0	0	3	...	12	9	18	29	5	0	11	0	11	1

5 rows × 53 columns

In [129]:
```
data_50 = pd.read_csv('asm_50%_variance_data.csv')
```

In [130]:
```
data_50.drop(columns=['Unnamed: 0'],inplace=True)
data_50.columns
```

Out[130]:
```
Index(['ID', 'ASM_14', 'ASM_20', 'ASM_21', 'ASM_28', 'ASM_29', 'ASM_31',
       'ASM_32', 'ASM_33', 'ASM_34',
       ...
       'ASM_990', 'ASM_991', 'ASM_992', 'ASM_993', 'ASM_994', 'ASM_995',
       'ASM_996', 'ASM_997', 'ASM_998', 'ASM_999'],
      dtype='object', length=501)
```

In [131]:
```
X_merged = pd.merge(result_asm, data_50, on='ID', how='inner')
X_merged.head()
```

Out[131]:

	ID	HEADER:	.text:	.Pav:	.idata:	.data:	.bss:	.rdata:	.edata:	.rsrc:	...	ASM_990	ASM_991	ASM_992	ASM_993	ASM_994	ASM_995	ASM_996	ASM_997	ASM_998	ASM_999
0	01kcPWA9K2BOxQeS5Rju	19	744	0	127	57	0	323	0	3	...	71	77	69	78	84	32	72	69	65	68
1	1E93CpP60RHFNiT5Qfvn	17	838	0	103	49	0	0	0	3	...	32	32	32	32	58	9	80	111	114	116
2	3ekVow2ajZHbTnBcsDfX	17	427	0	50	43	0	145	0	3	...	59	32	70	111	114	109	97	116	9	32
3	3X2nY7iQaPBIWDrAZqJe	17	227	0	43	19	0	0	0	3	...	32	32	32	32	58	9	80	111	114	116
4	46OZzdsSKDCFV8h7XWxf	17	402	0	59	170	0	0	0	3	...	32	32	32	32	58	9	80	111	114	116

5 rows × 553 columns

In [132]:
```
Y_merged = X_merged['Class']
X_merged = X_merged.drop(['ID','rtn','.BSS:','.CODE','Class'], axis=1)
```

In [133]:
```
### train test split
X_train, X_test, y_train, y_test = train_test_split(X_merged, Y_merged,stratify=Y_merged,test_size=0.20)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,stratify=y_train,test_size=0.20)
```

In [134]:
```
X_train = X_train.apply(pd.to_numeric, errors='coerce')
X_test = X_test.apply(pd.to_numeric, errors='coerce')
X_cv = X_cv.apply(pd.to_numeric, errors='coerce')
```

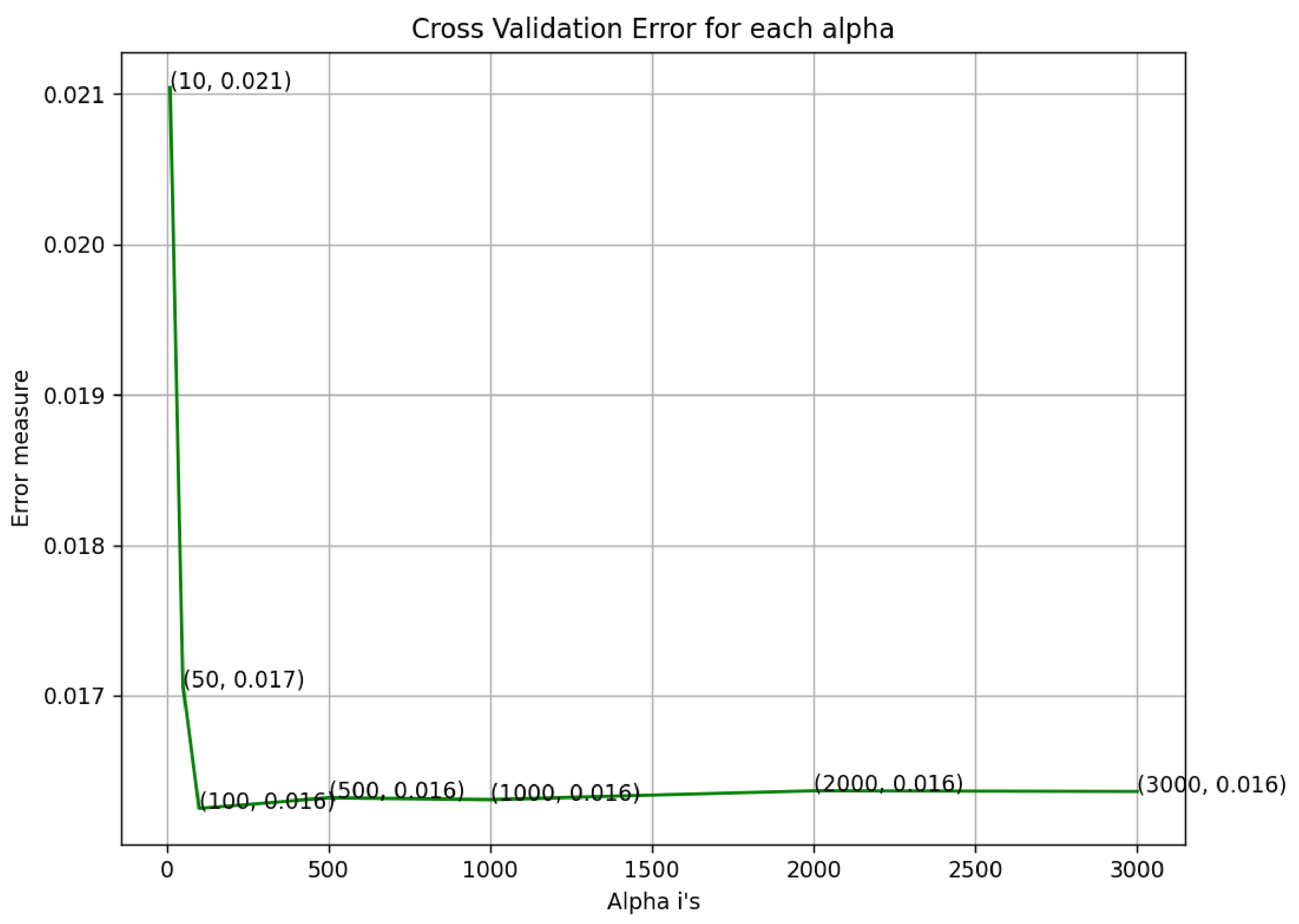
- since random forest and XGB performed better on bytes/asm unigrams we will use both

3.1.1 Random Forest Classifier with Asm-uni grams + ASM-Extracted image features

In [52]:

```
# --------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False,
# class_weight=None)

# Some of methods of RandomForestClassifier()
# fit(X, y, [sample_weight])    Fit the SVM model according to the given training data.
# predict(X)    Perform classification on samples in X.
# predict_proba (X) Perform classification on samples in X.

# some of attributes of  RandomForestClassifier()
# feature_importances_ : array of shape = [n_features]
# The feature importances (the higher, the more important the feature).

# --------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/random-forest-and-their-construction-2/
# --------------------------------

alpha=[10,50,100,500,1000,2000,3000]
cv_log_error_array=[]
train_log_error_array=[]
for i in alpha:
    r_cfl=RandomForestClassifier(n_estimators=i,random_state=42,n_jobs=-1)
    r_cfl.fit(X_train,y_train)
    sig_clf = CalibratedClassifierCV(r_cfl, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_cv)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=r_cfl.classes_, eps=1e-15))
    print('Parameter tuning for alpha:',i,' Loss:',log_loss(y_cv, predict_y, labels=r_cfl.classes_, eps=1e-15))
for i in range(len(cv_log_error_array)):
    print ('log_loss for c = ',alpha[i],'is',cv_log_error_array[i])


best_alpha = np.argmin(cv_log_error_array)

fig, ax = plt.subplots()
ax.plot(alpha, cv_log_error_array,c='g')
for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],cv_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()


r_cfl=RandomForestClassifier(n_estimators=alpha[best_alpha],random_state=42,n_jobs=-1)
r_cfl.fit(X_train,y_train)
sig_clf = CalibratedClassifierCV(r_cfl, 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))
predict_y = sig_clf.predict_proba(X_cv)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y))
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, predict_y))
```

```
Parameter tuning for alpha: 10  Loss: 0.021040358325913156
Parameter tuning for alpha: 50  Loss: 0.01706201515911518
Parameter tuning for alpha: 100  Loss: 0.01624964757759834
Parameter tuning for alpha: 500  Loss: 0.016321154903590304
Parameter tuning for alpha: 1000  Loss: 0.01630844779802613
Parameter tuning for alpha: 2000  Loss: 0.016367377506599643
Parameter tuning for alpha: 3000  Loss: 0.016362740697231582
log_loss for c =  10 is 0.021040358325913156
log_loss for c =  50 is 0.01706201515911518
log_loss for c =  100 is 0.01624964757759834
log_loss for c =  500 is 0.016321154903590304
log_loss for c =  1000 is 0.01630844779802613
log_loss for c =  2000 is 0.016367377506599643
log_loss for c =  3000 is 0.016362740697231582

<IPython.core.display.Javascript object>
```

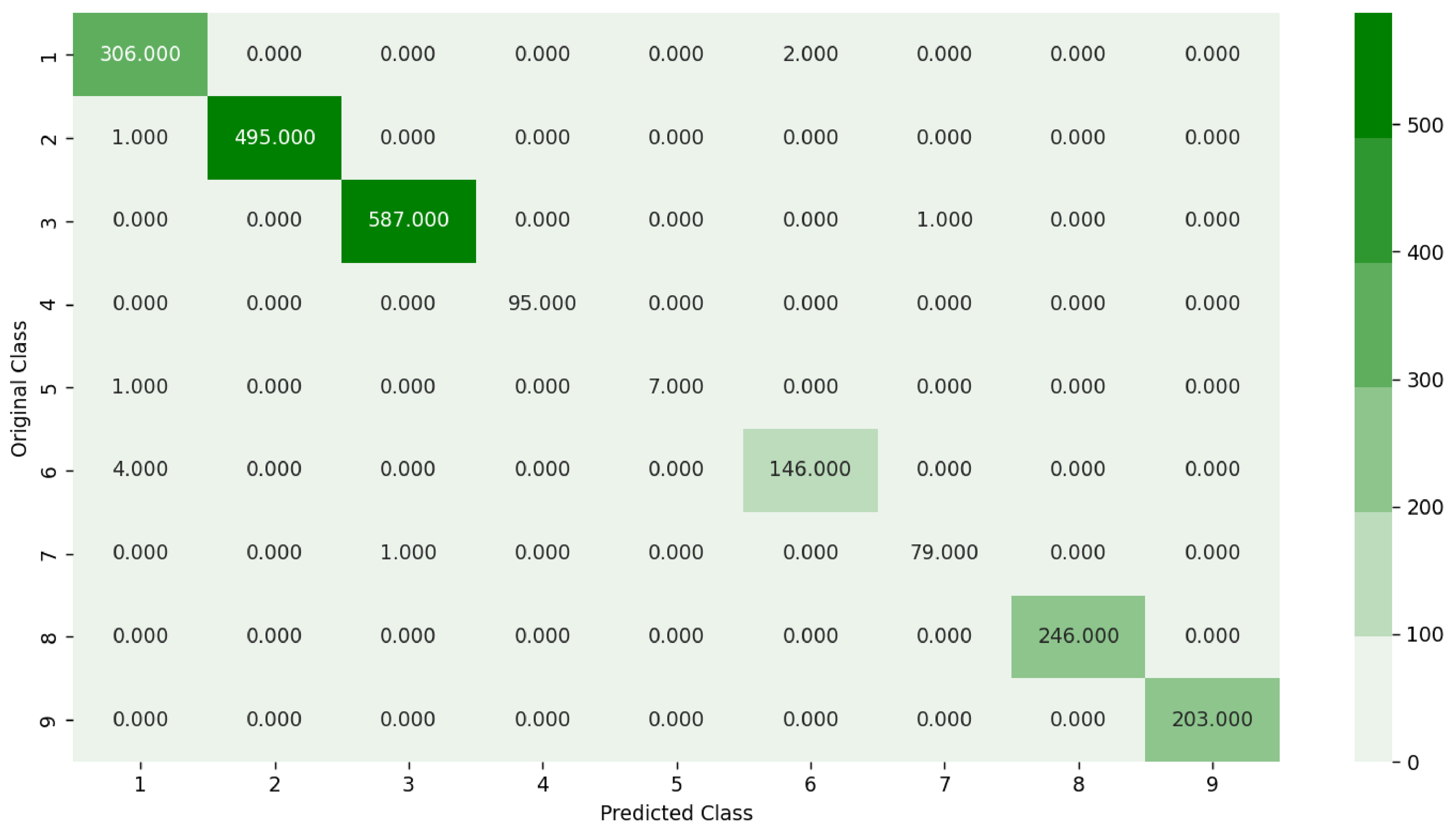



```
For values of best alpha =  100 The train log loss is: 0.012043317011966856
For values of best alpha =  100 The cross validation log loss is: 0.01624964757759834
For values of best alpha =  100 The test log loss is: 0.02383102390750985
```

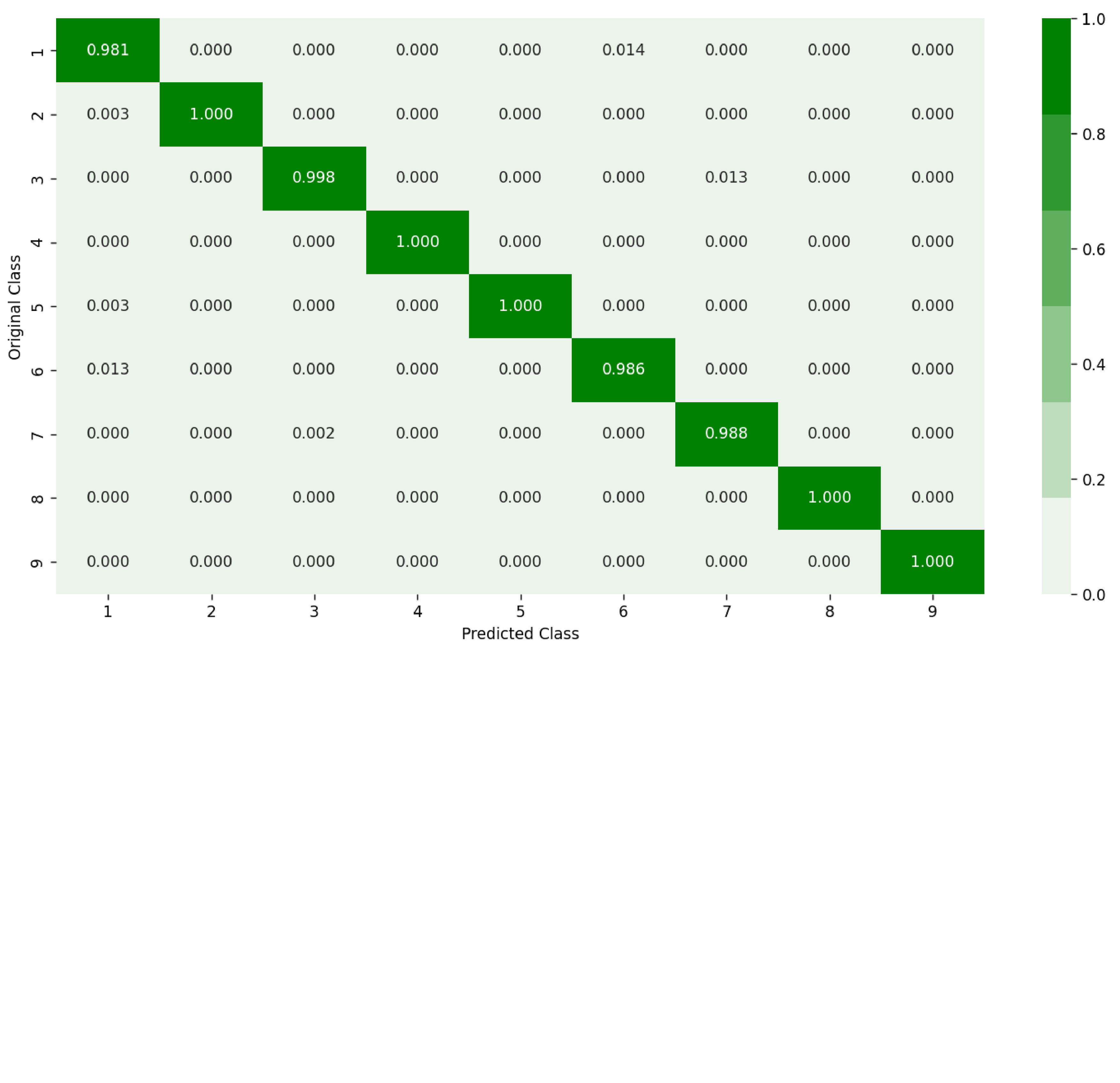
In [54]:

```
# predict_y is test prediction
predicted_y =np.argmax(predict_y, axis=1)
plot_confusion_matrix(y_test, predicted_y+1)
```

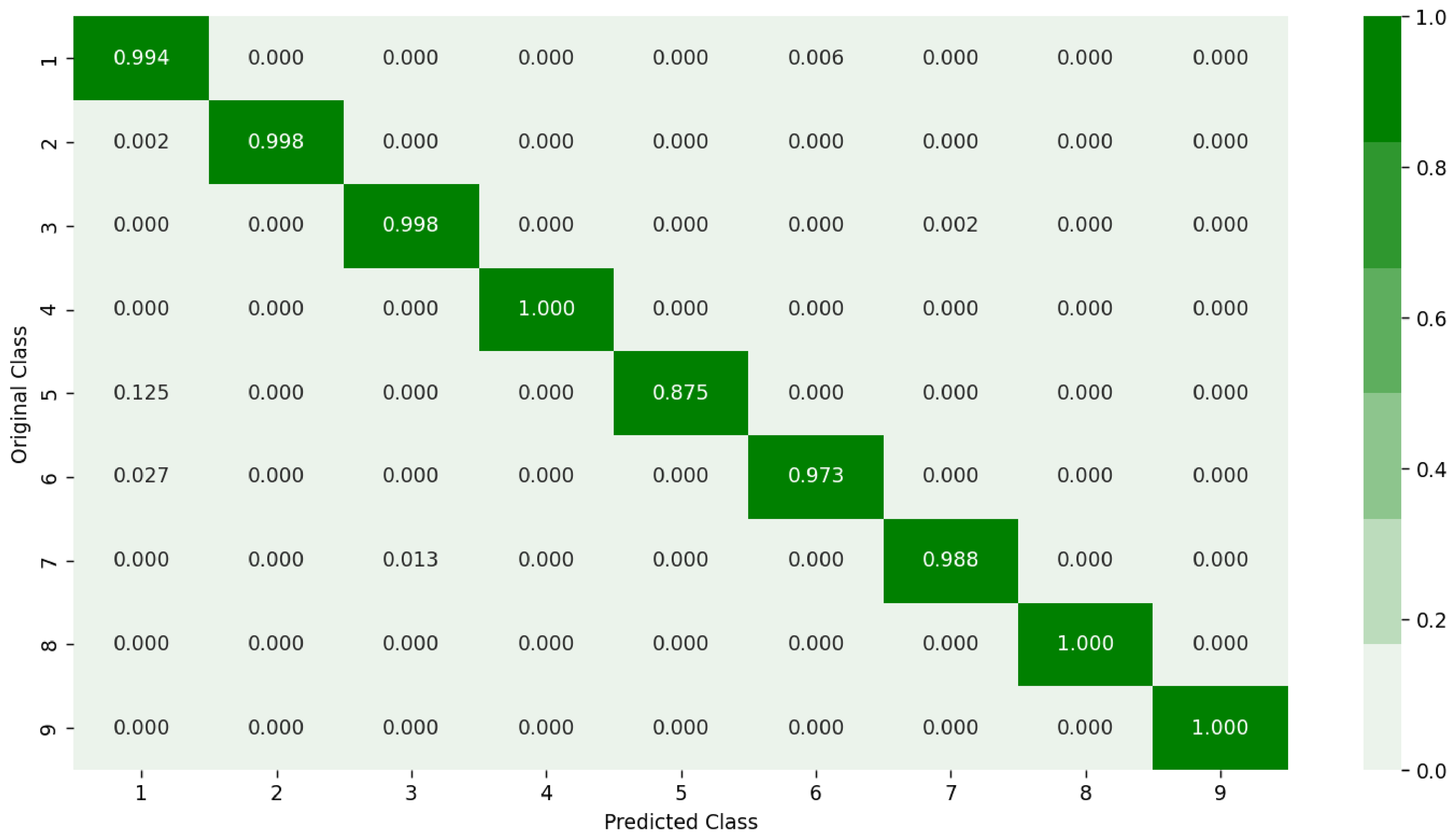
```
Number of misclassified points  0.45998160073597055
-------------------------------------------------- Confusion matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
-------------------------------------------------- Precision matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
-------------------------------------------------- Recall matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```



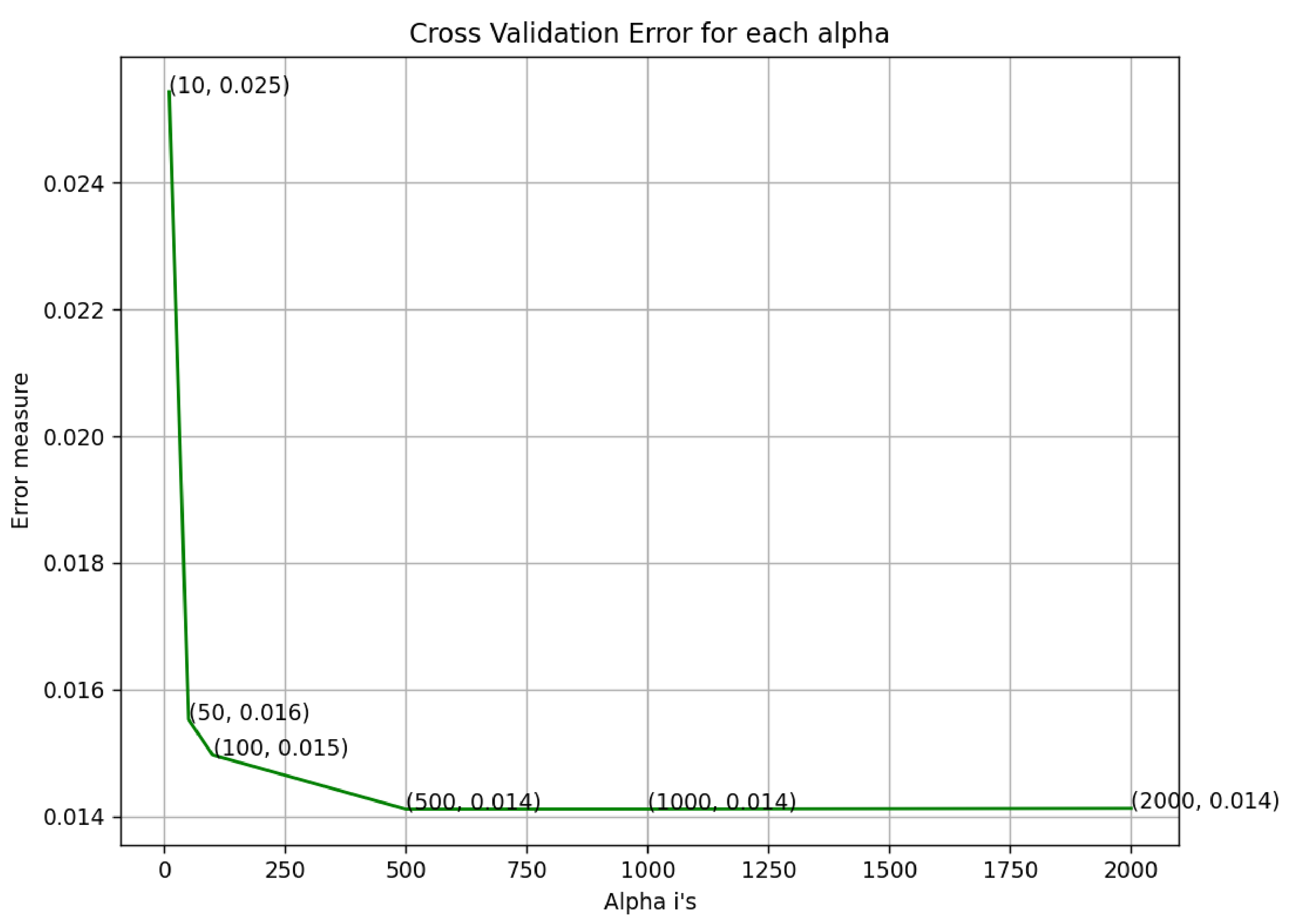

```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

3.1.2 XGB Classifier with Asm-uni grams + ASM-Extracted image features

In [79]:

```
%%time
# Training a hyper-parameter tuned Xg-Boost regressor on our train data

# find more about XGBClassifier function here http://xgboost.readthedocs.io/en/latest/python/python_api.html?#xgboost.XGBClassifier
# -------------------------
# default paramters
# class xgboost.XGBClassifier(max_depth=3, learning_rate=0.1, n_estimators=100, silent=True,
# objective='binary:logistic', booster='gbtree', n_jobs=1, nthread=None, gamma=0, min_child_weight=1,
# max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1,
# scale_pos_weight=1, base_score=0.5, random_state=0, seed=None, missing=None, **kwargs)

# some of methods of RandomForestRegressor()
# fit(X, y, sample_weight=None, eval_set=None, eval_metric=None, early_stopping_rounds=None, verbose=True, xgb_model=None)
# get_params([deep])    Get parameters for this estimator.
# predict(data, output_margin=False, ntree_limit=0) : Predict with data. NOTE: This function is not thread safe.
# get_score(importance_type='weight') -> get the feature importance
# -----------------------
# video link1: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/regression-using-decision-trees-2/
# video link2: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/what-are-ensembles/
# -----------------------

alpha=[10,50,100,500,1000,2000]
cv_log_error_array=[]
for i in alpha:
    x_cfl=XGBClassifier(n_estimators=i,nthread=-1,silent=True,verbosity=0)
    x_cfl.fit(X_train,y_train)
    sig_clf = CalibratedClassifierCV(x_cfl, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_cv)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=x_cfl.classes_, eps=1e-15))
    print('Parameter tuning for alpha:',i,' Loss:',log_loss(y_cv, predict_y, labels=r_cfl.classes_, eps=1e-15))


# for i in range(len(cv_log_error_array)):
#     print ('log_loss for c = ',alpha[i],'is',cv_log_error_array[i])


best_alpha = np.argmin(cv_log_error_array)

fig, ax = plt.subplots()
ax.plot(alpha, cv_log_error_array,c='g')
for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],cv_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()

x_cfl=XGBClassifier(n_estimators=alpha[best_alpha],nthread=-1,silent=True,verbosity=0)
x_cfl.fit(X_train,y_train)
sig_clf = CalibratedClassifierCV(x_cfl, 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))
predict_y = sig_clf.predict_proba(X_cv)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y))
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, predict_y))
```

```
Parameter tuning for alpha: 10  Loss: 0.025430225353088646
Parameter tuning for alpha: 50  Loss: 0.01553217557088968
Parameter tuning for alpha: 100  Loss: 0.014970418779213806
Parameter tuning for alpha: 500  Loss: 0.014116494279968701
Parameter tuning for alpha: 1000  Loss: 0.014117920024732094
Parameter tuning for alpha: 2000  Loss: 0.014129280344415004
log_loss for c =  10 is 0.025430225353088646
log_loss for c =  50 is 0.01553217557088968
log_loss for c =  100 is 0.014970418779213806
log_loss for c =  500 is 0.014116494279968701
log_loss for c =  1000 is 0.014117920024732094
log_loss for c =  2000 is 0.014129280344415004

<IPython.core.display.Javascript object>
```



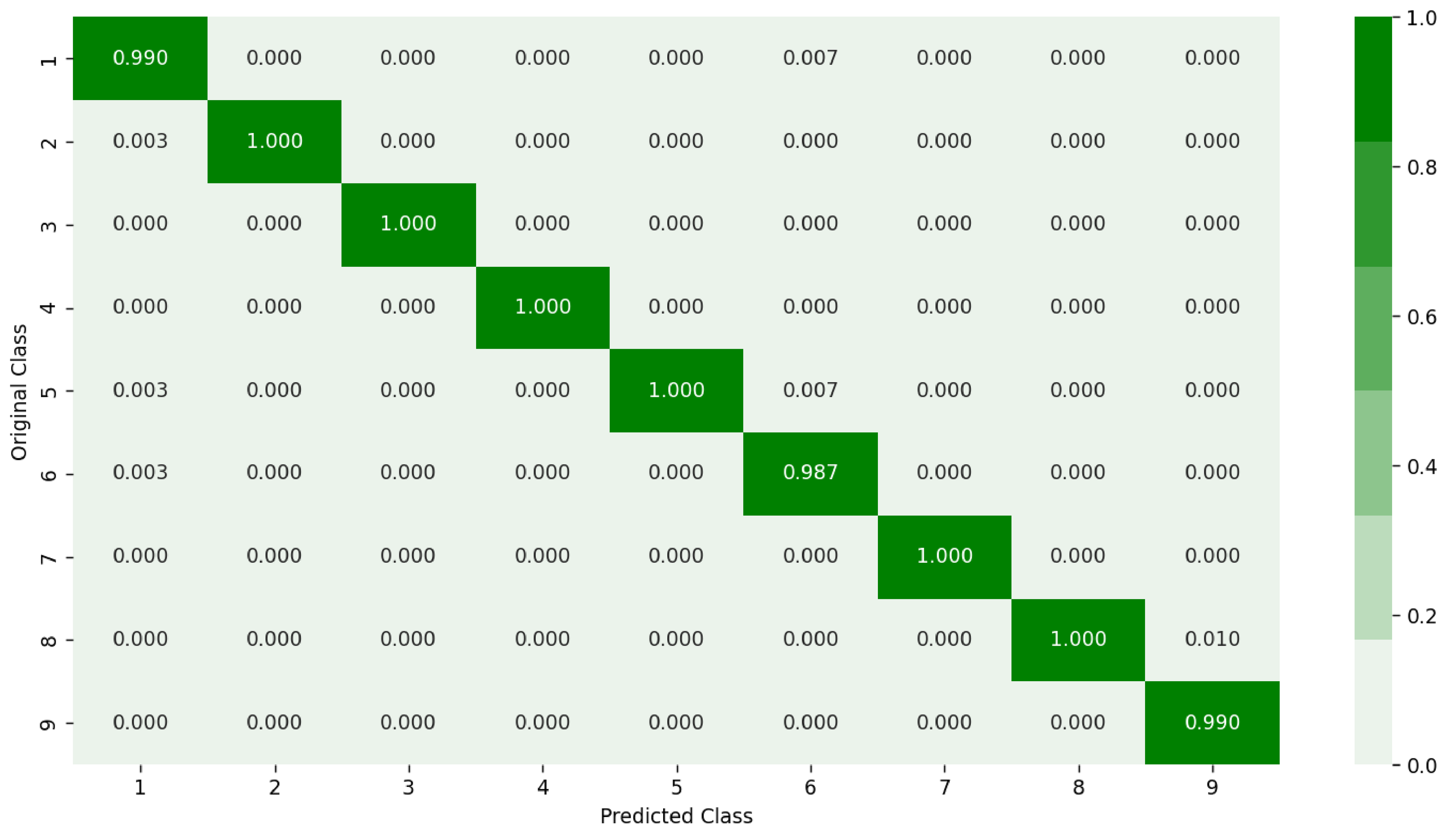

```
For values of best alpha =  500 The train log loss is: 0.010945769938987372
For values of best alpha =  500 The cross validation log loss is: 0.014116494279968701
For values of best alpha =  500 The test log loss is: 0.023801783541974996
```

In [80]:

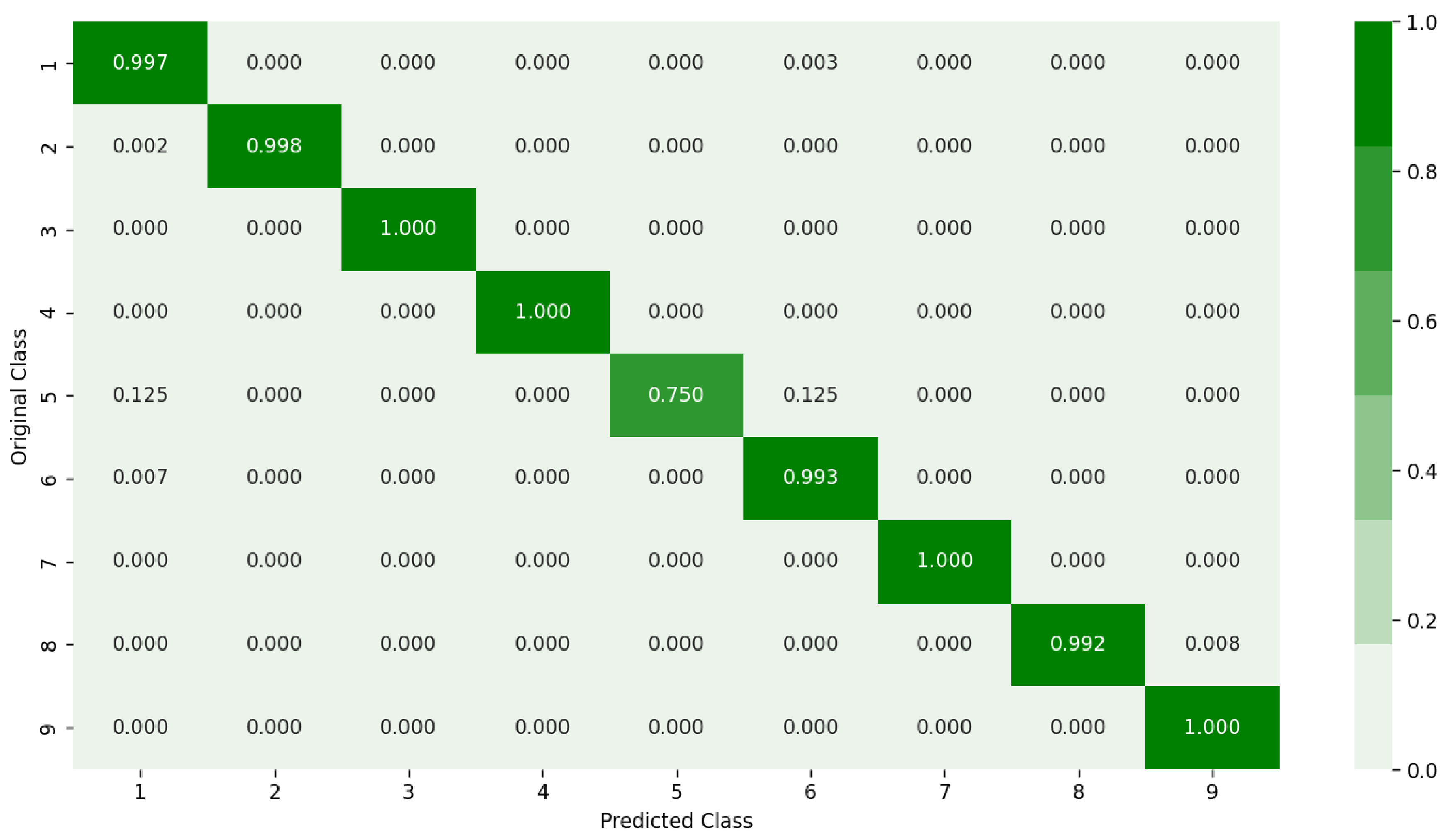
```
# predict_y is test prediction
predicted_y =np.argmax(predict_y, axis=1)
plot_confusion_matrix(y_test, predicted_y+1)
```

```
Number of misclassified points  0.3219871205151794
-------------------------------------------------- Confusion matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```


```
-------------------------------------------------- Precision matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
-------------------------------------------------- Recall matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

3.1.3 XGB Classifier with Asm-uni grams + ASM-Extracted image features (with best hyper paramters)

[https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/ (https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/)](https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/)

In [16]:
```
%%time
x_cfl=XGBClassifier(verbosity=0,silent=True)
prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500],
     'max_depth':[3,5,10],
    'colsample_bytree':[0.1,0.3,0.5,1],
    'subsample':[0.1,0.3,0.5,1]
}
random_cfl1=RandomizedSearchCV(x_cfl,param_distributions=prams,verbose=10,cv=4)
random_cfl1.fit(X_train,y_train)
```

```
Fitting 4 folds for each of 10 candidates, totalling 40 fits
[CV 1/4; 1/10] START colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3
[CV 1/4; 1/10] END colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3;, score=0.989 total time=  18.0s
[CV 2/4; 1/10] START colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3
[CV 2/4; 1/10] END colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3;, score=0.992 total time=  32.2s
[CV 3/4; 1/10] START colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3
[CV 3/4; 1/10] END colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3;, score=0.987 total time=  32.5s
[CV 4/4; 1/10] START colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3
[CV 4/4; 1/10] END colsample_bytree=0.3, learning_rate=0.03, max_depth=5, n_estimators=100, subsample=0.3;, score=0.994 total time=  33.0s
[CV 1/4; 2/10] START colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3
[CV 1/4; 2/10] END colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3;, score=0.997 total time=  43.6s
[CV 2/4; 2/10] START colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3
[CV 2/4; 2/10] END colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3;, score=0.997 total time=  43.0s
[CV 3/4; 2/10] START colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3
[CV 3/4; 2/10] END colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3;, score=0.995 total time=  42.8s
[CV 4/4; 2/10] START colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3
[CV 4/4; 2/10] END colsample_bytree=0.1, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.3;, score=0.997 total time=  43.4s
[CV 1/4; 3/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 1/4; 3/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.995 total time=  40.0s
[CV 2/4; 3/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 2/4; 3/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.992 total time=  39.3s
[CV 3/4; 3/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 3/4; 3/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.990 total time=  40.5s
[CV 4/4; 3/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 4/4; 3/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.996 total time=  41.0s
[CV 1/4; 4/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 1/4; 4/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.995 total time=  27.7s
[CV 2/4; 4/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 2/4; 4/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.993 total time=  27.0s
[CV 3/4; 4/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 3/4; 4/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.991 total time=  27.5s
[CV 4/4; 4/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1
[CV 4/4; 4/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=10, n_estimators=200, subsample=0.1;, score=0.995 total time=  27.7s
[CV 1/4; 5/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3
[CV 1/4; 5/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3;, score=0.995 total time= 1.6min
[CV 2/4; 5/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3
[CV 2/4; 5/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3;, score=0.995 total time= 1.6min
[CV 3/4; 5/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3
[CV 3/4; 5/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3;, score=0.993 total time= 1.6min
[CV 4/4; 5/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3
[CV 4/4; 5/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.3;, score=0.997 total time= 1.6min
[CV 1/4; 6/10] START colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3
[CV 1/4; 6/10] END colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3;, score=0.997 total time= 1.7min
[CV 2/4; 6/10] START colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3
[CV 2/4; 6/10] END colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3;, score=0.995 total time= 1.7min
[CV 3/4; 6/10] START colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3
[CV 3/4; 6/10] END colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3;, score=0.993 total time= 1.7min
[CV 4/4; 6/10] START colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3
[CV 4/4; 6/10] END colsample_bytree=1, learning_rate=0.2, max_depth=3, n_estimators=200, subsample=0.3;, score=0.996 total time= 1.8min
[CV 1/4; 7/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3
[CV 1/4; 7/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3;, score=0.995 total time=  24.8s
[CV 2/4; 7/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3
[CV 2/4; 7/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3;, score=0.995 total time=  24.3s
[CV 3/4; 7/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3
[CV 3/4; 7/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3;, score=0.993 total time=  24.4s
[CV 4/4; 7/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3
[CV 4/4; 7/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=100, subsample=0.3;, score=0.996 total time=  24.3s
[CV 1/4; 8/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1
[CV 1/4; 8/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1;, score=0.978 total time=  15.3s
[CV 2/4; 8/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1
[CV 2/4; 8/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1;, score=0.979 total time=  15.5s
[CV 3/4; 8/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1
[CV 3/4; 8/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1;, score=0.978 total time=  15.3s
[CV 4/4; 8/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1
[CV 4/4; 8/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=0.1;, score=0.984 total time=  15.0s
[CV 1/4; 9/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1
[CV 1/4; 9/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1;, score=0.995 total time= 1.2min
[CV 2/4; 9/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1
[CV 2/4; 9/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1;, score=0.994 total time= 1.1min
[CV 3/4; 9/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1
[CV 3/4; 9/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1;, score=0.992 total time= 1.2min
[CV 4/4; 9/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1
[CV 4/4; 9/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=10, n_estimators=500, subsample=0.1;, score=0.997 total time= 1.2min
[CV 1/4; 10/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3
[CV 1/4; 10/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3;, score=0.995 total time= 5.4min
[CV 2/4; 10/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3
[CV 2/4; 10/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3;, score=0.996 total time= 5.4min
[CV 3/4; 10/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3
[CV 3/4; 10/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3;, score=0.993 total time= 5.3min
[CV 4/4; 10/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3
[CV 4/4; 10/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=500, subsample=0.3;, score=0.996 total time= 5.4min
CPU times: user 1h 43min 44s, sys: 5.84 s, total: 1h 43min 50s
Wall time: 52min 11s
```

Out[16]:
```
RandomizedSearchCV(cv=4,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, gamma=None,
                                           gpu_id=None, importance_type='gain',
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=None,
                                           min_child_weight=None, missing=nan,
                                           monotone_constraints=None,
                                           n_estimators=100,...
                                           num_parallel_tree=None,
                                           random_state=None, reg_alpha=None,
                                           reg_lambda=None,
                                           scale_pos_weight=None, silent=True,
                                           subsample=None, tree_method=None,
                                           validate_parameters=None,
                                           verbosity=0),
                   param_distributions={'colsample_bytree': [0.1, 0.3, 0.5, 1],
                                        'learning_rate': [0.01, 0.03, 0.05, 0.1,
                                                          0.15, 0.2],
                                        'max_depth': [3, 5, 10],
                                        'n_estimators': [100, 200, 500],
                                        'subsample': [0.1, 0.3, 0.5, 1]},
                   verbose=10)
```

In [17]:
```
print (random_cfl1.best_params_)
```

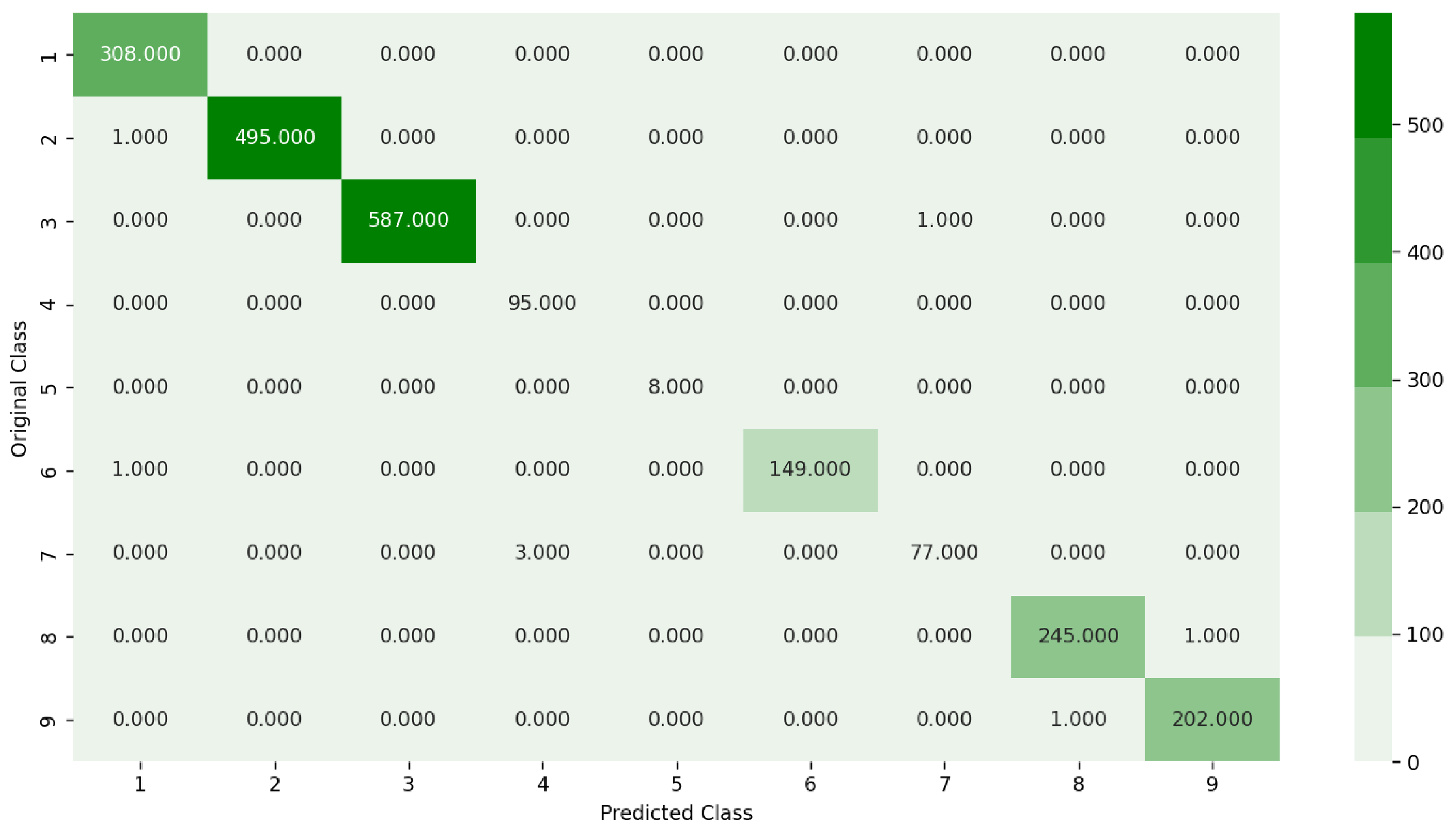
```
{'subsample': 0.3, 'n_estimators': 500, 'max_depth': 10, 'learning_rate': 0.1, 'colsample_bytree': 0.1}
```

In [135]:
```
%%time
# Training a hyper-parameter tuned Xg-Boost regressor on our train data

# find more about XGBClassifier function here http://xgboost.readthedocs.io/en/latest/python/python_api.html?#xgboost.XGBClassifier
# -------------------------
# default paramters
# class xgboost.XGBClassifier(max_depth=3, learning_rate=0.1, n_estimators=100, silent=True,
# objective='binary:logistic', booster='gbtree', n_jobs=1, nthread=None, gamma=0, min_child_weight=1,
# max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1,
# scale_pos_weight=1, base_score=0.5, random_state=0, seed=None, missing=None, **kwargs)

# some of methods of RandomForestRegressor()
# fit(X, y, sample_weight=None, eval_set=None, eval_metric=None, early_stopping_rounds=None, verbose=True, xgb_model=None)
# get_params([deep])    Get parameters for this estimator.
# predict(data, output_margin=False, ntree_limit=0) : Predict with data. NOTE: This function is not thread safe.
# get_score(importance_type='weight') -> get the feature importance
# -----------------------
# video link2: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/what-are-ensembles/
# -----------------------
x_cfl=XGBClassifier(n_estimators=500, subsample=0.3,learning_rate=0.1, colsample_bytree=0.1, max_depth=10)
x_cfl.fit(X_train,y_train)
c_cfl=CalibratedClassifierCV(x_cfl,method='sigmoid')
c_cfl.fit(X_train,y_train)

predict_y = c_cfl.predict_proba(X_train)
print ('train loss',log_loss(y_train, predict_y))
predict_y = c_cfl.predict_proba(X_cv)

print ('cv loss',log_loss(y_cv, predict_y))
predict_y = c_cfl.predict_proba(X_test)
print ('test loss',log_loss(y_test, predict_y))
```

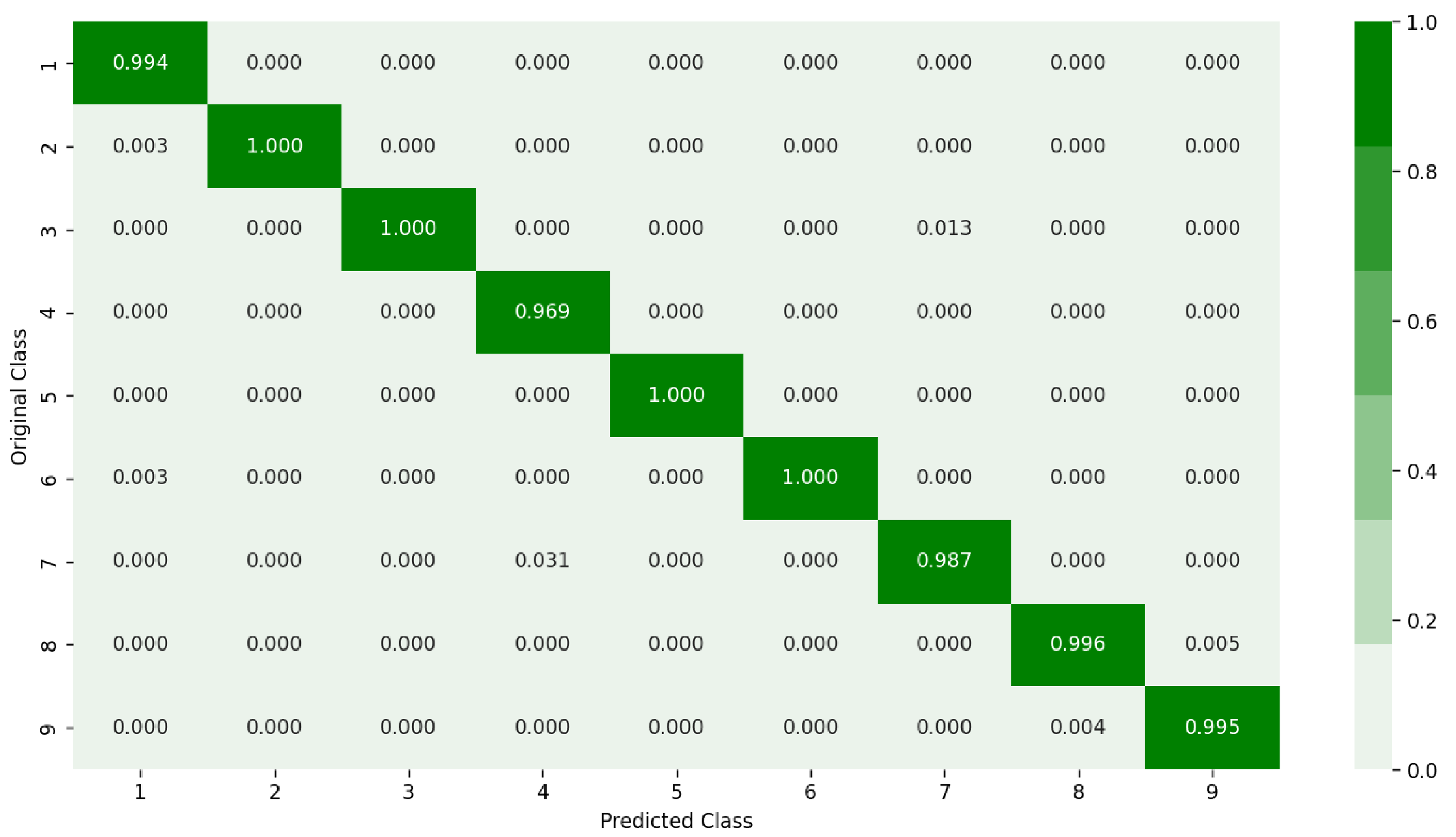
```
train loss 0.011000827447175468
cv loss 0.021335735778038614
test loss 0.025312012599598657
CPU times: user 5min 22s, sys: 530 ms, total: 5min 22s
Wall time: 2min 43s
```

In [136]:
```python
# predict_y is test prediction
predicted_y =np.argmax(predict_y, axis=1)
plot_confusion_matrix(y_test, predicted_y+1)
```

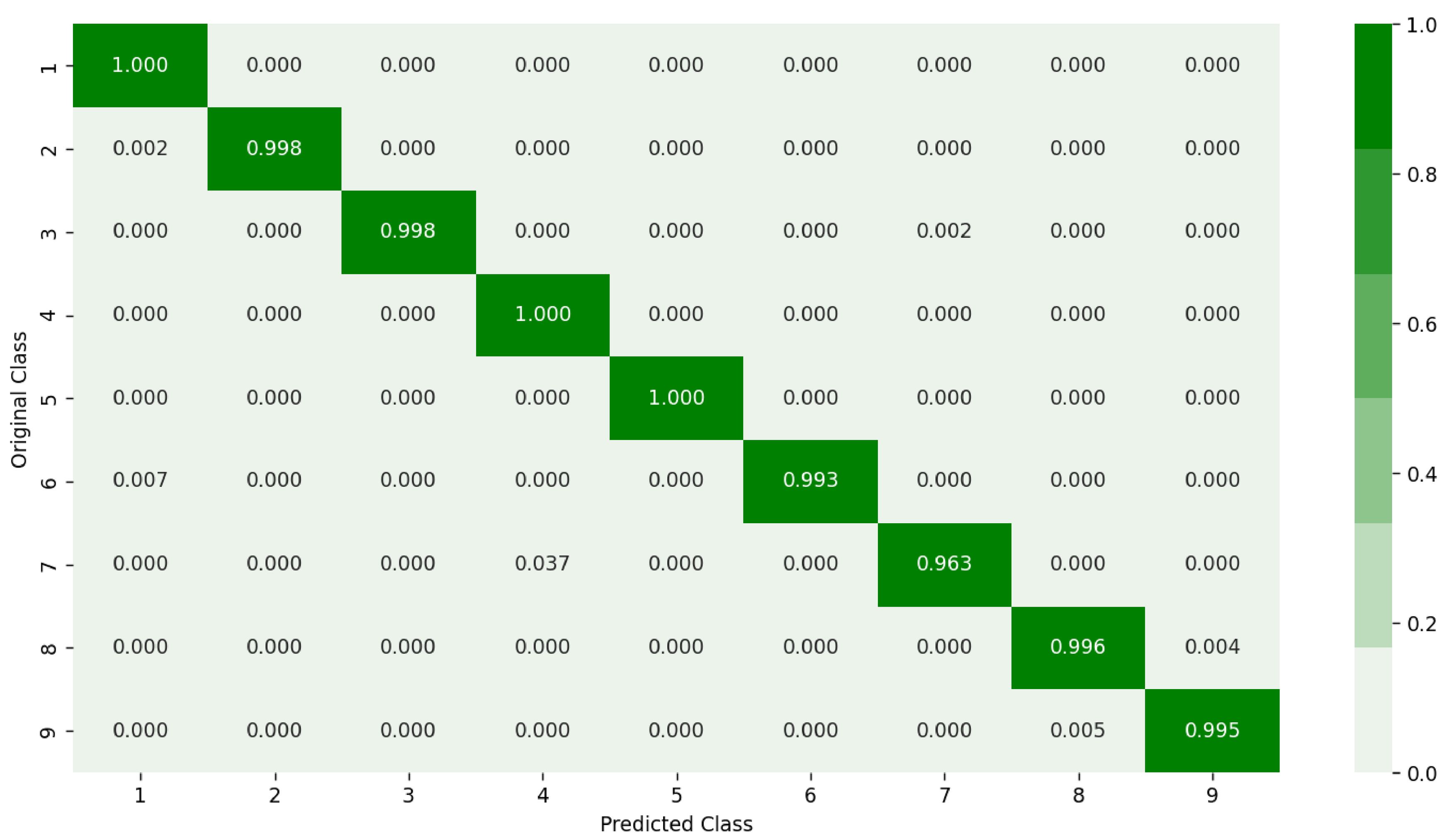
```
Number of misclassified points  0.24798528058877645
-------------------------------------------------- Confusion matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
-------------------------------------------------- Precision matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
-------------------------------------------------- Recall matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

3.2 Random Forest Classifier , Asm-uni grams , ASM-Extracted image features , Bytes uni-grams

In [3]:
```
byte_features=pd.read_csv("result.csv")
byte_features['ID']  = byte_features['ID'].str.split('.').str[0]
byte_features.head(2)
```

Out[3]:

	ID	0	1	2	3	4	5	6	7	8	...	f7	f8	f9	fa	fb	fc	fd	fe	ff	??
0	01azqd4InC7m9JpocGv5	601905	3905	2816	3832	3345	3242	3650	3201	2965	...	2804	3687	3101	3211	3097	2758	3099	2759	5753	1824
1	01IsoiSMh5gxyDYTl4CB	39755	8337	7249	7186	8663	6844	8420	7589	9291	...	451	6536	439	281	302	7639	518	17001	54902	8588

2 rows × 258 columns

In [4]:
```
byte_features_with_size = pd.read_csv("result_with_size.csv")
byte_features_with_size = byte_features_with_size.drop(columns='Unnamed: 0')
byte_features_with_size.head(5)
```

Out[4]:

	ID	0	1	2	3	4	5	6	7	8	...	f9	fa	fb	fc	fd	fe	ff	??	size	Class
0	01azqd4InC7m9JpocGv5	601905	3905	2816	3832	3345	3242	3650	3201	2965	...	3101	3211	3097	2758	3099	2759	5753	1824	4.148438	9
1	01IsoiSMh5gxyDYTl4CB	39755	8337	7249	7186	8663	6844	8420	7589	9291	...	439	281	302	7639	518	17001	54902	8588	5.425781	2
2	01jsnpXSAlgw6aPeDxrU	93506	9542	2568	2438	8925	9330	9007	2342	9107	...	2242	2885	2863	2471	2786	2680	49144	468	3.808594	9
3	01kcPWA9K2BOxQeS5Rju	21091	1213	726	817	1257	625	550	523	1078	...	485	462	516	1133	471	761	7998	13940	0.562500	1
4	01SuzwMJEIXsK7A8dQbl	19764	710	302	433	559	410	262	249	422	...	350	209	239	653	221	242	2199	9008	0.363281	8

5 rows × 260 columns

In [5]:
```
# https://stackoverflow.com/a/29651514
def normalize(df):
    result1 = df.copy()
    for feature_name in df.columns:
        if (str(feature_name) != str('ID') and str(feature_name)!=str('Class')):
            max_value = df[feature_name].max()
            min_value = df[feature_name].min()
            result1[feature_name] = (df[feature_name] - min_value) / (max_value - min_value)
    return result1
result = normalize(byte_features_with_size)

```

In [7]:
```
labels = pd.read_csv('trainLabels.csv')
data_50 = pd.read_csv('asm_50%_variance_data.csv')
```

In [8]:
```
result_asm = pd.read_csv("asmoutputfile.csv")

result_asm = normalize(result_asm)

labels.columns = ['ID', 'Class']
result_asm = pd.merge(result_asm, labels,on='ID', how='inner')
result_asm.head()
```

Out[8]:

	ID	HEADER:	.text:	.Pav:	.idata:	.data:	.bss:	.rdata:	.edata:	.rsrc:	...	edx	esi	eax	ebx	ecx	edi	ebp	esp	eip	Class
0	01kcPWA9K2BOxQeS5Rju	0.107345	0.001092	0.0	0.000761	0.000023	0.0	0.000084	0.0	0.000072	...	0.000343	0.000746	0.000301	0.000360	0.001057	0.0	0.030797	0.001468	0.003173	1
1	1E93CpP60RHFNiT5Qfvn	0.096045	0.001230	0.0	0.000617	0.000019	0.0	0.000000	0.0	0.000072	...	0.000343	0.000328	0.000965	0.000686	0.000153	0.0	0.025362	0.000000	0.002188	1
2	3ekVow2ajZHbTnBcsDfX	0.096045	0.000627	0.0	0.000300	0.000017	0.0	0.000038	0.0	0.000072	...	0.000248	0.000475	0.000201	0.000560	0.000178	0.0	0.019928	0.000000	0.000985	1
3	3X2nY7iQaPBIWDrAZqJe	0.096045	0.000333	0.0	0.000258	0.000008	0.0	0.000000	0.0	0.000072	...	0.000114	0.000090	0.000281	0.000059	0.000025	0.0	0.014493	0.000000	0.000657	1
4	46OZzdsSKDCFV8h7XWxf	0.096045	0.000590	0.0	0.000353	0.000068	0.0	0.000000	0.0	0.000072	...	0.000229	0.000102	0.000362	0.000243	0.000064	0.0	0.019928	0.000000	0.001204	1

5 rows × 53 columns

In [9]:
```
result.head()
```

Out[9]:

	ID	0	1	2	3	4	5	6	7	8	...	f9	fa	fb	fc	fd	fe	ff	??	size	Class
0	01azqd4InC7m9JpocGv5	0.262806	0.005498	0.001567	0.002067	0.002048	0.001835	0.002058	0.002946	0.002638	...	0.013560	0.013107	0.013634	0.031724	0.014549	0.014348	0.007843	0.000129	0.092219	9
1	01IsoiSMh5gxyDYTl4CB	0.017358	0.011737	0.004033	0.003876	0.005303	0.003873	0.004747	0.006984	0.008267	...	0.001920	0.001147	0.001329	0.087867	0.002432	0.088411	0.074851	0.000606	0.121236	2
2	01jsnpXSAlgw6aPeDxrU	0.040827	0.013434	0.001429	0.001315	0.005464	0.005280	0.005078	0.002155	0.008104	...	0.009804	0.011777	0.012604	0.028423	0.013080	0.013937	0.067001	0.000033	0.084499	9
3	01kcPWA9K2BOxQeS5Rju	0.009209	0.001708	0.000404	0.000441	0.000770	0.000354	0.000310	0.000481	0.000959	...	0.002121	0.001886	0.002272	0.013032	0.002211	0.003957	0.010904	0.000984	0.010759	1
4	01SuzwMJEIXsK7A8dQbl	0.008629	0.001000	0.000168	0.000234	0.000342	0.000232	0.000148	0.000229	0.000376	...	0.001530	0.000853	0.001052	0.007511	0.001038	0.001258	0.002998	0.000636	0.006233	8

5 rows × 260 columns

In [10]:
```
print('total features (asm unigram + bytes-unigram + asm-img):',result.shape[1]+result_asm.shape[1]+data_50.shape[1])
```

```
total features (asm unigram + bytes-unigram + asm-img): 815
```

In [11]:
```
X_merged = pd.merge(result_asm,result.drop(columns=['size','Class']),on='ID',how='inner')
X_merged = pd.merge(X_merged, data_50, on='ID', how='inner')
```

In [12]:
```
X_merged.shape
```

Out[12]:
```
(10868, 811)
```

In [13]:
```
Y_merged = X_merged['Class']
X_merged = X_merged.drop(['ID','rtn','.BSS:','.CODE','Class'], axis=1)
print('shape of x',X_merged.shape,' y:',Y_merged.shape)
```

```
shape of x (10868, 806)  y: (10868,)
```

In [14]:
```
### train test split
X_train, X_test, y_train, y_test = train_test_split(X_merged, Y_merged,stratify=Y_merged,test_size=0.20)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,stratify=y_train,test_size=0.20)
```

In [15]:
```
X_train = X_train.astype(float)
X_test = X_test.astype(float)
X_cv = X_cv.astype(float)
```

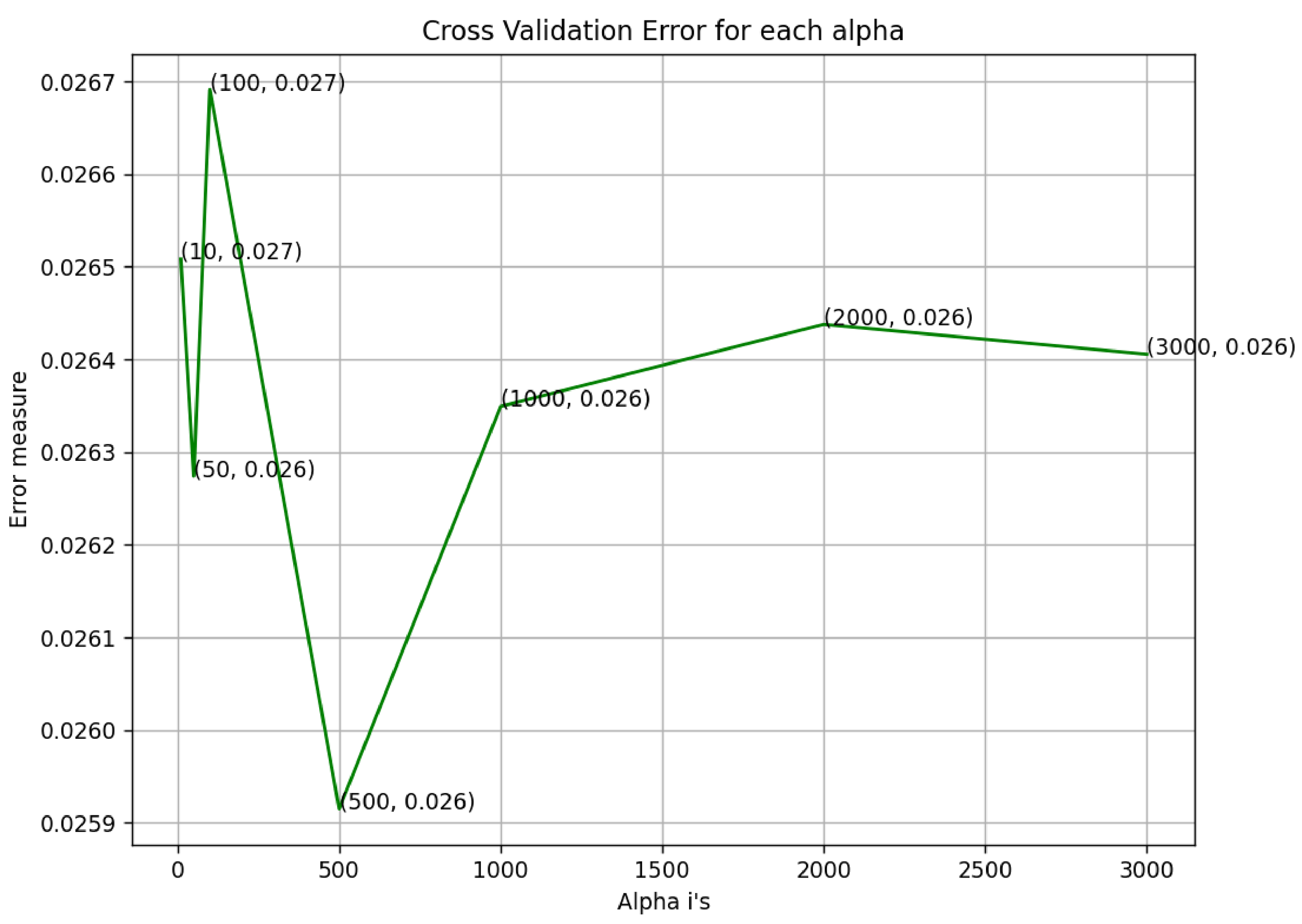
3.2.1 Random Forest Classifier with Asm-uni grams + ASM-Extracted image features + Byte uni gram

In [138]:

```
# --------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False,
# class_weight=None)

# Some of methods of RandomForestClassifier()
# fit(X, y, [sample_weight])    Fit the SVM model according to the given training data.
# predict(X)    Perform classification on samples in X.
# predict_proba (X) Perform classification on samples in X.

# some of attributes of  RandomForestClassifier()
# feature_importances_ : array of shape = [n_features]
# The feature importances (the higher, the more important the feature).

# --------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/random-forest-and-their-construction-2/
# --------------------------------

alpha=[10,50,100,500,1000,2000,3000]
cv_log_error_array=[]
train_log_error_array=[]
for i in alpha:
    r_cfl=RandomForestClassifier(n_estimators=i,random_state=42,n_jobs=-1)
    r_cfl.fit(X_train,y_train)
    sig_clf = CalibratedClassifierCV(r_cfl, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_cv)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=r_cfl.classes_, eps=1e-15))
    print('Parameter tuning for alpha:',i,' Loss:',log_loss(y_cv, predict_y, labels=r_cfl.classes_, eps=1e-15))

# for i in range(len(cv_log_error_array)):
#     print ('log_loss for c = ',alpha[i],'is',cv_log_error_array[i])

best_alpha = np.argmin(cv_log_error_array)

fig, ax = plt.subplots()
ax.plot(alpha, cv_log_error_array,c='g')

for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],cv_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()


r_cfl=RandomForestClassifier(n_estimators=alpha[best_alpha],random_state=42,n_jobs=-1)
r_cfl.fit(X_train,y_train)
sig_clf = CalibratedClassifierCV(r_cfl, 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))
predict_y = sig_clf.predict_proba(X_cv)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y))
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, predict_y))
```

```
Parameter tuning for alpha: 10  Loss: 0.026508445703490754
Parameter tuning for alpha: 50  Loss: 0.026274183912199087
Parameter tuning for alpha: 100  Loss: 0.026691217424787736
Parameter tuning for alpha: 500  Loss: 0.025914948184497147
Parameter tuning for alpha: 1000  Loss: 0.02634931812198413
Parameter tuning for alpha: 2000  Loss: 0.02643794647709983
Parameter tuning for alpha: 3000  Loss: 0.026405623759426712
```

<IPython.core.display.Javascript object>



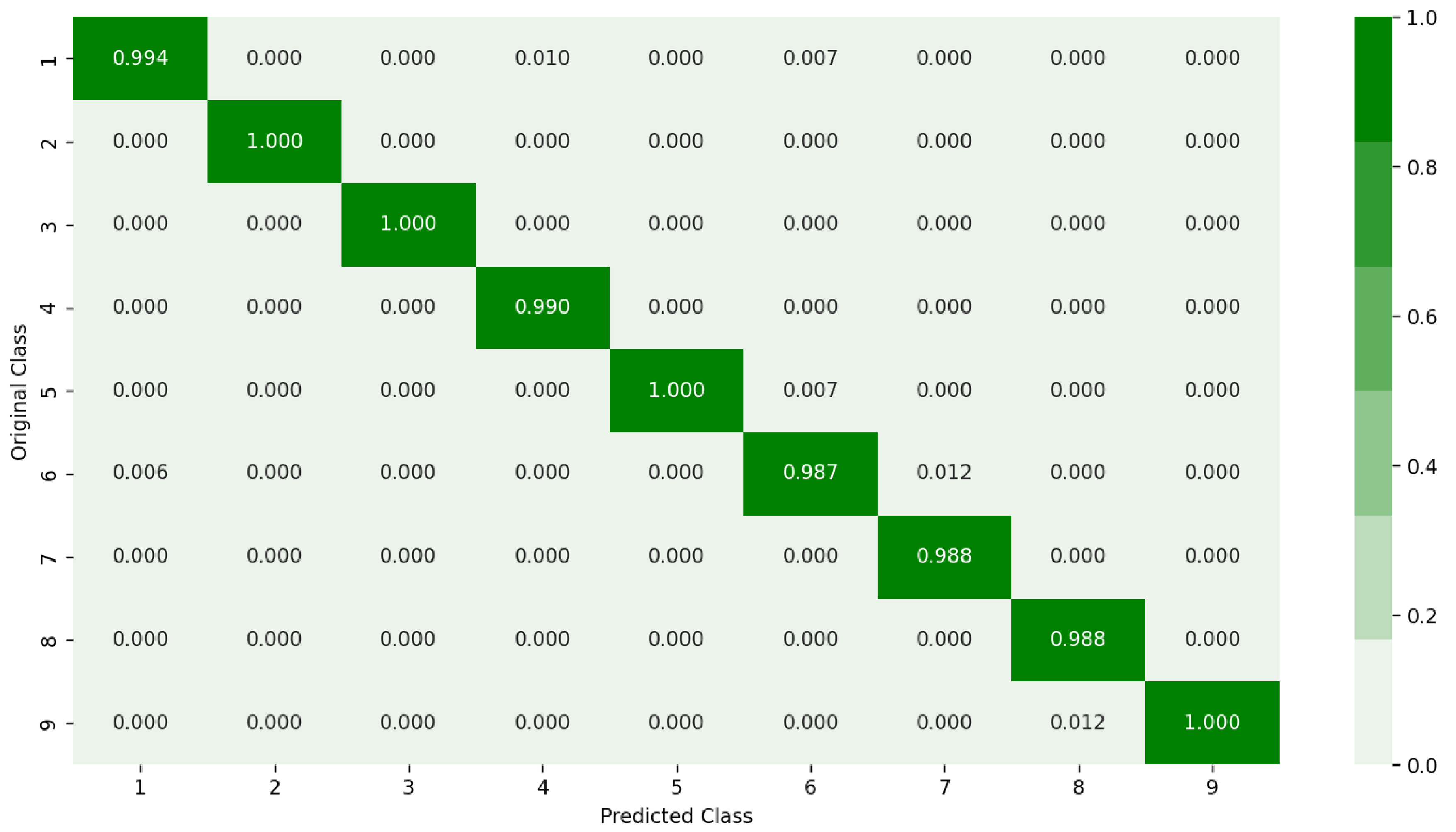

```
For values of best alpha =  500 The train log loss is: 0.010860945011193368
For values of best alpha =  500 The cross validation log loss is: 0.025914948184497147
For values of best alpha =  500 The test log loss is: 0.024136455428315266
```

In [139]:

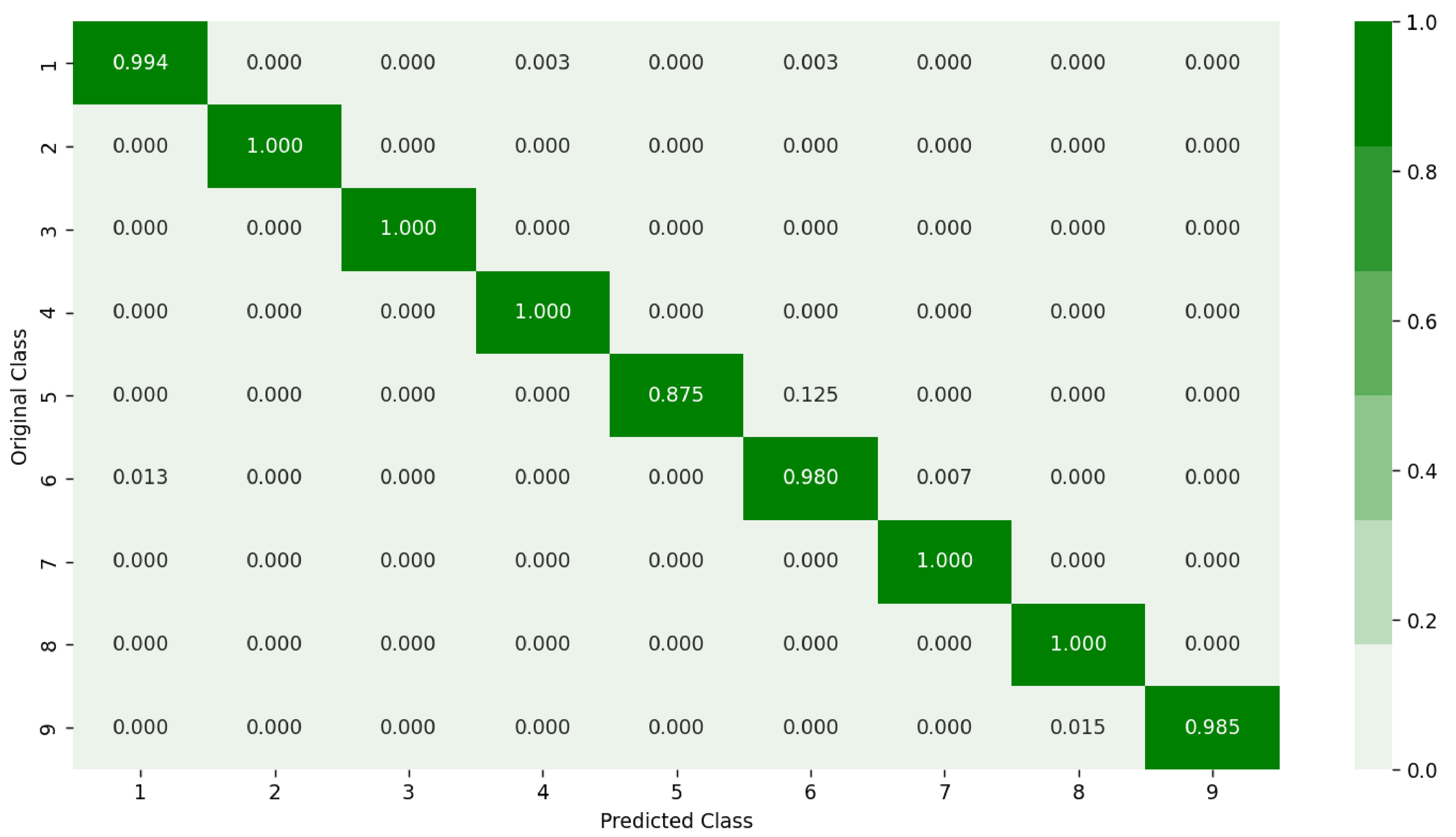
```
# predict_y is test prediction
predicted_y =np.argmax(predict_y, axis=1)
plot_confusion_matrix(y_test, predicted_y+1)
```

```
Number of misclassified points  0.41398344066237347
-------------------------------------------------- Confusion matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```


```
-------------------------------------------------- Precision matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
-------------------------------------------------- Recall matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

3.2.2 Xgboost Classifier with best hyperparameters

In [25]:

```
%%time
x_cfl=XGBClassifier(silent=True,verbosity=0)

prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,1000,2000],
     'max_depth':[3,5,10,12],
    'colsample_bytree':[0.1,0.3,0.5,1],
    'subsample':[0.1,0.3,0.5,1]
}
random_cfl=RandomizedSearchCV(x_cfl,param_distributions=prams,verbose=10,cv=4)
random_cfl.fit(X_train,y_train)
```

```
Fitting 4 folds for each of 10 candidates, totalling 40 fits
[CV 1/4; 1/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1
[CV 1/4; 1/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1;, score=0.996 total time= 6.6min
[CV 2/4; 1/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1
[CV 2/4; 1/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1;, score=0.997 total time= 6.5min
[CV 3/4; 1/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1
[CV 3/4; 1/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1;, score=0.993 total time= 6.5min
[CV 4/4; 1/10] START colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1
[CV 4/4; 1/10] END colsample_bytree=0.5, learning_rate=0.1, max_depth=10, n_estimators=1000, subsample=0.1;, score=0.997 total time= 6.6min
[CV 1/4; 2/10] START colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1
[CV 1/4; 2/10] END colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1;, score=0.997 total time=11.4min
[CV 2/4; 2/10] START colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1
[CV 2/4; 2/10] END colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1;, score=0.997 total time=11.3min
[CV 3/4; 2/10] START colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1
[CV 3/4; 2/10] END colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1;, score=0.994 total time=10.9min
[CV 4/4; 2/10] START colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1
[CV 4/4; 2/10] END colsample_bytree=0.3, learning_rate=0.05, max_depth=12, n_estimators=1000, subsample=1;, score=0.995 total time=11.3min
[CV 1/4; 3/10] START colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3
[CV 1/4; 3/10] END colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3;, score=0.996 total time= 5.8min
[CV 2/4; 3/10] START colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3
[CV 2/4; 3/10] END colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3;, score=0.995 total time= 5.6min
[CV 3/4; 3/10] START colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3
[CV 3/4; 3/10] END colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3;, score=0.994 total time= 5.6min
[CV 4/4; 3/10] START colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3
[CV 4/4; 3/10] END colsample_bytree=0.5, learning_rate=0.03, max_depth=5, n_estimators=200, subsample=0.3;, score=0.995 total time= 5.7min
[CV 1/4; 4/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1
[CV 1/4; 4/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1;, score=0.997 total time=95.4min
[CV 2/4; 4/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1
[CV 2/4; 4/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1;, score=0.995 total time=61.4min
[CV 3/4; 4/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1
[CV 3/4; 4/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1;, score=0.996 total time=55.4min
[CV 4/4; 4/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1
[CV 4/4; 4/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=2000, subsample=1;, score=0.994 total time=56.0min
[CV 1/4; 5/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1
[CV 1/4; 5/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1;, score=0.991 total time= 2.3min
[CV 2/4; 5/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1
[CV 2/4; 5/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1;, score=0.989 total time= 2.3min
[CV 3/4; 5/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1
[CV 3/4; 5/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1;, score=0.991 total time= 2.3min
[CV 4/4; 5/10] START colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1
[CV 4/4; 5/10] END colsample_bytree=1, learning_rate=0.05, max_depth=5, n_estimators=200, subsample=0.1;, score=0.994 total time= 2.3min
[CV 1/4; 6/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5
[CV 1/4; 6/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5;, score=0.997 total time= 3.7min
[CV 2/4; 6/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5
[CV 2/4; 6/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5;, score=0.997 total time= 3.7min
[CV 3/4; 6/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5
[CV 3/4; 6/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5;, score=0.995 total time= 3.7min
[CV 4/4; 6/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5
[CV 4/4; 6/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=10, n_estimators=200, subsample=0.5;, score=0.995 total time= 3.7min
[CV 1/4; 7/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1
[CV 1/4; 7/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1;, score=0.997 total time= 8.6min
[CV 2/4; 7/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1
[CV 2/4; 7/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1;, score=0.997 total time= 8.5min
[CV 3/4; 7/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1
[CV 3/4; 7/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1;, score=0.994 total time= 8.2min
[CV 4/4; 7/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1
[CV 4/4; 7/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=5, n_estimators=1000, subsample=1;, score=0.995 total time= 8.7min
[CV 1/4; 8/10] START colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1
[CV 1/4; 8/10] END colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1;, score=0.991 total time=11.5min
[CV 2/4; 8/10] START colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1
[CV 2/4; 8/10] END colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1;, score=0.990 total time=11.6min
[CV 3/4; 8/10] START colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1
[CV 3/4; 8/10] END colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1;, score=0.992 total time=12.0min
[CV 4/4; 8/10] START colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1
[CV 4/4; 8/10] END colsample_bytree=1, learning_rate=0.01, max_depth=5, n_estimators=1000, subsample=0.1;, score=0.993 total time=11.9min
[CV 1/4; 9/10] START colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3
[CV 1/4; 9/10] END colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3;, score=0.991 total time= 2.4min
[CV 2/4; 9/10] START colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3
[CV 2/4; 9/10] END colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3;, score=0.991 total time= 2.4min
[CV 3/4; 9/10] START colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3
[CV 3/4; 9/10] END colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3;, score=0.991 total time= 2.4min
[CV 4/4; 9/10] START colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3
[CV 4/4; 9/10] END colsample_bytree=1, learning_rate=0.03, max_depth=3, n_estimators=100, subsample=0.3;, score=0.992 total time= 2.4min
[CV 1/4; 10/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1
[CV 1/4; 10/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1;, score=0.988 total time= 1.1min
[CV 2/4; 10/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1
[CV 2/4; 10/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1;, score=0.990 total time= 1.1min
[CV 3/4; 10/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1
[CV 3/4; 10/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1;, score=0.987 total time= 1.1min
[CV 4/4; 10/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1
[CV 4/4; 10/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=3, n_estimators=100, subsample=1;, score=0.991 total time= 1.1min
CPU times: user 14h 3min 2s, sys: 55.2 s, total: 14h 3min 58s
Wall time: 8h 6min 31s
```

Out[25]:

```
RandomizedSearchCV(cv=4,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, gamma=None,
                                           gpu_id=None, importance_type='gain',
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=None,
                                           min_child_weight=None, missing=nan,
                                           monotone_constraints=None,
                                           n_estimators=100,...
                                           random_state=None, reg_alpha=None,
                                           reg_lambda=None,
                                           scale_pos_weight=None, silent=True,
                                           subsample=None, tree_method=None,
                                           validate_parameters=None,
                                           verbosity=0),
                   param_distributions={'colsample_bytree': [0.1, 0.3, 0.5, 1],
                                        'learning_rate': [0.01, 0.03, 0.05, 0.1,
                                                          0.15, 0.2],
                                        'max_depth': [3, 5, 10, 12],
                                        'n_estimators': [100, 200, 500, 1000,
                                                         2000],
                                        'subsample': [0.1, 0.3, 0.5, 1]},
                   verbose=10)
```

In [61]:

```
# files = pd.DataFrame(data=os.listdir('byteFiles'),columns=['ID'])
# files = pd.DataFrame(data = files['ID'].str.replace('.txt',''),columns=['ID'])
# labels = result[['ID','Class']]
```

In [65]:

```
# pd.merge(files,labels,on='ID',how='inner').to_csv('bi_gram_file_label_order.csv')
```

In [67]:

```
print (random_cfl.best_params_)
```

```
{'subsample': 0.5, 'n_estimators': 200, 'max_depth': 10, 'learning_rate': 0.05, 'colsample_bytree': 0.5}
```

In [71]:

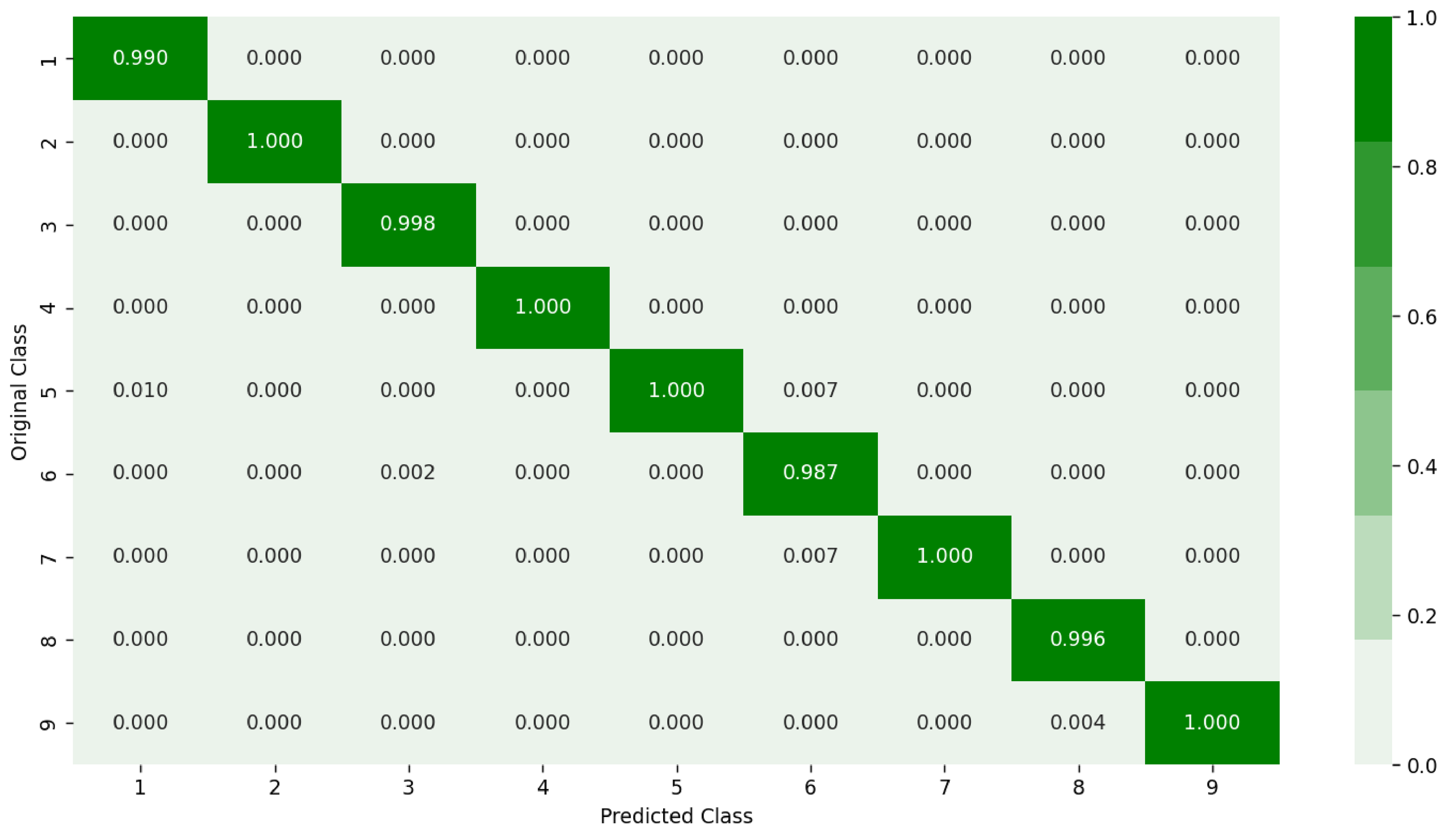
```
%%time
# Training a hyper-parameter tuned Xg-Boost regressor on our train data

# find more about XGBClassifier function here http://xgboost.readthedocs.io/en/latest/python/python_api.html?#xgboost.XGBClassifier
# -------------------------
# default paramters
# class xgboost.XGBClassifier(max_depth=3, learning_rate=0.1, n_estimators=100, silent=True,
# objective='binary:logistic', booster='gbtree', n_jobs=1, nthread=None, gamma=0, min_child_weight=1,
# max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1,
# scale_pos_weight=1, base_score=0.5, random_state=0, seed=None, missing=None, **kwargs)

# some of methods of RandomForestRegressor()
# fit(X, y, sample_weight=None, eval_set=None, eval_metric=None, early_stopping_rounds=None, verbose=True, xgb_model=None)
# get_params([deep])    Get parameters for this estimator.
# predict(data, output_margin=False, ntree_limit=0) : Predict with data. NOTE: This function is not thread safe.
# get_score(importance_type='weight') -> get the feature importance
# -----------------------
# video link2: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/what-are-ensembles/
# -----------------------
x_cfl=XGBClassifier(n_estimators=200, subsample=0.5,learning_rate=0.05, colsample_bytree=0.5, max_depth=10)
x_cfl.fit(X_train,y_train)
c_cfl=CalibratedClassifierCV(x_cfl,method='sigmoid')
c_cfl.fit(X_train,y_train)

predict_y = c_cfl.predict_proba(X_train)
print ('train loss',log_loss(y_train, predict_y))
predict_y = c_cfl.predict_proba(X_cv)

print ('cv loss',log_loss(y_cv, predict_y))
predict_y = c_cfl.predict_proba(X_test)
print ('test loss',log_loss(y_test, predict_y))
```

```
train loss 0.011415465124124244
cv loss 0.016410075115980253)
test loss 0.01475890827104327
CPU times: user 45min 36s, sys: 714 ms, total: 45min 37s
Wall time: 22min 55s
```

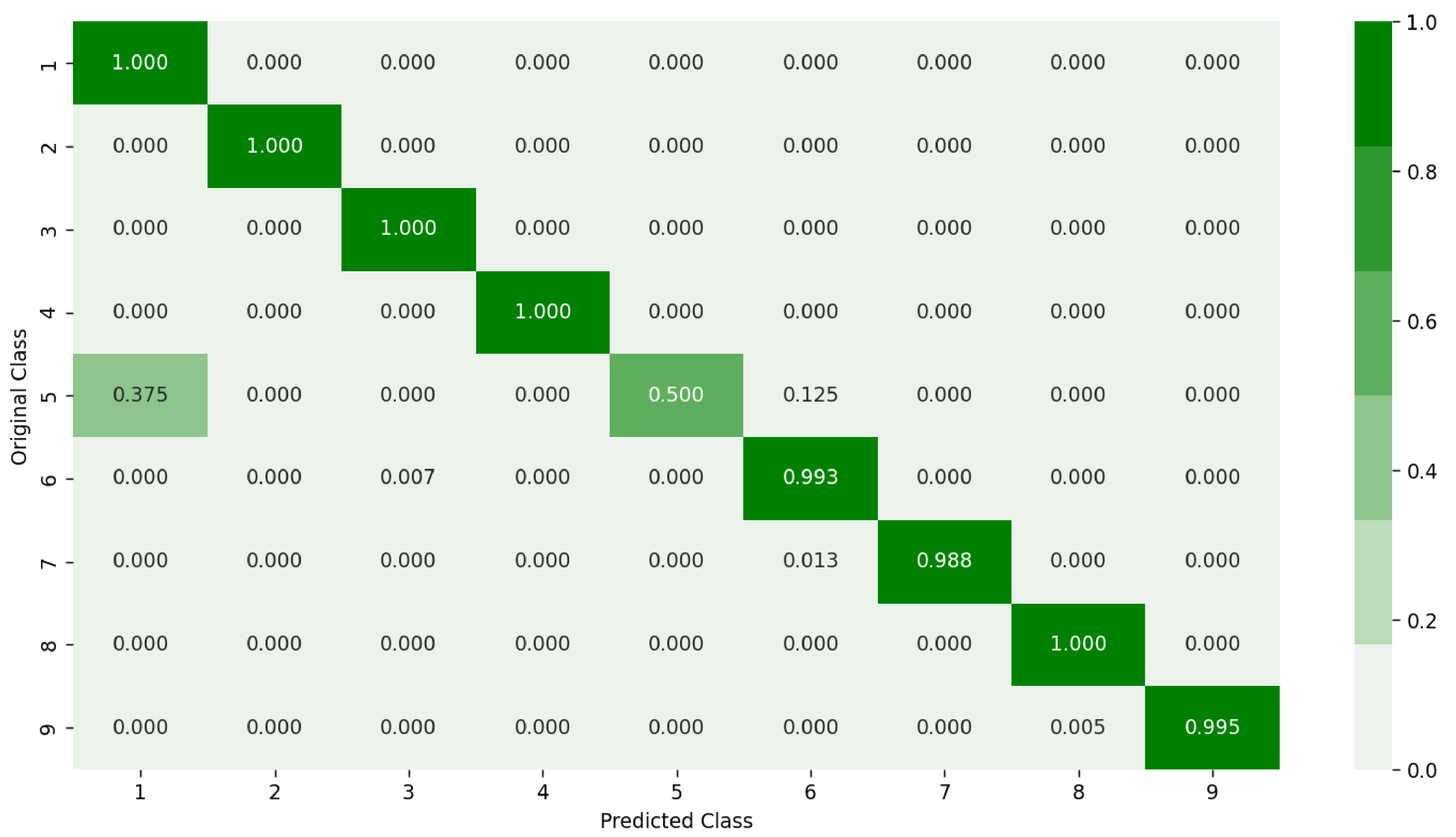
In [101]:
```python
# predict_y is test prediction
predicted_y =np.argmax(predict_y, axis=1)
plot_confusion_matrix(y_test, predicted_y+1)
```

```
Number of misclassified points  0.20198712051517942
-------------------------------------------------- Confusion matrix --------------------------------------------------

&lt;IPython.core.display.Javascript object&gt;
```


```
-------------------------------------------------- Precision matrix --------------------------------------------------

&lt;IPython.core.display.Javascript object&gt;
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
-------------------------------------------------- Recall matrix --------------------------------------------------

&lt;IPython.core.display.Javascript object&gt;
```




```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

3.3 Byte- uni + ASM - Image Features

In [85]:
```python
X_merged = pd.merge(result.drop(columns=['size']),data_50,on='ID',how='inner')
Y_merged = X_merged['Class']
X_merged = X_merged.drop(columns=['ID'])
print('shape of x',X_merged.shape,' y:',Y_merged.shape)
```

```
shape of x (10868, 759)  y: (10868,)
```

In [86]:
```python
X_merged.columns
```

Out[86]:
```
Index(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
       ...
       'ASM_990', 'ASM_991', 'ASM_992', 'ASM_993', 'ASM_994', 'ASM_995',
       'ASM_996', 'ASM_997', 'ASM_998', 'ASM_999'],
      dtype='object', length=759)
```

In [87]:
```python
### train test split
X_train, X_test, y_train, y_test = train_test_split(X_merged, Y_merged,stratify=Y_merged,test_size=0.20)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,stratify=y_train,test_size=0.20)
```

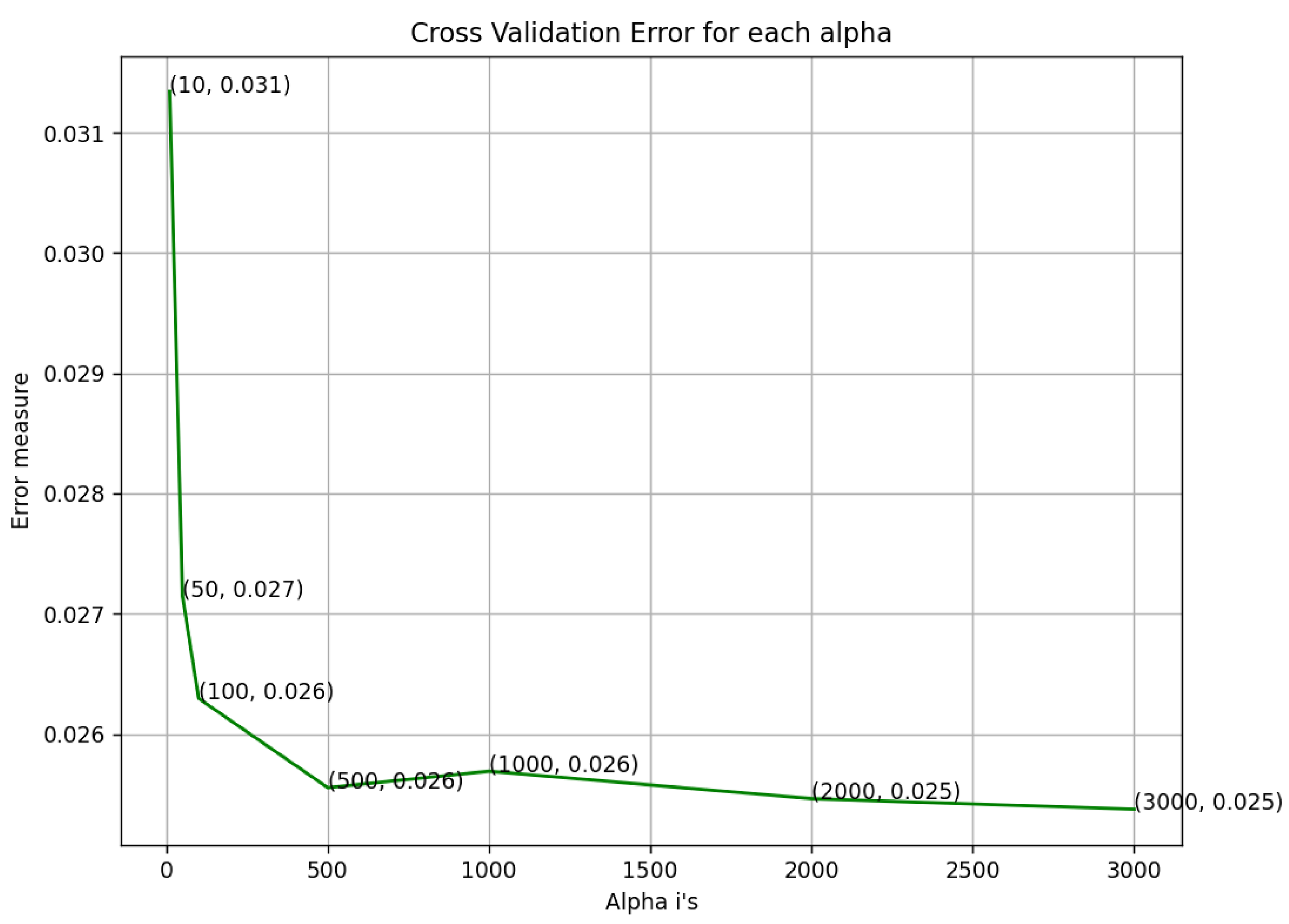
In [88]:
```python
X_train = X_train.astype(float)
X_test = X_test.astype(float)
X_cv = X_cv.astype(float)
```

3.3.1 Random Forest Classifier for ASM-Extracted image features + Byte uni gram

In [89]:

```
# --------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False,
# class_weight=None)

# Some of methods of RandomForestClassifier()
# fit(X, y, [sample_weight])    Fit the SVM model according to the given training data.
# predict(X)    Perform classification on samples in X.
# predict_proba (X) Perform classification on samples in X.

# some of attributes of  RandomForestClassifier()
# feature_importances_ : array of shape = [n_features]
# The feature importances (the higher, the more important the feature).

# --------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/random-forest-and-their-construction-2/
# --------------------------------

alpha=[10,50,100,500,1000,2000,3000]
cv_log_error_array=[]
train_log_error_array=[]
for i in alpha:
    r_cfl=RandomForestClassifier(n_estimators=i,random_state=42,n_jobs=-1)
    r_cfl.fit(X_train,y_train)
    sig_clf = CalibratedClassifierCV(r_cfl, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_cv)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=r_cfl.classes_, eps=1e-15))
    print('Parameter tuning for alpha:',i,' Loss:',log_loss(y_cv, predict_y, labels=r_cfl.classes_, eps=1e-15))

# for i in range(len(cv_log_error_array)):
#     print ('log_loss for c = ',alpha[i],'is',cv_log_error_array[i])


best_alpha = np.argmin(cv_log_error_array)

fig, ax = plt.subplots()
ax.plot(alpha, cv_log_error_array,c='g')

for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],cv_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()


r_cfl=RandomForestClassifier(n_estimators=alpha[best_alpha],random_state=42,n_jobs=-1)
r_cfl.fit(X_train,y_train)
sig_clf = CalibratedClassifierCV(r_cfl, 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))
predict_y = sig_clf.predict_proba(X_cv)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y))
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, predict_y))
```

```
Parameter tuning for alpha: 10  Loss: 0.0313426542602312
Parameter tuning for alpha: 50  Loss: 0.02714685745594213
Parameter tuning for alpha: 100  Loss: 0.026295166349999472
Parameter tuning for alpha: 500  Loss: 0.025554102636726757
Parameter tuning for alpha: 1000  Loss: 0.025690196753958717
Parameter tuning for alpha: 2000  Loss: 0.025462073418468168
Parameter tuning for alpha: 3000  Loss: 0.025374208756105795

<IPython.core.display.Javascript object>
```

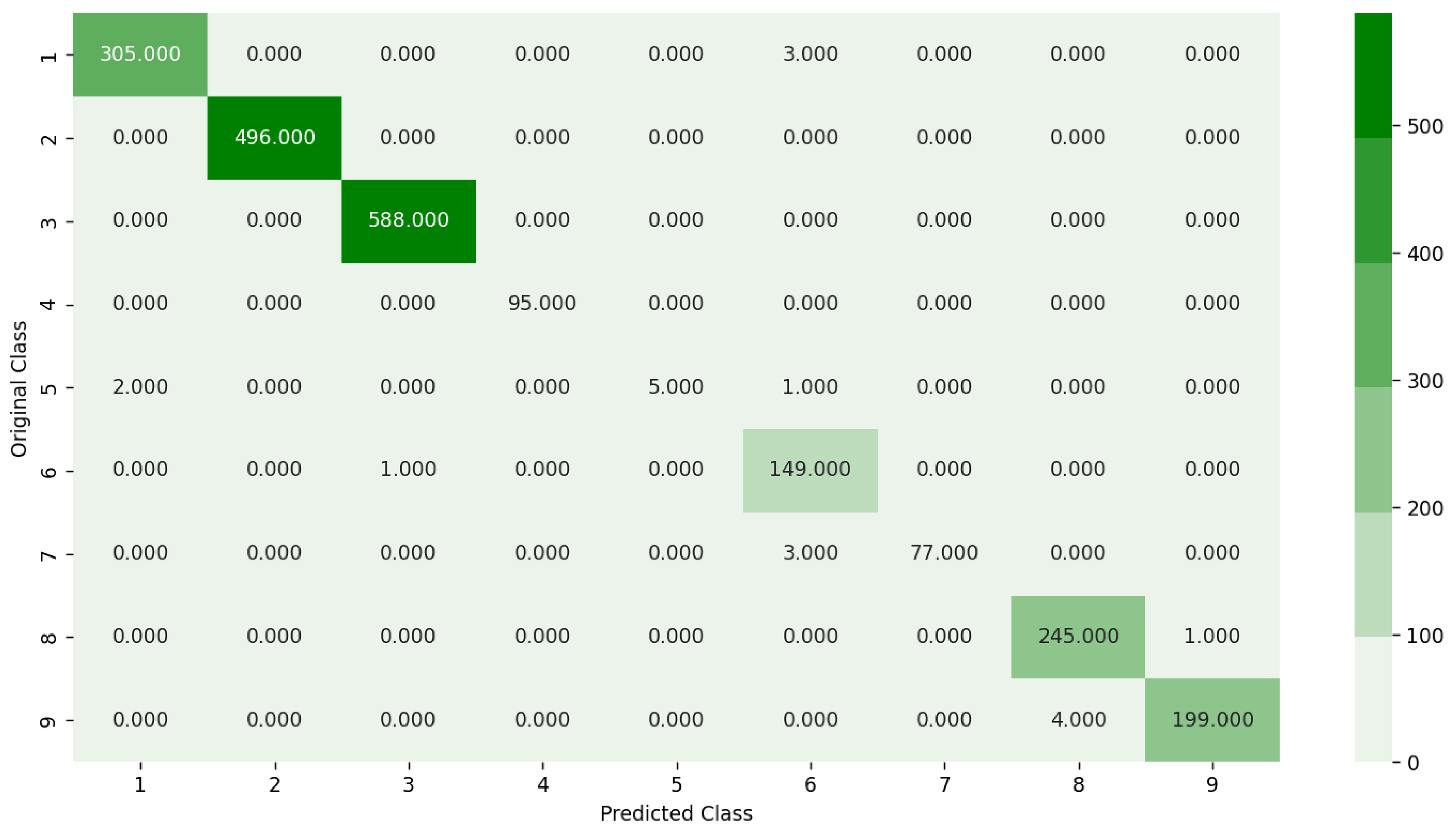



```
For values of best alpha =  3000 The train log loss is: 0.010686512051691533
For values of best alpha =  3000 The cross validation log loss is: 0.025374208756105795
For values of best alpha =  3000 The test log loss is: 0.03933044530569612
```

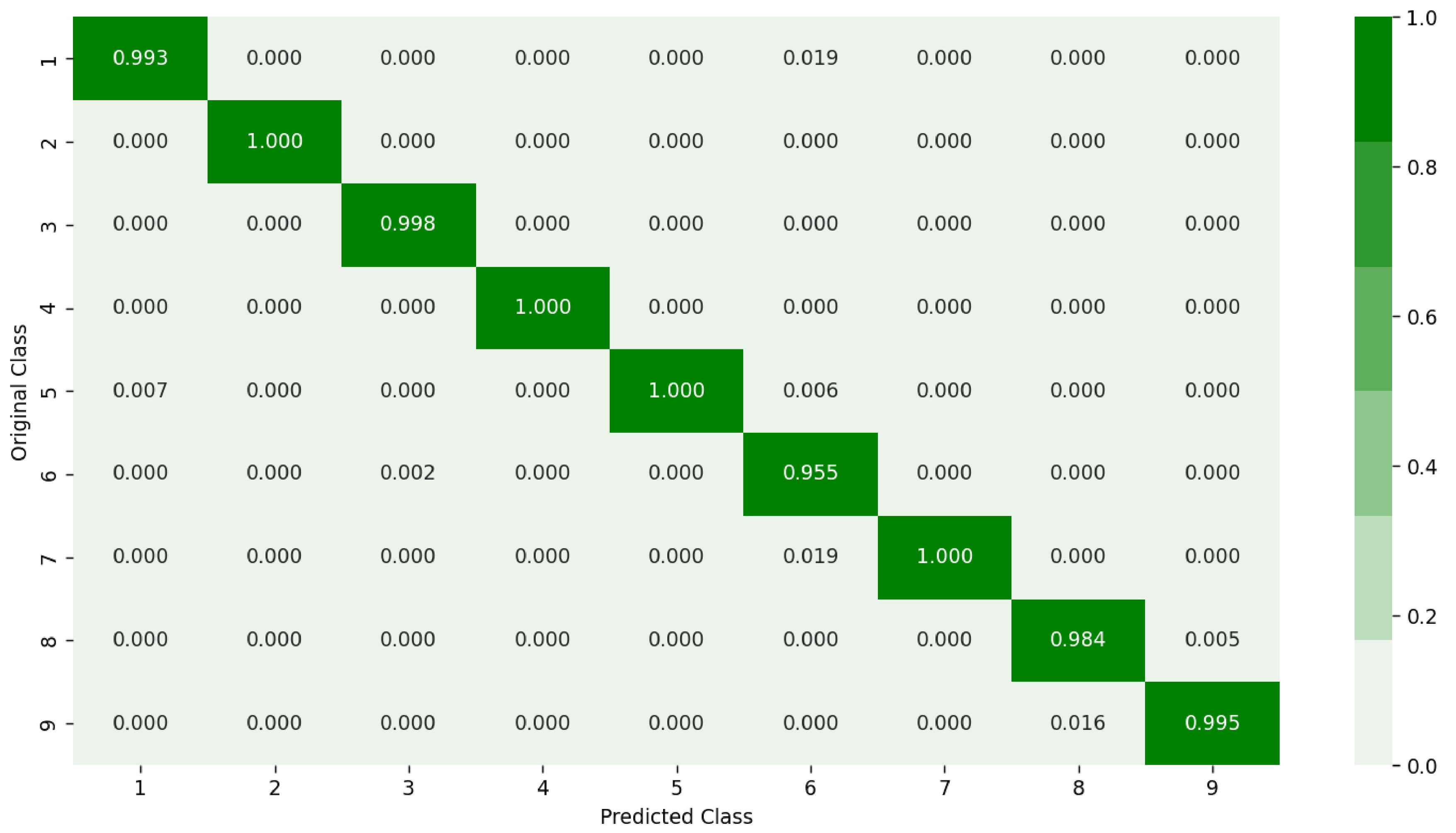
In [90]:

```
# predict_y is test prediction
predicted_y =np.argmax(predict_y, axis=1)
plot_confusion_matrix(y_test, predicted_y+1)
```

```
Number of misclassified points  0.6899724011039559
-------------------------------------------------- Confusion matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
-------------------------------------------------- Precision matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
-------------------------------------------------- Recall matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```


```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

3.3.2 XgBoost Classifier with best params for ASM-Extracted image features + Byte uni gram

In [103]:

```
%%time
x_cfl=XGBClassifier(silent=True,verbosity=0)

prams={
    'learning_rate':[0.01,0.03,0.05,0.1,0.15,0.2],
     'n_estimators':[100,200,500,1000,2000],
     'max_depth':[3,5,10,12],
    'colsample_bytree':[0.1,0.3,0.5,1],
    'subsample':[0.1,0.3,0.5,1]
}
random_cfl=RandomizedSearchCV(x_cfl,param_distributions=prams,verbose=10,cv=4)
random_cfl.fit(X_train,y_train)
```

```
Fitting 4 folds for each of 10 candidates, totalling 40 fits
[CV 1/4; 1/10] START colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1
[CV 1/4; 1/10] END colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1;, score=0.999 total time=  39.7s
[CV 2/4; 1/10] START colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1
[CV 2/4; 1/10] END colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1;, score=0.999 total time=  40.1s
[CV 3/4; 1/10] START colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1
[CV 3/4; 1/10] END colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1;, score=0.998 total time=  39.5s
[CV 4/4; 1/10] START colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1
[CV 4/4; 1/10] END colsample_bytree=0.5, learning_rate=0.2, max_depth=5, n_estimators=500, subsample=0.1;, score=1.000 total time= 1.3min
[CV 1/4; 2/10] START colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5
[CV 1/4; 2/10] END colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5;, score=0.998 total time= 2.8min
[CV 2/4; 2/10] START colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5
[CV 2/4; 2/10] END colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5;, score=1.000 total time= 2.9min
[CV 3/4; 2/10] START colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5
[CV 3/4; 2/10] END colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5;, score=0.999 total time= 2.8min
[CV 4/4; 2/10] START colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5
[CV 4/4; 2/10] END colsample_bytree=1, learning_rate=0.15, max_depth=10, n_estimators=200, subsample=0.5;, score=1.000 total time= 2.8min
[CV 1/4; 3/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3
[CV 1/4; 3/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3;, score=0.999 total time=17.1min
[CV 2/4; 3/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3
[CV 2/4; 3/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3;, score=1.000 total time=17.2min
[CV 3/4; 3/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3
[CV 3/4; 3/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3;, score=0.999 total time=17.1min
[CV 4/4; 3/10] START colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3
[CV 4/4; 3/10] END colsample_bytree=1, learning_rate=0.01, max_depth=12, n_estimators=1000, subsample=0.3;, score=1.000 total time=17.1min
[CV 1/4; 4/10] START colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5
[CV 1/4; 4/10] END colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5;, score=0.997 total time= 1.7min
[CV 2/4; 4/10] START colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5
[CV 2/4; 4/10] END colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5;, score=0.998 total time= 1.7min
[CV 3/4; 4/10] START colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5
[CV 3/4; 4/10] END colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5;, score=0.996 total time= 1.7min
[CV 4/4; 4/10] START colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5
[CV 4/4; 4/10] END colsample_bytree=0.1, learning_rate=0.03, max_depth=5, n_estimators=500, subsample=0.5;, score=0.997 total time= 1.7min
[CV 1/4; 5/10] START colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1
[CV 1/4; 5/10] END colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1;, score=0.999 total time=  29.3s
[CV 2/4; 5/10] START colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1
[CV 2/4; 5/10] END colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1;, score=0.997 total time=  29.7s
[CV 3/4; 5/10] START colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1
[CV 3/4; 5/10] END colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1;, score=0.997 total time=  29.4s
[CV 4/4; 5/10] START colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1
[CV 4/4; 5/10] END colsample_bytree=0.3, learning_rate=0.15, max_depth=12, n_estimators=200, subsample=0.1;, score=0.998 total time=  29.6s
[CV 1/4; 6/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1
[CV 1/4; 6/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1;, score=0.999 total time= 2.1min
[CV 2/4; 6/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1
[CV 2/4; 6/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1;, score=1.000 total time= 2.1min
[CV 3/4; 6/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1
[CV 3/4; 6/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1;, score=0.999 total time= 2.1min
[CV 4/4; 6/10] START colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1
[CV 4/4; 6/10] END colsample_bytree=0.3, learning_rate=0.1, max_depth=3, n_estimators=500, subsample=1;, score=0.999 total time= 2.1min
[CV 1/4; 7/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1
[CV 1/4; 7/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.995 total time= 1.4min
[CV 2/4; 7/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1
[CV 2/4; 7/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.997 total time= 1.4min
[CV 3/4; 7/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1
[CV 3/4; 7/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.994 total time= 1.4min
[CV 4/4; 7/10] START colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1
[CV 4/4; 7/10] END colsample_bytree=0.3, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.998 total time= 1.4min
[CV 1/4; 8/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1
[CV 1/4; 8/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1;, score=0.998 total time= 1.5min
[CV 2/4; 8/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1
[CV 2/4; 8/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1;, score=1.000 total time= 1.5min
[CV 3/4; 8/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1
[CV 3/4; 8/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1;, score=0.999 total time= 1.5min
[CV 4/4; 8/10] START colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1
[CV 4/4; 8/10] END colsample_bytree=0.5, learning_rate=0.15, max_depth=3, n_estimators=200, subsample=1;, score=1.000 total time= 1.5min
[CV 1/4; 9/10] START colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3
[CV 1/4; 9/10] END colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3;, score=0.994 total time= 1.6min
[CV 2/4; 9/10] START colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3
[CV 2/4; 9/10] END colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3;, score=0.992 total time= 1.6min
[CV 3/4; 9/10] START colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3
[CV 3/4; 9/10] END colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3;, score=0.989 total time= 1.6min
[CV 4/4; 9/10] START colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3
[CV 4/4; 9/10] END colsample_bytree=0.1, learning_rate=0.01, max_depth=5, n_estimators=500, subsample=0.3;, score=0.996 total time= 1.6min
[CV 1/4; 10/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1
[CV 1/4; 10/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1;, score=0.997 total time=  34.2s
[CV 2/4; 10/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1
[CV 2/4; 10/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1;, score=0.993 total time=  34.0s
[CV 3/4; 10/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1
[CV 3/4; 10/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1;, score=0.993 total time=  33.8s
[CV 4/4; 10/10] START colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1
[CV 4/4; 10/10] END colsample_bytree=0.5, learning_rate=0.05, max_depth=3, n_estimators=100, subsample=0.1;, score=0.996 total time=  34.0s
CPU times: user 4h 45min 26s, sys: 15.3 s, total: 4h 45min 41s
Wall time: 2h 23min 39s
```

Out[103]:

```
RandomizedSearchCV(cv=4,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, gamma=None,
                                           gpu_id=None, importance_type='gain',
                                           interaction_constraints=None,
                                           learning_rate=None,
                                           max_delta_step=None, max_depth=None,
                                           min_child_weight=None, missing=nan,
                                           monotone_constraints=None,
                                           n_estimators=100,...
                                           random_state=None, reg_alpha=None,
                                           reg_lambda=None,
                                           scale_pos_weight=None, silent=True,
                                           subsample=None, tree_method=None,
                                           validate_parameters=None,
                                           verbosity=0),
                   param_distributions={'colsample_bytree': [0.1, 0.3, 0.5, 1],
                                        'learning_rate': [0.01, 0.03, 0.05, 0.1,
                                                          0.15, 0.2],
                                        'max_depth': [3, 5, 10, 12],
                                        'n_estimators': [100, 200, 500, 1000,
                                                         2000],
                                        'subsample': [0.1, 0.3, 0.5, 1]},
                   verbose=10)
```

In [106]:

```
print (random_cfl.best_params_)
```

```
{'subsample': 0.3, 'n_estimators': 1000, 'max_depth': 12, 'learning_rate': 0.01, 'colsample_bytree': 1}
```

In [125]:

```
%%time
# Training a hyper-parameter tuned Xg-Boost regressor on our train data

# find more about XGBClassifier function here http://xgboost.readthedocs.io/en/latest/python/python_api.html?#xgboost.XGBClassifier
# -------------------------
# default paramters
# class xgboost.XGBClassifier(max_depth=3, learning_rate=0.1, n_estimators=100, silent=True,
# objective='binary:logistic', booster='gbtree', n_jobs=1, nthread=None, gamma=0, min_child_weight=1,
# max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1,
# scale_pos_weight=1, base_score=0.5, random_state=0, seed=None, missing=None, **kwargs)

# some of methods of RandomForestRegressor()
# fit(X, y, sample_weight=None, eval_set=None, eval_metric=None, early_stopping_rounds=None, verbose=True, xgb_model=None)
# get_params([deep])    Get parameters for this estimator.
# predict(data, output_margin=False, ntree_limit=0) : Predict with data. NOTE: This function is not thread safe.
# get_score(importance_type='weight') -> get the feature importance
# -----------------------
# video link2: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/what-are-ensembles/
# -----------------------
x_cfl=XGBClassifier(n_estimators=1000, subsample=0.3,learning_rate=0.01, colsample_bytree=1, max_depth=12)
x_cfl.fit(X_train,y_train)
c_cfl=CalibratedClassifierCV(x_cfl,method='sigmoid')
c_cfl.fit(X_train,y_train)

predict_y =random_cfl.best_estimator_.predict_proba(X_train)
print ('train loss',log_loss(y_train, predict_y))
predict_y = random_cfl.best_estimator_.predict_proba(X_cv)

print ('cv loss',log_loss(y_cv, predict_y))
predict_y = random_cfl.best_estimator_.predict_proba(X_test)
print ('test loss',log_loss(y_test, predict_y))
```

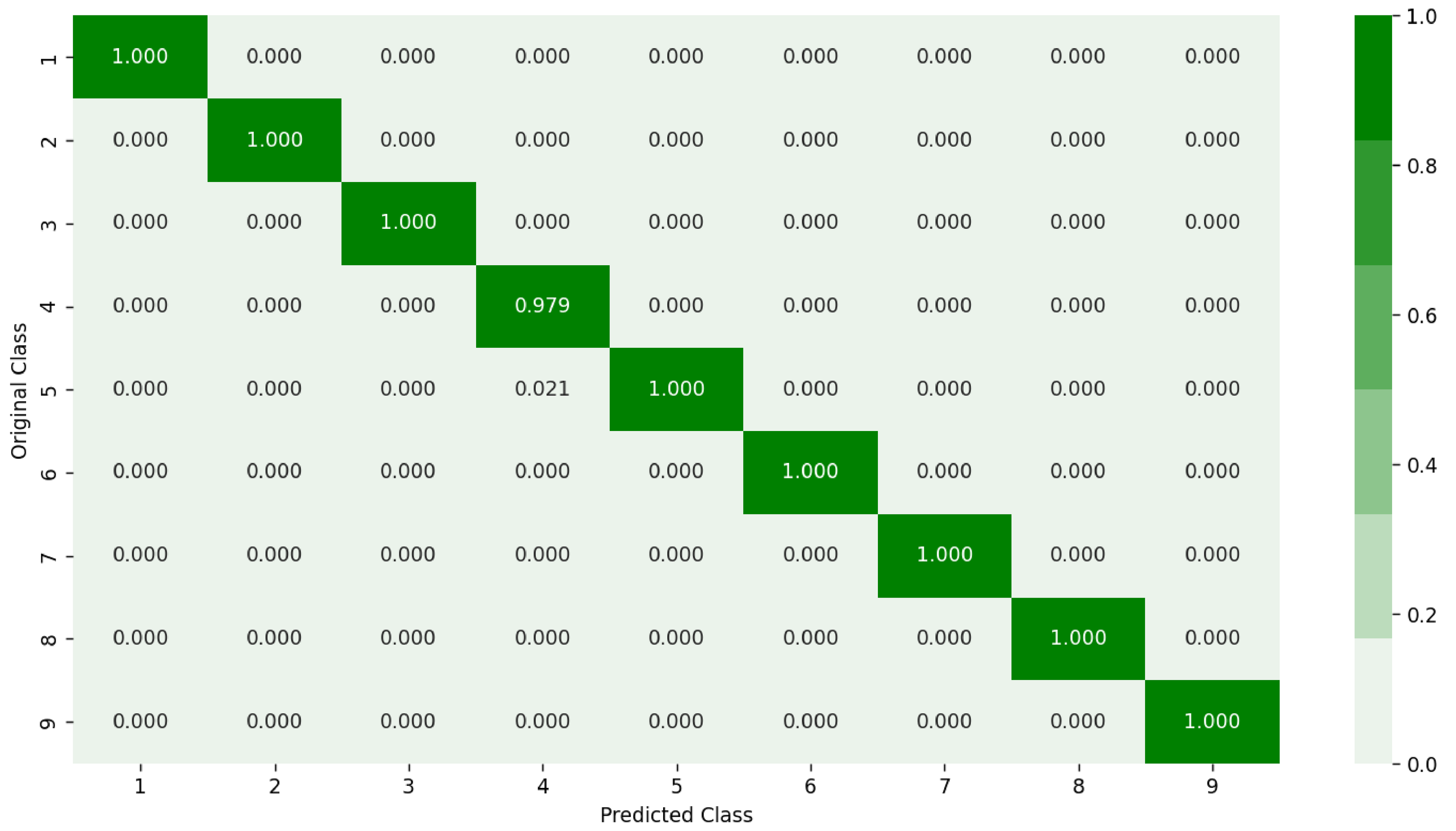
```
train loss 0.002346527399823316
cv loss 0.004251100181973078
test loss 0.007054805073482494
CPU times: user 1.58 s, sys: 5.76 ms, total: 1.59 s
Wall time: 802 ms
```

In [126]:

```
# predict_y is test prediction
predicted_y =np.argmax(predict_y, axis=1)
plot_confusion_matrix(y_test, predicted_y+1)
```

```
Number of misclassified points  -0.028003679852805885
-------------------------------------------------- Confusion matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```


```
-------------------------------------------------- Precision matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
-------------------------------------------------- Recall matrix --------------------------------------------------

<IPython.core.display.Javascript object>
```


```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

Final Summary

In [139]:

```
from prettytable import PrettyTable
table = PrettyTable()
table.field_names  = ['Model', 'Features','Best HyperParameter','Test Log Loss']

table.add_row(['Random Forest Classifier','Byte-Bigrams',500,0.0538])
table.add_row(['Random Forest Classifier','Byte-Bigrams + ASM Image Features',50,0.0206])
table.add_row(['Random Forest Classifier','ASM-unigrams + ASM Image Features',100,0.0238])
table.add_row(['Random Forest Classifier','Byte-unigrams + ASM Image Features',3000,0.0393])
table.add_row(['Random Forest Classifier','Byte-unigrams + ASM Image Features + ASM-unigrams',500,0.0241])
table.add_row(['','','',''])
table.add_row(['XGB(best_params) Classifier','Byte-Bigrams','500,max_depth=3',0.034])
table.add_row(['XGB(best_params) Classifier','Byte-Bigrams + ASM Image Features',500,0.034])
table.add_row(['XGB(best_params) Classifier','ASM-unigrams + ASM Image Features','500,max_depth=10',0.0253])
table.add_row(['XGB(best_params) Classifier','Byte-unigrams + ASM Image Features',1000,0.007])
table.add_row(['XGB(best_params) Classifier','Byte-unigrams + ASM Image Features + ASM-unigrams','200,max_depth=10',0.0147])

table.add_row(['XGB Classifier','ASM-unigrams + ASM image features',500,0.0238])

print(table)
```

```
+-----------------------------+---------------------------------------------------+---------------------+---------------+
|            Model            |                      Features                     | Best HyperParameter | Test Log Loss |
+-----------------------------+---------------------------------------------------+---------------------+---------------+
|   Random Forest Classifier  |                    Byte-Bigrams                   |         500         |     0.0538    |
|   Random Forest Classifier  |         Byte-Bigrams + ASM Image Features         |          50         |     0.0206    |
|   Random Forest Classifier  |         ASM-unigrams + ASM Image Features         |         100         |     0.0238    |
|   Random Forest Classifier  |         Byte-unigrams + ASM Image Features        |         3000        |     0.0393    |
|   Random Forest Classifier  | Byte-unigrams + ASM Image Features + ASM-unigrams |         500         |     0.0241    |
|                             |                                                   |                     |               |
| XGB(best_params) Classifier |                    Byte-Bigrams                   |   500,max_depth=3   |     0.034     |
| XGB(best_params) Classifier |         Byte-Bigrams + ASM Image Features         |         500         |      0.05     |
| XGB(best_params) Classifier |         ASM-unigrams + ASM Image Features         |   500,max_depth=10  |     0.0253    |
| XGB(best_params) Classifier |         Byte-unigrams + ASM Image Features        |         1000        |     0.007     |
| XGB(best_params) Classifier | Byte-unigrams + ASM Image Features + ASM-unigrams |   200,max_depth=10  |     0.0147    |
|        XGB Classifier       |         ASM-unigrams + ASM image features         |         500         |     0.0238    |
+-----------------------------+---------------------------------------------------+---------------------+---------------+
```

Conclusion :

Carried out the following note-book in the following sequence--

- Created byte-bigrams from the byte files and stored the generated 66000 unique keys in pickle file.
- created bi-gram datamatrix with(10868 rows,~66000 columns).
- Using Randomforest classifier performed feature selection and took top 200 bibram features.
- Created ASM image features then chi square test ,selected optimal features which stores 50% variance.
- Trained multiple models like (RandomForest,XGB) models combination of features like
  - Byte-Bigrams
  - Byte-Bigrams + ASM Image Features
  - ASM-unigrams + ASM Image Features
  - Byte-unigrams + ASM Image Features
  - Byte-unigrams + ASM Image Features + ASM-unigrams

XGB classifier with best parameters featureized using Byte-unigrams + ASM Image Features performed best on test data reducing loss to 0.007