Personalized cancer diagnosis

1. Business Problem

1.1. Description

Source: https://www.kaggle.com/c/msk-redefining-cancer-treatment/

Data: Memorial Sloan Kettering Cancer Center (MSKCC)

Download training_variants.zip and training_text.zip from Kaggle.

Context:

Source: https://www.kaggle.com/c/msk-redefining-cancer-treatment/discussion/35336#198462

Problem statement :

Classify the given genetic variations/mutations based on evidence from text-based clinical literature.

1.2. Source/Useful Links

Some articles and reference blogs about the problem statement

1. https://www.forbes.com/sites/matthewherper/2017/06/03/a-new-cancer-drug-helped-almost-everyone-who-took-it-almost-heres-what-it-teaches-us/#2a44ee2f6b25
2. https://www.youtube.com/watch?v=UwbuW7oK8rk
3. https://www.youtube.com/watch?v=qxXRKVompI8

1.3. Real-world/Business objectives and constraints.

- No low-latency requirement.
- Interpretability is important.
- Errors can be very costly.
- Probability of a data-point belonging to each class is needed.

2. Machine Learning Problem Formulation

2.1. Data

2.1.1. Data Overview

- Source: https://www.kaggle.com/c/msk-redefining-cancer-treatment/data
- We have two data files: one conatins the information about the genetic mutations and the other contains the clinical evidence (text) that human experts/pathologists use to classify the genetic mutations.
- Both these data files are have a common column called ID
- Data file's information:
  - training_variants (ID , Gene, Variations, Class)
  - training_text (ID, Text)

2.1.2. Example Data Point

training_variants

```
ID,Gene,Variation,Class
0,FAM58A,Truncating Mutations,1
1,CBL,W802*,2
2,CBL,Q249E,2
...
```

training_text

ID,Text
0||Cyclin-dependent kinases (CDKs) regulate a variety of fundamental cellular processes. CDK10 stands out as one of the last orphan CDKs for which no activating cyclin has been identified and no kinase activity revealed. Previous work has shown that CDK10 silencing increases ETS2 (v-ets erythroblastosis virus E26 oncogene homolog 2)-driven activation of the MAPK pathway, which confers tamoxifen resistance to breast cancer cells. The precise mechanisms by which CDK10 modulates ETS2 activity, and more generally the functions of CDK10, remain elusive. Here we demonstrate that CDK10 is a cyclin-dependent kinase by identifying cyclin M as an activating cyclin. Cyclin M, an orphan cyclin, is the product of FAM58A, whose mutations cause STAR syndrome, a human developmental anomaly whose features include toe syndactyly, telecanthus, and anogenital and renal malformations. We show that STAR syndrome-associated cyclin M mutants are unable to interact with CDK10. Cyclin M silencing phenocopies CDK10 silencing in increasing c-Raf and in conferring tamoxifen resistance to breast cancer cells. CDK10/cyclin M phosphorylates ETS2 in vitro, and in cells it positively controls ETS2 degradation by the proteasome. ETS2 protein levels are increased in cells derived from a STAR patient, and this increase is attributable to decreased cyclin M levels. Altogether, our results reveal an additional regulatory mechanism for ETS2, which plays key roles in cancer and development. They also shed light on the molecular mechanisms underlying STAR syndrome.Cyclin-dependent kinases (CDKs) play a pivotal role in the control of a number of fundamental cellular processes (1). The human genome contains 21 genes encoding proteins that can be considered as members of the CDK family owing to their sequence similarity with bona fide CDKs, those known to be activated by cyclins (2). Although discovered almost 20 y ago (3, 4), CDK10 remains one of the two CDKs without an identified cyclin partner. This knowledge gap has largely impeded the exploration of its biological functions. CDK10 can act as a positive cell cycle regulator in some cells (5, 6) or as a tumor suppressor in others (7, 8). CDK10 interacts with the ETS2 (v-ets erythroblastosis virus E26 oncogene homolog 2) transcription factor and inhibits its transcriptional activity through an unknown mechanism (9). CDK10 knockdown derepresses ETS2, which increases the expression of the c-Raf protein kinase, activates the MAPK pathway, and induces resistance of MCF7 cells to tamoxifen (6). ...

2.2. Mapping the real-world problem to an ML problem

2.2.1. Type of Machine Learning Problem

There are nine different classes a genetic mutation can be classified into => Multi class classification problem

2.2.2. Performance Metric

Source: https://www.kaggle.com/c/msk-redefining-cancer-treatment#evaluation

Metric(s):

- Multi class log-loss
- Confusion matrix

2.2.3. Machine Learing Objectives and Constraints

Objective: Predict the probability of each data-point belonging to each of the nine classes.

Constraints:

- Interpretability
- Class probabilities are needed.
- Penalize the errors in class probabilites => Metric is Log-loss.

- No Latency constraints.

2.3. Train, CV and Test Datasets

Split the dataset randomly into three parts train, cross validation and test with 64%,16%, 20% of data respectively

3. Exploratory Data Analysis

In [56]:

```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
from nltk.corpus import stopwords
# from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
# from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
# from imblearn.over_sampling import SMOTE
from collections import Counter
from scipy.sparse import hstack
# from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
# from sklearn.cross_validation import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
# from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier
warnings.filterwarnings("ignore")

from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression

from tqdm import tqdm_notebook
from sklearn.feature_selection import SelectKBest
```

3.1. Reading Data

3.1.1. Reading Gene and Variation Data

In [57]:

```
data = pd.read_csv('training_variants')
print('Number of data points : ', data.shape[0])
print('Number of features : ', data.shape[1])
print('Features : ', data.columns.values)
data.head()
```

```
Number of data points :  3321
Number of features :  4
Features :  ['ID' 'Gene' 'Variation' 'Class']
```

Out[57]:

	ID	Gene	Variation	Class
0	0	FAM58A	Truncating Mutations	1
1	1	CBL	W802*	2
2	2	CBL	Q249E	2
3	3	CBL	N454D	3
4	4	CBL	L399V	4

training/training_variants is a comma separated file containing the description of the genetic mutations used for training.
Fields are

- **ID :** the id of the row used to link the mutation to the clinical evidence
- **Gene :** the gene where this genetic mutation is located
- **Variation :** the aminoacid change for this mutations
- **Class :** 1-9 the class this genetic mutation has been classified on

3.1.2. Reading Text Data

In [58]:

```
# note the seprator in this file
data_text =pd.read_csv("training_text",sep="\|\|",engine="python",names=["ID","TEXT"],skiprows=1)
print('Number of data points : ', data_text.shape[0])
print('Number of features : ', data_text.shape[1])
print('Features : ', data_text.columns.values)
data_text.head()
```

```
Number of data points :  3321
Number of features :  2
Features :  ['ID' 'TEXT']
```

Out[58]:

	ID	TEXT
0	0	Cyclin-dependent kinases (CDKs) regulate a var...
1	1	Abstract Background Non-small cell lung canc...
2	2	Abstract Background Non-small cell lung canc...
3	3	Recent evidence has demonstrated that acquired...
4	4	Oncogenic mutations in the monomeric Casitas B...

3.1.3. Preprocessing of text

In [59]:

```
# loading stop words from nltk library
stop_words = set(stopwords.words('english'))

def nlp_preprocessing(total_text, index, column):
    if type(total_text) is not int:
        string = ""

        # replace every special char with space
        total_text = re.sub('[^a-zA-Z0-9\n]', ' ', total_text)
        # replace multiple spaces with single space
        total_text = re.sub('\s+',' ', total_text)

        # converting all the chars into lower-case.
        total_text = total_text.lower()

        for word in total_text.split():
        # if the word is a not a stop word then retain that word from the data
            if not word in stop_words:
                string += word + " "

        data_text[column][index] = string
```

In [60]:

```
#text processing stage.
start_time = time.clock()
for index, row in tqdm_notebook(data_text.iterrows()):
    if type(row['TEXT']) is str:
        nlp_preprocessing(row['TEXT'], index, 'TEXT')
    else:
        print("there is no text description for id:",index)
print('Time took for preprocessing the text :',time.clock() - start_time, "seconds")
```

```
there is no text description for id: 1109
there is no text description for id: 1277
there is no text description for id: 1407
there is no text description for id: 1639
there is no text description for id: 2755

Time took for preprocessing the text : 233.2920079999999 seconds
```

In [61]:

```
#merging both gene_variations and text data based on ID
result = pd.merge(data, data_text,on='ID', how='left')
result.head()
```

Out[61]:

	ID	Gene	Variation	Class	TEXT
0	0	FAM58A	Truncating Mutations	1	cyclin dependent kinases cdks regulate variety...
1	1	CBL	W802*	2	abstract background non small cell lung cancer...
2	2	CBL	Q249E	2	abstract background non small cell lung cancer...
3	3	CBL	N454D	3	recent evidence demonstrated acquired uniparen...
4	4	CBL	L399V	4	oncogenic mutations monomeric casitas b lineag...

In [62]:

```
result[result.isnull().any(axis=1)]
```

Out[62]:

	ID	Gene	Variation	Class	TEXT
1109	1109	FANCA	S1088F	1	NaN
1277	1277	ARID5B	Truncating Mutations	1	NaN
1407	1407	FGFR3	K508M	6	NaN
1639	1639	FLT1	Amplification	6	NaN
2755	2755	BRAF	G596C	7	NaN

In [63]:

In [63]:

```
result.loc[result['TEXT'].isnull(),'TEXT'] = result['Gene'] +' '+result['Variation']
```

In [64]:

```
result[result['ID']==1109]
```

Out[64]:

	ID	Gene	Variation	Class	TEXT
1109	1109	FANCA	S1088F	1	FANCA S1088F

3.1.4. Test, Train and Cross Validation Split

3.1.4.1. Splitting data into train, test and cross validation (64:20:16)

In [65]:

```
y_true = result['Class'].values
result.Gene = result.Gene.str.replace('\s+', '_')
result.Variation = result.Variation.str.replace('\s+', '_')

# split the data into test and train by maintaining same distribution of output varaible 'y_true' [stra
tify=y_true]
X_train, test_df, y_train, y_test = train_test_split(result, y_true, stratify=y_true, test_size=0.2, ra
ndom_state=1)

# split the train data into train and cross validation by maintaining same distribution of output varai
ble 'y_train' [stratify=y_train]
train_df, cv_df, y_train, y_cv = train_test_split(X_train, y_train, stratify=y_train, test_size=0.2, ra
ndom_state=1)
```

We split the data into train, test and cross validation data sets, preserving the ratio of class distribution in the original data set

In [66]:

```
print('Number of data points in train data:', train_df.shape[0])
print('Number of data points in test data:', test_df.shape[0])
print('Number of data points in cross validation data:', cv_df.shape[0])
```

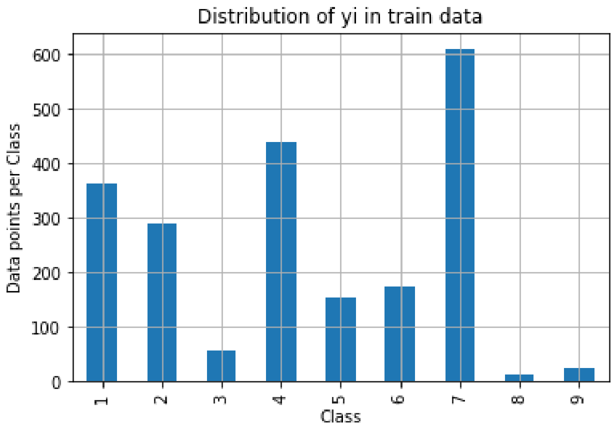
```
Number of data points in train data: 2124
Number of data points in test data: 665
Number of data points in cross validation data: 532
```

3.1.4.2. Distribution of y_i's in Train, Test and Cross Validation datasets

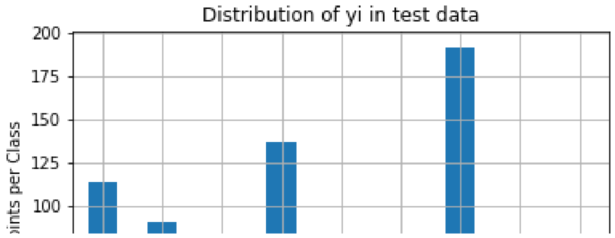
In [67]:

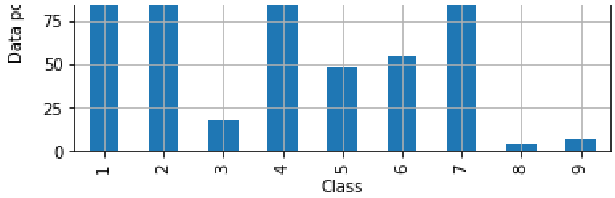
```
# it returns a dict, keys as class labels and values as the number of data points in that class
train_class_distribution = train_df['Class'].value_counts().sort_index()
test_class_distribution = test_df['Class'].value_counts().sort_index()
cv_class_distribution = cv_df['Class'].value_counts().sort_index()

my_colors = 'rgbkymc'
train_class_distribution.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in train data')
plt.grid()
plt.show()

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html
# -(train_class_distribution.values): the minus sign will give us in decreasing order
sorted_yi = np.argsort(-train_class_distribution.values)
```

```
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',train_class_distribution.values[i], '(', np.round(
(train_class_distribution.values[i]/train_df.shape[0]*100), 3), '%)')


print('-'*80)
my_colors = 'rgbkymc'
test_class_distribution.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in test data')
plt.grid()
plt.show()

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html
# -(train_class_distribution.values): the minus sign will give us in decreasing order
sorted_yi = np.argsort(-test_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',test_class_distribution.values[i], '(', np.round((
test_class_distribution.values[i]/test_df.shape[0]*100), 3), '%)')

print('-'*80)
my_colors = 'rgbkymc'
cv_class_distribution.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in cross validation data')
plt.grid()
plt.show()

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html
# -(train_class_distribution.values): the minus sign will give us in decreasing order
sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',cv_class_distribution.values[i], '(', np.round((cv
_class_distribution.values[i]/cv_df.shape[0]*100), 3), '%)')
```

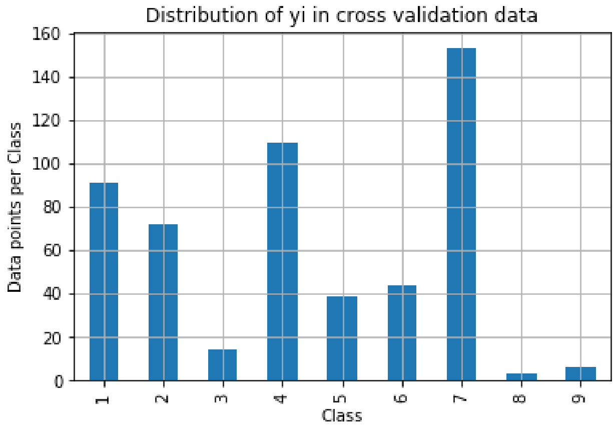


```
Number of data points in class 7 : 609 ( 28.672 %)
Number of data points in class 4 : 439 ( 20.669 %)
Number of data points in class 1 : 363 ( 17.09 %)
Number of data points in class 2 : 289 ( 13.606 %)
Number of data points in class 6 : 176 ( 8.286 %)
Number of data points in class 5 : 155 ( 7.298 %)
Number of data points in class 3 : 57 ( 2.684 %)
Number of data points in class 9 : 24 ( 1.13 %)
Number of data points in class 8 : 12 ( 0.565 %)
--------------------------------------------------------------------------------
```





```
Number of data points in class 7 : 191 ( 28.722 %)
Number of data points in class 4 : 137 ( 20.602 %)
Number of data points in class 1 : 114 ( 17.143 %)
Number of data points in class 2 : 91 ( 13.684 %)
Number of data points in class 6 : 55 ( 8.271 %)
Number of data points in class 5 : 48 ( 7.218 %)
Number of data points in class 3 : 18 ( 2.707 %)
Number of data points in class 9 : 7 ( 1.053 %)
Number of data points in class 8 : 4 ( 0.602 %)
--------------------------------------------------------------------------------
```



```
Number of data points in class 7 : 153 ( 28.759 %)
Number of data points in class 4 : 110 ( 20.677 %)
Number of data points in class 1 : 91 ( 17.105 %)
Number of data points in class 2 : 72 ( 13.534 %)
Number of data points in class 6 : 44 ( 8.271 %)
Number of data points in class 5 : 39 ( 7.331 %)
Number of data points in class 3 : 14 ( 2.632 %)
Number of data points in class 9 : 6 ( 1.128 %)
Number of data points in class 8 : 3 ( 0.564 %)
```

3.2 Prediction using a 'Random' Model

In a 'Random' Model, we generate the NINE class probabilites randomly such that they sum to 1.

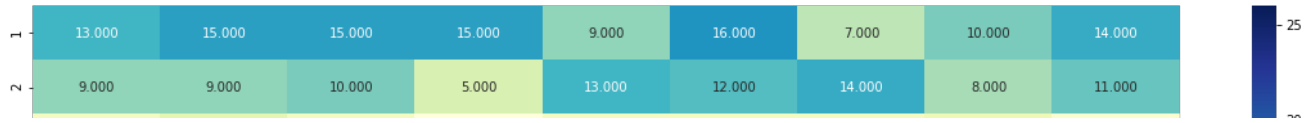
In [68]:

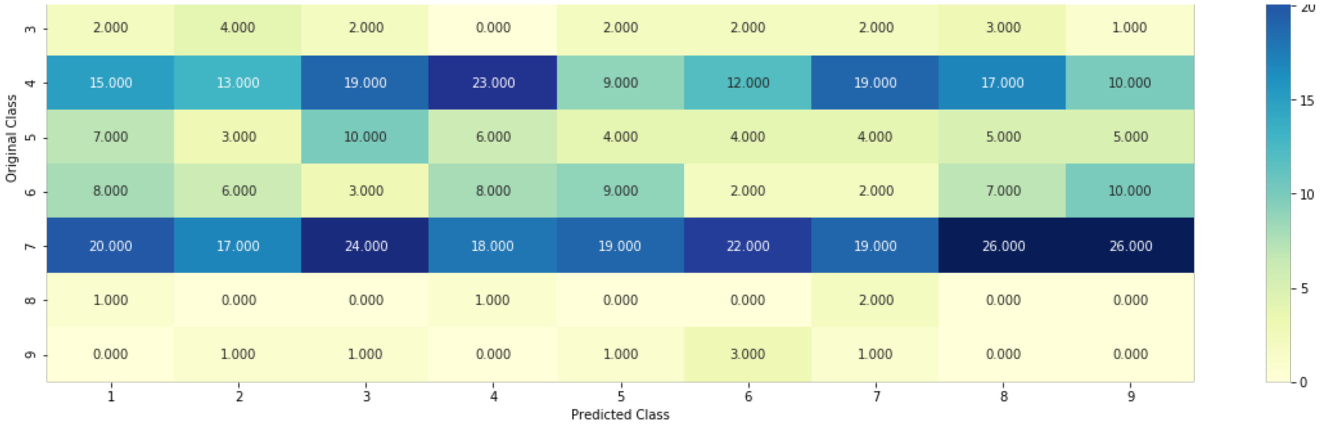
```
# This function plots the confusion matrices given y_i, y_i_hat.
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    # C = 9,9 matrix, each cell (i,j) represents number of points of class i are predicted class j
    
    A =(((C.T)/(C.sum(axis=1))).T)
    #divid each element of the confusion matrix with the sum of elements in that column
    
    # C = [[1, 2],
    #      [3, 4]]
    # C.T = [[1, 3],
    #         [2, 4]]
    # C.sum(axis = 1)  axis=0 corresonds to columns and axis=1 corresponds to rows in two diamensional
array
    # C.sum(axix =1) = [[3, 7]]
    # ((C.T)/(C.sum(axis=1))) = [[1/3, 3/7]
    #                            [2/3, 4/7]]
    # ((C.T)/(C.sum(axis=1))).T = [[1/3, 2/3]
    #                             [3/7, 4/7]]
    # sum of row elements = 1

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

    labels = [1,2,3,4,5,6,7,8,9]
    # representing A in heatmap format
    print("-"*20, "Confusion matrix", "-"*20)
    plt.figure(figsize=(20,7))
    sns.heatmap(C, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    print("-"*20, "Precision matrix (Columm Sum=1)", "-"*20)
    plt.figure(figsize=(20,7))
    sns.heatmap(B, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    # representing B in heatmap format
    print("-"*20, "Recall matrix (Row sum=1)", "-"*20)
    plt.figure(figsize=(20,7))
    sns.heatmap(A, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
```

In [69]:

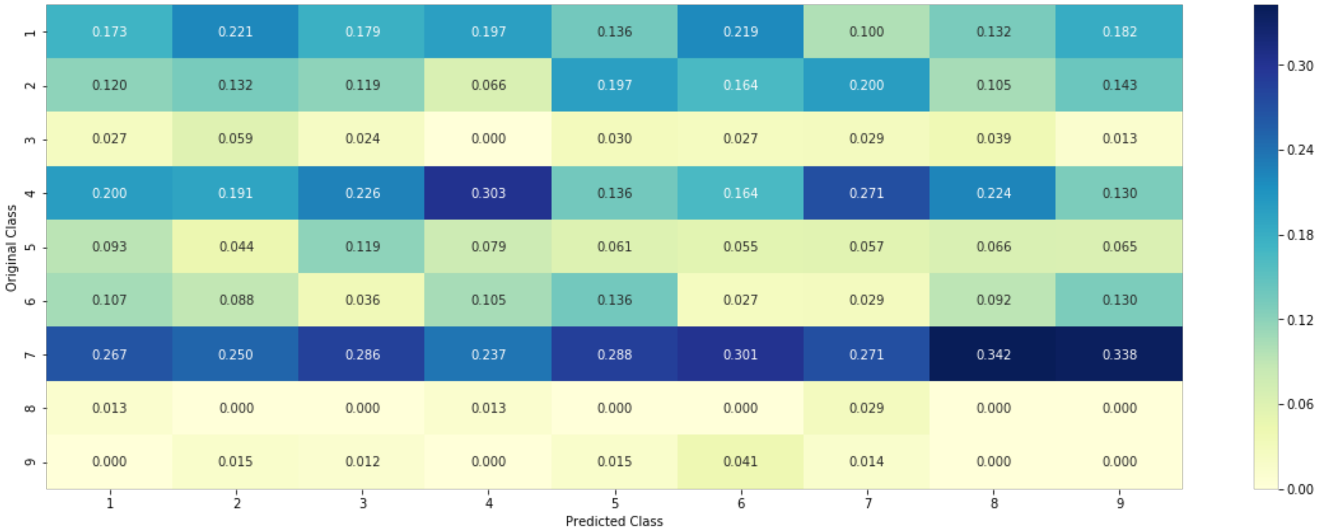
```
# we need to generate 9 numbers and the sum of numbers should be 1
# one solution is to genarate 9 numbers and divide each of the numbers by their sum
# ref: https://stackoverflow.com/a/18662466/4084039
test_data_len = test_df.shape[0]
cv_data_len = cv_df.shape[0]

# we create a output array that has exactly same size as the CV data
cv_predicted_y = np.zeros((cv_data_len,9))
for i in range(cv_data_len):
    rand_probs = np.random.rand(1,9)
    cv_predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Cross Validation Data using Random Model",log_loss(y_cv,cv_predicted_y, eps=1e-15))


# Test-Set error.
#we create a output array that has exactly same as the test data
test_predicted_y = np.zeros((test_data_len,9))
for i in range(test_data_len):
    rand_probs = np.random.rand(1,9)
    test_predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Test Data using Random Model",log_loss(y_test,test_predicted_y, eps=1e-15))

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

```
Log loss on Cross Validation Data using Random Model 2.4307298002650075
Log loss on Test Data using Random Model 2.4847505969236976
-------------------- Confusion matrix --------------------
```

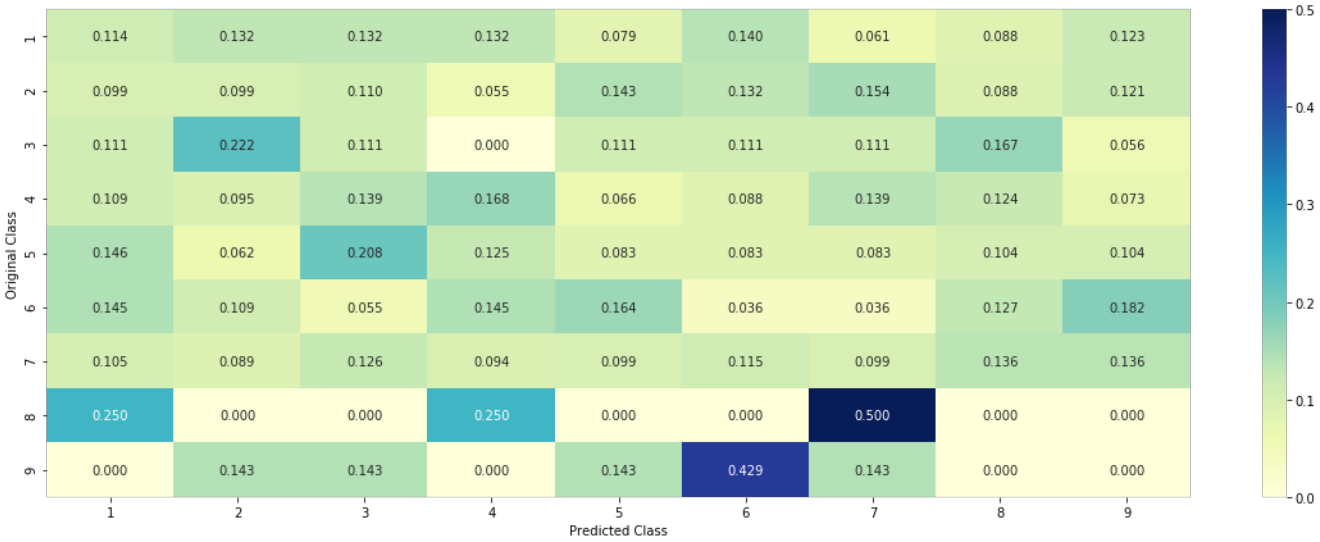




```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



```
-------------------- Recall matrix (Row sum=1) --------------------
```



3.3 Univariate Analysis

In [70]:

```
# code for response coding with Laplace smoothing.
# alpha : used for laplace smoothing
# feature: ['gene', 'variation']
# df: ['train_df', 'test_df', 'cv_df']
# algorithm
# ----------
# Consider all unique values and the number of occurances of given feature in train data dataframe
# build a vector (1*9) , the first element = (number of times it occured in class1 + 10*alpha / number 
of time it occurred in total data+90*alpha)
# gv_dict is like a look up table, for every gene it store a (1*9) representation of it
# for a value of feature in df:
# if it is in train data:
# we add the vector that was stored in 'gv_dict' look up table to 'gv_fea'
# if it is not there is train:
# we add [1/9, 1/9, 1/9, 1/9,1/9, 1/9, 1/9, 1/9, 1/9] to 'gv_fea'
# return 'gv_fea'
# ----------------------

# get_gv_fea_dict: Get Gene varaition Feature Dict
def get_gv_fea_dict(alpha, feature, df):
    # value_count: it contains a dict like
    # print(train_df['Gene'].value_counts())
    # output:
    #        {BRCA1      174
    #         TP53       106
    #         EGFR        86
    #         BRCA2       75
    #         PTEN        69
    #         KIT         61
    #         BRAF        60
    #         ERBB2       47
    #         PDGFRA      46
    #         ...}
    # print(train_df['Variation'].value_counts())
    # output:
    # {
    # Truncating_Mutations                     63
    # Deletion                                 43
    # Amplification                            43
    # Fusions                                  22
    # Overexpression                            3
    # E17K                                      3
    # Q61L                                      3
    # S222D                                     2
    # P130S                                     2
    # ...
    # }
    value_count = train_df[feature].value_counts()

    # gv_dict : Gene Variation Dict, which contains the probability array for each gene/variation
    gv_dict = dict()

    # denominator will contain the number of time that particular feature occured in whole data
    for i, denominator in value_count.items():
        # vec will contain (p(yi==1/Gi) probability of gene/variation belongs to perticular class
        # vec is 9 diamensional vector
        vec = []
        for k in range(1,10):
            # print(train_df.loc[(train_df['Class']==1) & (train_df['Gene']=='BRCA1')])
            #         ID   Gene             Variation  Class
            # 2470  2470  BRCA1                S1715C      1
            # 2486  2486  BRCA1                S1841R      1
            # 2614  2614  BRCA1                   M1R      1
            # 2432  2432  BRCA1                L1657P      1
            # 2567  2567  BRCA1                T1685A      1
            # 2583  2583  BRCA1                E1660G      1
            # 2634  2634  BRCA1                W1718L      1
            # cls_cnt.shape[0] will return the number of rows

            cls_cnt = train_df.loc[(train_df['Class']==k) & (train_df[feature]==i)]

            # cls_cnt.shape[0](numerator) will contain the number of time that particular feature occur
ed in whole data
            vec.append((cls_cnt.shape[0] + alpha*10)/ (denominator + 90*alpha))

        # we are adding the gene/variation to the dict as key and vec as value
        gv_dict[i]=vec
    return gv_dict

# Get Gene variation feature
def get_gv_feature(alpha, feature, df):
```

```
def get_gv_feature(alpha, feature, df):
    # print(gv_dict)
    #     {'BRCA1': [0.20075757575757575, 0.03787878787878788, 0.068181818181818177, 0.13636363636363
5, 0.25, 0.19318181818181818, 0.03787878787878788, 0.03787878787878788, 0.03787878787878788],
    #      'TP53': [0.32142857142857145, 0.061224489795918366, 0.061224489795918366, 0.270408163265306
5, 0.061224489795918366, 0.066326530612244902, 0.051020408163265307, 0.051020408163265307, 0.0561224489
79591837],
    #      'EGFR': [0.056818181818181816, 0.21590909090909091, 0.0625, 0.068181818181818177, 0.06818181
8181818177, 0.0625, 0.34659090909090912, 0.0625, 0.056818181818181816],
    #      'BRCA2': [0.13333333333333333, 0.060606060606060608, 0.060606060606060608, 0.078787878787878
782, 0.13939393939393939394, 0.34545454545454546, 0.060606060606060608, 0.060606060606060608, 0.06060606060
6060608],
    #      'PTEN': [0.069182389937106917, 0.062893081761006289, 0.069182389937106917, 0.46540880503144
655, 0.075471698113207544, 0.062893081761006289, 0.069182389937106917, 0.062893081761006289, 0.062893081
761006289],
    #      'KIT': [0.066225165562913912, 0.25165562913907286, 0.072847682119205295, 0.07284768211920529
5, 0.066225165562913912, 0.066225165562913912, 0.27152317880794702, 0.066225165562913912, 0.06622516556
2913912],
    #      'BRAF': [0.066666666666666666, 0.17999999999999999, 0.073333333333333334, 0.073333333333333
34, 0.093333333333333338, 0.080000000000000002, 0.29999999999999999, 0.066666666666666666, 0.0666666666
66666666],
    #     ...
    #     }
    gv_dict = get_gv_fea_dict(alpha, feature, df)
    # value_count is similar in get_gv_fea_dict
    value_count = train_df[feature].value_counts()

    # gv_fea: Gene_variation feature, it will contain the feature for each feature value in the data
    gv_fea = []
    # for every feature values in the given data frame we will check if it is there in the train data t
hen we will add the feature to gv_fea
    # if not we will add [1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9] to gv_fea
    for index, row in df.iterrows():
        if row[feature] in dict(value_count).keys():
            gv_fea.append(gv_dict[row[feature]])
        else:
            gv_fea.append([1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9])
#             gv_fea.append([-1,-1,-1,-1,-1,-1,-1,-1,-1])
    return gv_fea
```

when we caculate the probability of a feature belongs to any particular class, we apply laplace smoothing

- (numerator + 10*alpha) / (denominator + 90*alpha)

3.2.1 Univariate Analysis on Gene Feature

Q1. Gene, What type of feature it is ?

Ans. Gene is a categorical variable

Q2. How many categories are there and How they are distributed?

In [71]:

```
unique_genes = train_df['Gene'].value_counts()
print('Number of Unique Genes :', unique_genes.shape[0])
# the top 10 genes that occured most
print(unique_genes.head(10))
```

```
Number of Unique Genes : 235
BRCA1     168
TP53      111
PTEN       86
EGFR       86
BRCA2      80
BRAF       65
KIT        60
ALK        44
ERBB2      39
PDGFRA     38
Name: Gene, dtype: int64
```

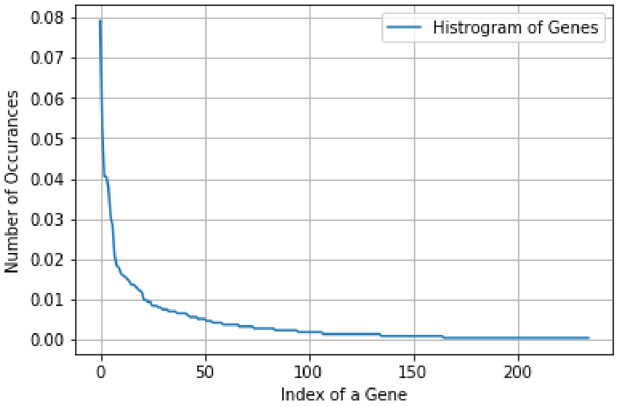
In [72]:

```
print("Ans: There are", unique_genes.shape[0] ,"different categories of genes in the train data, and th
ey are distibuted as follows",)
```

```
Ans: There are 235 different categories of genes in the train data, and they are distibuted as follows
```

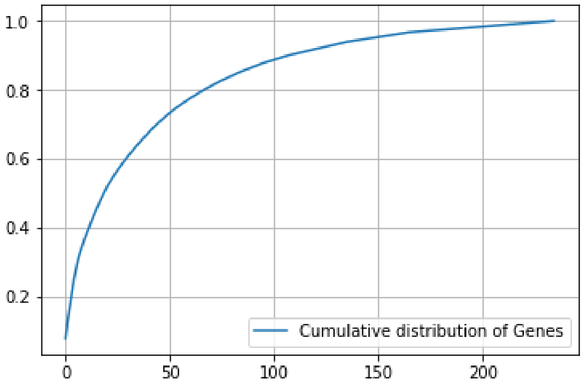
In [73]:

```
s = sum(unique_genes.values);
h = unique_genes.values/s;
plt.plot(h, label="Histrogram of Genes")
plt.xlabel('Index of a Gene')
plt.ylabel('Number of Occurances')
plt.legend()
plt.grid()
plt.show()
```



In [74]:

```
c = np.cumsum(h)
plt.plot(c,label='Cumulative distribution of Genes')
plt.grid()
plt.legend()
plt.show()
```



Q3. How to featurize this Gene feature ?

Ans.there are two ways we can featurize this variable check out this video: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/

1. One hot Encoding
2. Response coding

We will choose the appropriate featurization based on the ML model we use. For this problem of multi-class classification with categorical features, one-hot encoding is better for Logistic regression while response coding is better for Random Forests.

In [75]:

```
# Response-coding of the Gene feature

# alpha is used for laplace smoothing
alpha = 1

# train gene feature
train_gene_feature_responseCoding = np.array(get_gv_feature(alpha, "Gene", train_df))
# test gene feature
test_gene_feature_responseCoding = np.array(get_gv_feature(alpha, "Gene", test_df))
# cross validation gene feature
cv_gene_feature_responseCoding = np.array(get_gv_feature(alpha, "Gene", cv_df))
```

In [76]:

```
print("train_gene_feature_responseCoding is converted feature using respone coding method. The shape of
gene feature:", train_gene_feature_responseCoding.shape)
```

```
train_gene_feature_responseCoding is converted feature using respone coding method. The shape of gene f
eature: (2124, 9)
```

In [77]:

```
# one-hot encoding of Gene feature.
gene_vectorizer = CountVectorizer()
train_gene_feature_onehotCoding = gene_vectorizer.fit_transform(train_df['Gene'])
test_gene_feature_onehotCoding = gene_vectorizer.transform(test_df['Gene'])
cv_gene_feature_onehotCoding = gene_vectorizer.transform(cv_df['Gene'])
```

In [78]:

```
gene_vectorizer.get_feature_names()
```

Out[78]:

```
['abl1',
 'acvr1',
 'ago2',
 'akt1',
 'akt2',
 'akt3',
 'alk',
 'apc',
 'ar',
 'araf',
 'arid1a',
 'arid1b',
 'arid2',
 'arid5b',
 'asxl1',
 'asxl2',
 'atm',
 'atr',
 'atrx',
 'aurka',
 'axin1',
 'axl',
 'b2m',
 'bap1',
 'bard1',
 'bcl10',
 'bcl2l11',
 'bcor',
 'braf',
 'brca1',
 'brca2',
 'brd4',
 'brip1',
 'btk',
 'card11',
 'carm1',
 'casp8',
 'cbl',
 'ccnd1',
 'ccnd2',
 'ccnd3',
 'ccne1',
 'cdh1',
 'cdk12',
 'cdk4',
 'cdk6',
 'cdkn1a',
 'cdkn1b',
 'cdkn2a',
 'cdkn2b',
 'cdkn2c',
 'cebpa',
 'chek2',
 'cic',
 'crebbp',
 'ctcf',
 'ctla4',
 'ctnnb1',
 'ddr2',
 'dicer1',
 'dnmt3a',
 'dnmt3b',
 'egfr',
 'elf3',
 'ep300',
 'epas1',
 'epcam',
 'erbb2',
 'erbb3',
 'erbb4',
 'ercc2',
 'ercc3',
 'ercc4',
 'erg',
 'errfi1',
 'esr1',
 'etv1',
 'etv6',
 'ewsr1',
 'ezh2',
 'fanca',
 'fat1',
 'fbxw7',
 'fgf4',
 'fgfr1',
 'fgfr2',
 'fgfr3',
 'fgfr4',
 'flt1',
 'flt3',
 'foxa1',
 'foxl2',
 'foxp1',
 'fubp1',
 'gata3',
 'gli1',
 'gnaq',
 'gnas',
 'h3f3a',
 'hist1h1c',
 'hla',
 'hras',
 'idh1',
 'idh2',
 'igf1r',
 'ikbke',
 'ikzf1',
 'jak1',
 'jak2',
 'jun',
 'kdm5c',
 'kdm6a'
```

```
 'kdm6a',
 'kdr',
 'keap1',
 'kit',
 'kmt2a',
 'kmt2b',
 'kmt2c',
 'kmt2d',
 'knstrn',
 'kras',
 'lats2',
 'map2k1',
 'map2k2',
 'map2k4',
 'map3k1',
 'mapk1',
 'mdm2',
 'mdm4',
 'med12',
 'mef2b',
 'men1',
 'met',
 'mlh1',
 'mpl',
 'msh2',
 'msh6',
 'mtor',
 'myc',
 'mycn',
 'myd88',
 'ncor1',
 'nf1',
 'nf2',
 'nfe2l2',
 'nfkbia',
 'nkx2',
 'notch1',
 'notch2',
 'nras',
 'ntrk1',
 'ntrk2',
 'ntrk3',
 'nup93',
 'pak1',
 'pax8',
 'pbrm1',
 'pdgfra',
 'pdgfrb',
 'pik3ca',
 'pik3cb',
 'pik3cd',
 'pik3r1',
 'pik3r2',
 'pik3r3',
 'pim1',
 'pms1',
 'pms2',
 'pole',
 'ppm1d',
 'ppp2r1a',
 'ppp6c',
 'prdm1',
 'ptch1',
 'pten',
 'ptpn11',
 'ptprd',
 'ptprt',
 'rab35',
 'rac1',
 'rad21',
 'rad50',
 'rad51b',
 'rad51d',
 'raf1',
 'rara',
 'rasa1',
 'rb1',
 'rbm10'
 'rbm10',
 'ret',
 'rheb',
 'rhoa',
 'rictor',
 'rit1',
 'rnf43',
 'ros1',
 'rras2',
 'runx1',
 'rybp',
 'sdhb',
 'sdhc',
 'setd2',
 'sf3b1',
 'shoc2',
 'smad2',
 'smad3',
 'smad4',
 'smarca4',
 'smarcb1',
 'smo',
 'sox9',
 'spop',
 'src',
 'stag2',
 'stat3',
 'stk11',
 'tcf7l2',
 'tert',
 'tet1',
 'tet2',
 'tgfbr1',
 'tgfbr2',
 'tmprss2',
 'tp53',
 'tp53bp1',
 'tsc1',
 'tsc2',
 'u2af1',
 'vegfa',
 'vhl',
 'whsc1',
 'whsc1l1',
 'xpo1',
 'xrcc2',
 'yap1']
```

In [79]:

```
print("train_gene_feature_onehotCoding is converted feature using one-hot encoding method. The shape of
gene feature:", train_gene_feature_onehotCoding.shape)
```

```
train_gene_feature_onehotCoding is converted feature using one-hot encoding method. The shape of gene f
eature: (2124, 235)
```

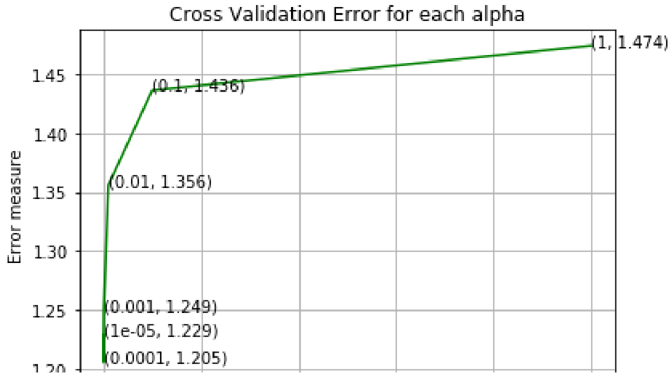
Q4. How good is this gene feature in predicting y_i?

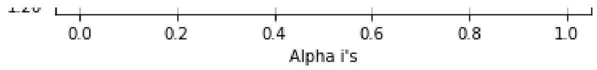
There are many ways to estimate how good a feature is, in predicting y_i. One of the good methods is to build a proper ML model using just this feature. In this case, we will build a logistic regression model using only Gene feature (one hot encoded) to predict y_i.

In [80]:

```
alpha = [10 ** x for x in range(-5, 1)] # hyperparam for SGD classifier.

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_mo
del.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=N
one, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0,
power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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


cv_log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_gene_feature_onehotCoding, y_train)

    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_gene_feature_onehotCoding, y_train)

    predict_y = sig_clf.predict_proba(cv_gene_feature_onehotCoding)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_
, eps=1e-15))

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()


best_alpha = np.argmin(cv_log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)

clf.fit(train_gene_feature_onehotCoding, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_gene_feature_onehotCoding, y_train)

predict_y = sig_clf.predict_proba(train_gene_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pred
ict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_gene_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_
cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_gene_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predic
t_y, labels=clf.classes_, eps=1e-15))
```

```
For values of alpha =  1e-05 The log loss is: 1.2285570397154246
For values of alpha =  0.0001 The log loss is: 1.205397678433504
For values of alpha =  0.001 The log loss is: 1.2491650009150361
For values of alpha =  0.01 The log loss is: 1.3556362010078462
For values of alpha =  0.1 The log loss is: 1.4361429435939435
For values of alpha =  1 The log loss is: 1.4740301432719325
```





```
For values of best alpha =  0.0001 The train log loss is: 1.004691587658098
For values of best alpha =  0.0001 The cross validation log loss is: 1.205397678433504
For values of best alpha =  0.0001 The test log loss is: 1.1962093819056512
```

Q5. Is the Gene feature stable across all the data sets (Test, Train, Cross validation)?

Ans. Yes, it is. Otherwise, the CV and Test errors would be significantly more than train error.

In [81]:

```
print("Q6. How many data points in Test and CV datasets are covered by the ", unique_genes.shape[0], " genes in train dataset?")

test_coverage=test_df[test_df['Gene'].isin(list(set(train_df['Gene'])))].shape[0]
cv_coverage=cv_df[cv_df['Gene'].isin(list(set(train_df['Gene'])))].shape[0]

print('Ans\n1. In test data',test_coverage, 'out of',test_df.shape[0], ":",(test_coverage/test_df.shape[0])*100)
print('2. In cross validation data',cv_coverage, 'out of ',cv_df.shape[0],":" ,(cv_coverage/cv_df.shape[0])*100)
```

```
Q6. How many data points in Test and CV datasets are covered by the  235  genes in train dataset?
Ans
1. In test data 644 out of 665 : 96.84210526315789
2. In cross validation data 511 out of  532 : 96.05263157894737
```

3.2.2 Univariate Analysis on Variation Feature

Q7. Variation, What type of feature is it ?

Ans. Variation is a categorical variable

Q8. How many categories are there?

In [82]:

```
unique_variations = train_df['Variation'].value_counts()
print('Number of Unique Variations :', unique_variations.shape[0])
# the top 10 variations that occured most
print(unique_variations.head(10))
```

```
Number of Unique Variations : 1920
Truncating_Mutations          58
Amplification                 51
Deletion                      45
Fusions                       25
G12V                           4
Overexpression                 3
Promoter_Hypermethylation      2
Q61K                           2
G12C                           2
A146V                          2
Name: Variation, dtype: int64
```

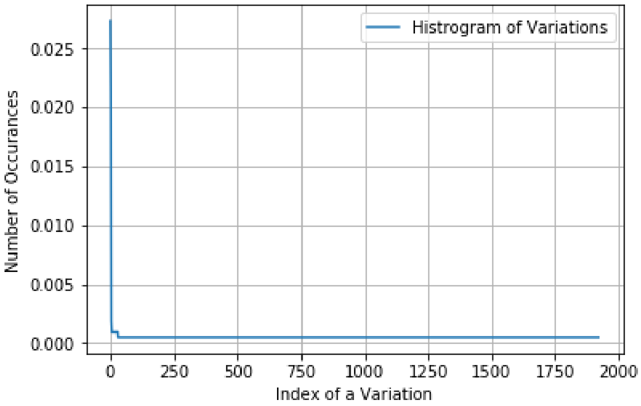
In [83]:

```
print("Ans: There are", unique_variations.shape[0] ,"different categories of variations in the train data, and they are distibuted as follows",)
```

```
Ans: There are 1920 different categories of variations in the train data, and they are distibuted as follows
```

In [84]:

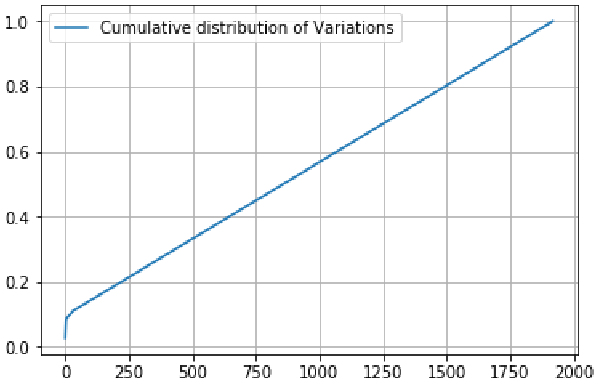
```
s = sum(unique_variations.values);
h = unique_variations.values/s;
plt.plot(h, label="Histrogram of Variations")
plt.xlabel('Index of a Variation')
plt.ylabel('Number of Occurances')
plt.legend()
plt.grid()
plt.show()
```



In [85]:

```
c = np.cumsum(h)
print(c)
plt.plot(c,label='Cumulative distribution of Variations')
plt.grid()
plt.legend()
plt.show()
```

```
[0.02730697 0.05131827 0.07250471 ... 0.99905838 0.99952919 1.        ]
```



Q9. How to featurize this Variation feature ?

Ans. There are two ways we can featurize this variable check out this video: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/

1. One hot Encoding
2. Response coding

We will be using both these methods to featurize the Variation Feature

In [86]:

```
# Response coding on 'Variation'

# alpha is used for laplace smoothing
```

```
alpha = 1
# train gene feature
train_variation_feature_responseCoding = np.array(get_gv_feature(alpha, "Variation", train_df))
# test gene feature
test_variation_feature_responseCoding = np.array(get_gv_feature(alpha, "Variation", test_df))
# cross validation gene feature
cv_variation_feature_responseCoding = np.array(get_gv_feature(alpha, "Variation", cv_df))
```

In [87]:

```
print("train_variation_feature_responseCoding is a converted feature using the response coding method. 
The shape of Variation feature:", train_variation_feature_responseCoding.shape)
```

```
train_variation_feature_responseCoding is a converted feature using the response coding method. The sha
pe of Variation feature: (2124, 9)
```

In [88]:

```
# one-hot encoding of variation feature.
variation_vectorizer = CountVectorizer()
train_variation_feature_onehotCoding = variation_vectorizer.fit_transform(train_df['Variation'])
test_variation_feature_onehotCoding = variation_vectorizer.transform(test_df['Variation'])
cv_variation_feature_onehotCoding = variation_vectorizer.transform(cv_df['Variation'])
```

In [89]:

```
print("train_variation_feature_onehotEncoded is converted feature using the onne-hot encoding method. T
he shape of Variation feature:", train_variation_feature_onehotCoding.shape)
```

```
train_variation_feature_onehotEncoded is converted feature using the onne-hot encoding method. The shap
e of Variation feature: (2124, 1948)
```

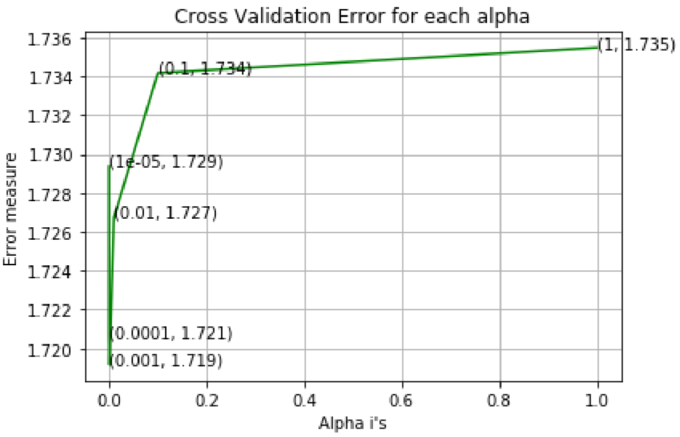
Q10. How good is this Variation feature in predicting y_i?

Let's build a model just like the earlier!

In [90]:

```
alpha = [10 ** x for x in range(-5, 1)]

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_mo
del.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=N
one, tol=None, 
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0, 
power_t=0.5, 
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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


cv_log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_variation_feature_onehotCoding, y_train)
    
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_variation_feature_onehotCoding, y_train)
    predict_y = sig_clf.predict_proba(cv_variation_feature_onehotCoding)
    
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))

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()


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

predict_y = sig_clf.predict_proba(train_variation_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_variation_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_variation_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

```
For values of alpha =  1e-05 The log loss is: 1.7293899332490221
For values of alpha =  0.0001 The log loss is: 1.7205800598416228
For values of alpha =  0.001 The log loss is: 1.719154795433691
For values of alpha =  0.01 The log loss is: 1.7267443626734562
For values of alpha =  0.1 The log loss is: 1.7341851235193924
For values of alpha =  1 The log loss is: 1.7354863846993358
```



```
For values of best alpha =  0.001 The train log loss is: 1.0984554907261728
For values of best alpha =  0.001 The cross validation log loss is: 1.719154795433691
For values of best alpha =  0.001 The test log loss is: 1.7066671376163376
```

Q11. Is the Variation feature stable across all the data sets (Test, Train, Cross validation)?

Ans. Not sure! But lets be very sure using the below analysis.

In [91]:

```
print("Q12. How many data points are covered by total ", unique_variations.shape[0], " genes in test and cross validation data sets?")
test_coverage=test_df[test_df['Variation'].isin(list(set(train_df['Variation'])))].shape[0]
cv_coverage=cv_df[cv_df['Variation'].isin(list(set(train_df['Variation'])))].shape[0]
print('Ans\n1. In test data',test_coverage, 'out of',test_df.shape[0], ":",(test_coverage/test_df.shape[0])*100)
print('2. In cross validation data',cv_coverage, 'out of ',cv_df.shape[0],":" ,(cv_coverage/cv_df.shape
[0])*100)
```

```
Q12. How many data points are covered by total  1920  genes in test and cross validation data sets?
Ans
1. In test data 61 out of 665 : 9.172932330827068
2. In cross validation data 58 out of  532 : 10.902255639097744
```

3.2.3 Univariate Analysis on Text Feature

1. How many unique words are present in train data?
2. How are word frequencies distributed?
3. How to featurize text field?
4. Is the text feature useful in predicitng y_i?
5. Is the text feature stable across train, test and CV datasets?

In [92]:

```
# cls_text is a data frame
# for every row in data fram consider the 'TEXT'
# split the words by space
# make a dict with those words
# increment its count whenever we see that word

def extract_dictionary_paddle(cls_text):
    dictionary = defaultdict(int)
    for index, row in cls_text.iterrows():
        for word in row['TEXT'].split():
            dictionary[word] +=1
    return dictionary
```

In [93]:

```
import math
#https://stackoverflow.com/a/1602964
def get_text_responsecoding(df):
    text_feature_responseCoding = np.zeros((df.shape[0],9))
    for i in range(0,9):
        row_index = 0
        for index, row in df.iterrows():
            sum_prob = 0
            for word in row['TEXT'].split():
                sum_prob += math.log(((dict_list[i].get(word,0)+10 )/(total_dict[i+1]+90)))
            text_feature_responseCoding[row_index][i] = math.exp(sum_prob/len(row['TEXT'].split()))
            row_index += 1
    return text_feature_responseCoding
```

In [94]:

```
# building a CountVectorizer with all the words that occured minimum 3 times in train data
text_vectorizer = TfidfVectorizer(min_df=3)
train_text_feature_onehotCoding = text_vectorizer.fit_transform(train_df['TEXT'])
# Select 1000 top feature
selector = SelectKBest(k=1000).fit(train_text_feature_onehotCoding, y_train)
train_text_feature_onehotCoding = selector.transform(train_text_feature_onehotCoding)

# getting all the feature names (words)
train_text_features= [text_vectorizer.get_feature_names()[i] for i in selector.get_support(indices=True
)]

# train_text_feature_onehotCoding.sum(axis=0).A1 will sum every row and returns (1*number of features)
vector
train_text_fea_counts = train_text_feature_onehotCoding.sum(axis=0).A1

# zip(list(text_features),text_fea_counts) will zip a word with its number of times it occured
text_fea_dict = dict(zip(list(train_text_features),train_text_fea_counts))


print("Total number of unique words in train data :", len(train_text_features))
```

```
Total number of unique words in train data : 1000
```

In [95]:

```
dict_list = []
# dict_list =[] contains 9 dictoinaries each corresponds to a class
for i in range(1,10):
    cls_text = train_df[train_df['Class']==i]
    # build a word dict based on the words in that class
    dict_list.append(extract_dictionary_paddle(cls_text))
    # append it to dict_list

# dict_list[i] is build on i'th  class text data
# total_dict is buid on whole training text data
total_dict = train_df.groupby('Class').count().to_dict().get('ID')


confuse_array = []
for i in train_text_features:
    ratios = []
    max_val = -1
    for j in range(0,9):
        ratios.append((dict_list[j][i]+10 )/(total_dict[j+1]+90))
    confuse_array.append(ratios)
confuse_array = np.array(confuse_array)
```

In [96]:

```
#response coding of text features
train_text_feature_responseCoding  = get_text_responsecoding(train_df)
test_text_feature_responseCoding  = get_text_responsecoding(test_df)
cv_text_feature_responseCoding  = get_text_responsecoding(cv_df)
```

In [97]:

```
# https://stackoverflow.com/a/16202486
# we convert each row values such that they sum to 1
train_text_feature_responseCoding = (train_text_feature_responseCoding.T/train_text_feature_responseCod
ing.sum(axis=1)).T
test_text_feature_responseCoding = (test_text_feature_responseCoding.T/test_text_feature_responseCoding
.sum(axis=1)).T
cv_text_feature_responseCoding = (cv_text_feature_responseCoding.T/cv_text_feature_responseCoding.sum(a
xis=1)).T
```

In [98]:

```
# don't forget to normalize every feature
train_text_feature_onehotCoding = normalize(train_text_feature_onehotCoding, axis=0)

# we use the same vectorizer that was trained on train data
test_text_feature_onehotCoding = text_vectorizer.transform(test_df['TEXT'])
test_text_feature_onehotCoding = selector.transform(test_text_feature_onehotCoding)
# don't forget to normalize every feature
test_text_feature_onehotCoding = normalize(test_text_feature_onehotCoding, axis=0)

# we use the same vectorizer that was trained on train data
cv_text_feature_onehotCoding = text_vectorizer.transform(cv_df['TEXT'])
cv_text_feature_onehotCoding = selector.transform(cv_text_feature_onehotCoding)
# don't forget to normalize every feature
cv_text_feature_onehotCoding = normalize(cv_text_feature_onehotCoding, axis=0)
```

In [99]:

```
#https://stackoverflow.com/a/2258273/4084039
sorted_text_fea_dict = dict(sorted(text_fea_dict.items(), key=lambda x: x[1] , reverse=True))
sorted_text_occur = np.array(list(sorted_text_fea_dict.values()))
```

In [100]:

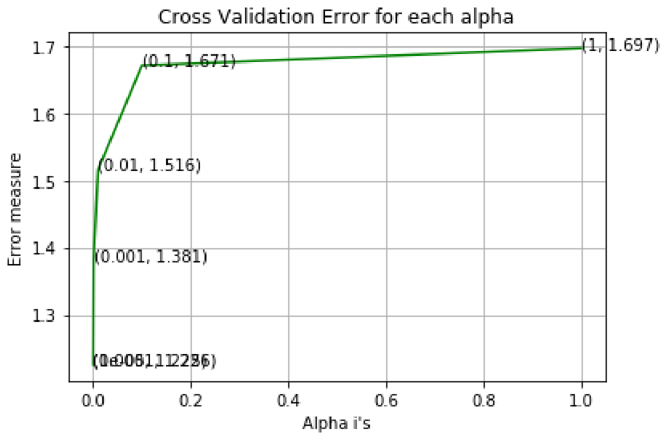
```
# Number of words for a given frequency.
print(Counter(sorted_text_occur))
```

```
Counter({0.32495870152864204: 155, 0.01652430145600372: 118, 0.022479891286147392: 49, 0.02454050835282
067: 36, 0.020245611399583485: 30, 0.019944948855856102: 16, 0.044959782572294785: 13, 0.03304860291200
744: 10, 0.08988823748945371: 9, 0.04908101670564134: 9, 0.04049122279916697: 6, 0.6499174030572841: 5,
0.45445296758139714: 4, 0.12270254176410336: 4, 0.08098244559833394: 4, 0.02133935460126388: 4, 0.14171
927979708437: 3, 0.11203101994981784: 3, 0.06073683419875044: 3, 0.039889897711712204: 3, 0.03595694990
787546: 3, 0.026318336396854616: 3, 0.02366757433404738: 3, 2.5996696122291363: 2, 1.6247935076432085:
2, 0.3731832980016789: 2, 0.11004065670464983: 2, 0.09955728112309031: 2, 0.07517606844159966: 2, 0.066
09720582401488: 2, 0.059834846567568295: 2, 0.03846278690100718: 2, 0.036767986161292496: 2, 0.03013148
8149426428: 2, 0.024593086453672117: 2, 111.35107626589098: 1, 77.53941019768409: 1, 65.84086897826612:
1, 55.51920156381058: 1, 51.80448202568179: 1, 47.368317802996906: 1, 35.76644510661583: 1, 31.91801191
8420188: 1, 28.51302890848519: 1, 28.23318822838354: 1, 28.13277823982883: 1, 27.69575026837424: 1, 26.
923304952376238: 1, 22.479977591826234: 1, 21.722225000438474: 1, 21.42219644532191: 1, 21.303226731815
993: 1, 21.224076284477608: 1, 19.780701511954078: 1, 17.62360431825005: 1, 17.489719728596274: 1, 16.4
6566487028806: 1, 16.436618977472556: 1, 15.282873965720384: 1, 14.592480616431832: 1, 14.5885607055764
29: 1, 14.453493317917397: 1, 14.16393113527651: 1, 13.378243537571857: 1, 12.447019648506943: 1, 12.20
025704585993: 1, 11.646108340548954: 1, 11.59824904084342: 1, 11.526567338382735: 1, 11.445363310586533
: 1, 11.033369044755748: 1, 10.63007269469517: 1, 10.40314990514361: 1, 9.916772895161312: 1, 9.7877955
71913598: 1, 9.262229432915346: 1, 9.205220571107265: 1, 8.905065776951865: 1, 8.663417866866201: 1, 8.
394334747849875: 1, 8.245328377640005: 1, 7.885884043802392: 1, 7.250780211320633: 1, 6.778519625456560
5: 1, 6.734679895140337: 1, 6.4839930922396425: 1, 6.430263637767948: 1, 6.364963455658964: 1, 6.105538
192386494: 1, 6.092026833706915: 1, 5.8927131708968785: 1, 5.891516668123302: 1, 5.497206212479451: 1,
5.242690946170982: 1, 5.208184493010314: 1, 5.104881467097212: 1, 5.0321551141960255: 1, 5.020905717716
182: 1, 4.9407525094191: 1, 4.926996814588912: 1, 4.867155231380508: 1, 4.843989376995893: 1, 4.7841748
55521388: 1, 4.773601169820314: 1, 4.767666106191997: 1, 4.751153037100462: 1, 4.707919418907081: 1, 4.
5706013742478415: 1, 4.4329775328706145: 1, 4.382961990975239: 1, 4.346330281895747: 1, 4.3144113175762
73: 1, 4.196023558465757: 1, 4.179584111996331: 1, 4.109676895888104: 1, 4.085748403088712: 1, 4.028658
913686374: 1, 3.9829149793323837: 1, 3.961701895042505: 1, 3.944707340171829: 1, 3.901212042488328: 1,
3.851385338464144: 1, 3.707333984568552: 1, 3.6740283554784177: 1, 3.6558539662247: 1, 3.55865429562307
: 1, 3.4884849227398544: 1, 3.4439547350549757: 1, 3.3790045066422274: 1, 3.3527531694937895: 1, 3.3222
850173271845: 1, 3.3060376252422663: 1, 3.174947804754172: 1, 3.0278524350800144: 1, 3.022118340969618:
1, 2.9789384654852973: 1, 2.9166921183952463: 1, 2.904369406222105: 1, 2.7033818938422263: 1, 2.6922424
498357134: 1, 2.6209911905846455: 1, 2.609674756157791: 1, 2.59732728312186: 1, 2.4921190569298877: 1,
2.39403243457792: 1, 2.1841981947503726: 1, 2.131860494889712: 1, 2.0912770023386233: 1, 1.989343598649
6326: 1, 1.9497522091718507: 1, 1.930470887678805: 1, 1.863374850506634: 1, 1.8159976181087298: 1, 1.80
05423316193776: 1, 1.7597815457064583: 1, 1.7408967359784453: 1, 1.7091488437416373: 1, 1.6948720733118
223: 1, 1.6870377753396377: 1, 1.6169235116072005: 1, 1.5545806926101022: 1, 1.4799080987705675: 1, 1.4
399894951162004: 1, 1.4258109912261752: 1, 1.4216331215338305: 1, 1.3972119748376042: 1, 1.395746842380
1933: 1, 1.3808524810479816: 1, 1.3159315114109051: 1, 1.3095223362737807: 1, 1.2933524675340462: 1, 1.
2919829630408062: 1, 1.2700973510471731: 1, 1.2367391112396877: 1, 1.2209115818258045: 1, 1.20117686577
16226: 1, 1.1963445555381966: 1, 1.187217348883826: 1, 1.1674672425232118: 1, 1.16737362624993: 1, 1.16
6263163389194: 1, 1.1550997677731696: 1, 1.13130227836724: 1, 1.1288633842297509: 1, 1.1195876143937802
: 1, 1.1183413530751913: 1, 1.1065196010285199: 1, 1.1058187987849994: 1, 1.0962556603612328: 1, 1.0893
921757965637: 1, 1.087197025200732: 1, 1.0688405501696134: 1, 1.0147214131706015: 1, 0.9980235452559253
: 1, 0.9825996876482519: 1, 0.977334365730696: 1, 0.9748761045859253: 1, 0.9735204554822113: 1, 0.97238
64132383582: 1, 0.9667415030422138: 1, 0.9425904874519219: 1, 0.9126826505968663: 1, 0.9032814965851792
: 1, 0.8700125149713818: 1, 0.8674835721416917: 1, 0.8653720023405089: 1, 0.8474488524787407: 1, 0.8372
238896221166: 1, 0.8333666571110749: 1, 0.8310787715567959: 1, 0.822089442642823: 1, 0.8141221172200479
: 1, 0.8118148206614072: 1, 0.804932453457258: 1, 0.7760968771270774: 1, 0.7719019274288277: 1, 0.76514
29737146924: 1, 0.7634611653579344: 1, 0.7614966544320998: 1, 0.7599607618361996: 1, 0.7554741895211549
: 1, 0.7526076102028216: 1, 0.7506765149941651: 1, 0.7504311427804414: 1, 0.7433230972823655: 1, 0.7302
357093767193: 1, 0.7274478226320484: 1, 0.7250507736673353: 1, 0.7134694412941792: 1, 0.713408500929634
9: 1, 0.7042712402381252: 1, 0.6899889210821898: 1, 0.6857854474243708: 1, 0.6840874220447599: 1, 0.679
8163349079323: 1, 0.6734122935647492: 1, 0.6622524729758896: 1, 0.6611318116810216: 1, 0.65386918844479
29: 1, 0.6525861172579679: 1, 0.6500429215628497: 1, 0.6497330509243404: 1, 0.6496473921855896: 1, 0.64
71757502135962: 1, 0.6468567206708755: 1, 0.6294369560121269: 1, 0.6165244552902106: 1, 0.6102325685993
308: 1, 0.6017050655416807: 1, 0.5823415111727014: 1, 0.5644316921148754: 1, 0.5558864932857942: 1, 0.5
453483031558654: 1, 0.534754208303897: 1, 0.5321121239957189: 1, 0.5258278314684964: 1, 0.5218137361550
665: 1, 0.513242082973655: 1, 0.5107690499448186: 1, 0.5044151443908532: 1, 0.5000544899272336: 1, 0.48
292833956197223: 1, 0.480878847319131: 1, 0.4743201582529693: 1, 0.47111697640467703: 1, 0.462670324146
40785: 1, 0.4599055472398652: 1, 0.4583404407254025: 1, 0.45717643229878213: 1, 0.45632781647327614: 1,
0.45541287937873476: 1, 0.45143892423496806: 1, 0.44695804250437154: 1, 0.4431308609459857: 1, 0.440896
28921676915: 1, 0.43756312122707375: 1, 0.42806882690799475: 1, 0.42741173823810136: 1, 0.4214511339470
661: 1, 0.4170179208215904: 1, 0.41541110698780936: 1, 0.4106364655860783: 1, 0.4062118360516669: 1, 0.
4047961032141352: 1, 0.4034236198780219: 1, 0.4033873114639696: 1, 0.3993325524730069: 1, 0.39893068536
36607: 1, 0.3982716635751502: 1, 0.39691418345650786: 1, 0.3963713688605521: 1, 0.39529423660431706: 1,
0.394495154810567: 1, 0.3914442744603375: 1, 0.3883724967870287: 1, 0.3874045422162291: 1, 0.3860252892
1286067: 1, 0.382199543237332: 1, 0.3815968013137942: 1, 0.37945187597665914: 1, 0.3789917998856425: 1,
0.37830833415100534: 1, 0.37818604450262777: 1, 0.3767484287288613: 1, 0.37175072352049326: 1, 0.371294
5944137803: 1, 0.37003076905171334: 1, 0.3672671151849012: 1, 0.365384450795828: 1, 0.36300662878631124
: 1, 0.3623896618636614: 1, 0.3619137319933302: 1, 0.36188537766255885: 1, 0.36003972840039633: 1, 0.35
95124484813945: 1, 0.35946067043627833: 1, 0.35935493776605393: 1, 0.3579014894170126: 1, 0.35786404202
25546: 1, 0.35783235206016056: 1, 0.357493805802147: 1, 0.35651203350259286: 1, 0.3564376324281159: 1,
0.3527598724174223: 1, 0.3522540693883608: 1, 0.3517839465980552: 1, 0.3517270987235007: 1, 0.351413454
14185376: 1, 0.3494827906300039: 1, 0.3487966100343703: 1, 0.34843114516649737: 1, 0.34398636541334066:
1, 0.34289837422536756: 1, 0.3424536470102653: 1, 0.3424306074898789: 1, 0.3424129953432549: 1, 0.34214
40721476366: 1, 0.3419638146999599: 1, 0.3419103655288711: 1, 0.34162161801509866: 1, 0.340963316964373
6: 1, 0.3408906999339146: 1, 0.3398277260831469: 1, 0.3395779226493497: 1, 0.3393438077963931: 1, 0.337
1983692922109: 1, 0.33526724689921855: 1, 0.3352402399204137: 1, 0.3348585035977028: 1, 0.3336986664063
511: 1, 0.3319806720458087: 1, 0.3289104869161509: 1, 0.3282382230660919: 1, 0.327762045465379: 1, 0.32
673660990083947: 1, 0.3267200221599415: 1, 0.3263211702065251: 1, 0.32596846736203655: 1, 0.32562808464
8878: 1, 0.32454491454798406: 1, 0.31902660858666876: 1, 0.31471847800606345: 1, 0.3004398684098109: 1,
0.2987597192525426: 1, 0.294486100233848: 1, 0.2918074949287253: 1, 0.2901845620400291: 1, 0.2856690981
4344366: 1, 0.28435749813947847: 1, 0.2816245500453901: 1, 0.2756498128326077: 1, 0.2699455918810274: 1
, 0.2660213050865175: 1, 0.26372056980385655: 1, 0.26198596491045634: 1, 0.25843162403054365: 1, 0.2561
741480290479: 1, 0.25054947095650143: 1, 0.25025147506038664: 1, 0.24764155198881954: 1, 0.242840530713
1449: 1, 0.24270687593602816: 1, 0.24255939257748899: 1, 0.24014379578593703: 1, 0.23968415212857946: 1
, 0.23142748993495502: 1, 0.23020292102306647: 1, 0.22954537986555218: 1, 0.2240193057692743: 1, 0.2215
320734490861: 1, 0.22086457517538602: 1, 0.2097030064462463: 1, 0.2065375423688052: 1, 0.20342987299136
606: 1, 0.20274576049628598: 1, 0.19632406682256537: 1, 0.19464723261940103: 1, 0.19399441767206987: 1,
0.1886197132897548: 1, 0.18677966842475663: 1, 0.18544038962378306: 1, 0.1821316016127788: 1, 0.1797764
7497890742: 1, 0.17727103032835423: 1, 0.1758732431652925: 1, 0.17178355846974475: 1, 0.165603381696887
74: 1, 0.16218713900330836: 1, 0.16196489119666788: 1, 0.15735923900303173: 1, 0.14776416893575825: 1,
0.14551425491283565: 1, 0.14543435345548586: 1, 0.1451755427521486: 1, 0.14126699997094247: 1, 0.135578
30178867783: 1, 0.13487934771688437: 1, 0.13458617042616516: 1, 0.13410580700266772: 1, 0.1329591046904
2305: 1, 0.13137017581825197: 1, 0.13078599289498613: 1, 0.13020485966389508: 1, 0.128271240390442: 1,
0.12270848291870817: 1, 0.12197670429588275: 1, 0.12099195143489876: 1, 0.12044032109227168: 1, 0.11875
685931214616: 1, 0.11114275191196951: 1, 0.10882027407778505: 1, 0.10677134601265283: 1, 0.101228056997
91741: 1, 0.10095962871963576: 1, 0.09832297847944632: 1, 0.09816203341128268: 1, 0.09642191161842519:
1, 0.09332641279139701: 1, 0.0916896900819318: 1, 0.08991956514458957: 1, 0.08875154857917153: 1, 0.087
08836001028308: 1, 0.08633389580636595: 1, 0.08552134093169944: 1, 0.08390513826087491: 1, 0.0828448535
1254015: 1, 0.0826215072800186: 1, 0.08093043050872745: 1, 0.08057951542863326: 1, 0.07977979542342441:
1, 0.07657342452848431: 1, 0.07556330392677577: 1, 0.073621525058462: 1, 0.07023140421916665: 1, 0.0658
2456960481248: 1, 0.06559222940498981: 1, 0.06397099861994109: 1, 0.061975623766149936: 1, 0.0613054236
6973554: 1, 0.061214184813142204: 1, 0.059495251015456024: 1, 0.05845115542357572: 1, 0.058261315240934
88: 1, 0.058253861449951254: 1, 0.05622515341461286: 1, 0.05338567300632641: 1, 0.052050467061633: 1, 0
.04957290436801115: 1, 0.04593589913770659: 1, 0.045117603706063256: 1, 0.04330384736817841: 1, 0.04267
870920252776: 1, 0.042549803490260524: 1, 0.0423218941280323: 1, 0.04201327956691802: 1, 0.040951964276
66902: 1, 0.04083675885316261: 1, 0.03989573106618683: 1, 0.03987734949069831: 1, 0.03935071125538381:
1, 0.03934239193934391: 1, 0.03645114092755358: 1, 0.036225095398325066: 1, 0.03548409453174573: 1, 0.0
33931912987503365: 1, 0.032021108798040016: 1, 0.031826612661383404: 1, 0.031595277571053154: 1, 0.0315
02723362525735: 1, 0.030201406632218562: 1, 0.02939928317797682: 1, 0.02931614538799334: 1, 0.029139803
092753457: 1, 0.02880430816931507: 1, 0.027986316061046117: 1, 0.027017586801460935: 1, 0.0267882926420
3163: 1, 0.02635573825595771: 1, 0.025683848711092774: 1, 0.025329901665905045: 1, 0.02503821850264127:
1, 0.02444955142540944: 1, 0.024257868163627867: 1, 0.02417801532728871: 1, 0.02378167543685163: 1, 0.0
23434373900727463: 1, 0.023342218888303675: 1, 0.022889764051669367: 1, 0.022059615484074323: 1, 0.0219
85756854574382: 1, 0.021947530197911333: 1, 0.02190385849460721: 1, 0.02188693339735155: 1, 0.021486461
48199694: 1, 0.01901608170328279: 1, 0.018244189284644524: 1, 0.017742047265872866: 1, 0.01416798137964
5355: 1})
```

In [101]:

```
# Train a Logistic regression+Calibration model using text features whicha re on-hot encoded
alpha = [10 ** x for x in range(-5, 1)]

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_mo
del.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=N
one, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0,
power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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


cv_log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_text_feature_onehotCoding, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_text_feature_onehotCoding, y_train)
    predict_y = sig_clf.predict_proba(cv_text_feature_onehotCoding)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_
, eps=1e-15))

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()


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

predict_y = sig_clf.predict_proba(train_text_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pred
ict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_text_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_
cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_text_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predic
t_y, labels=clf.classes_, eps=1e-15))
```

```
For values of alpha =  1e-05 The log loss is: 1.2250095032736767
For values of alpha =  0.0001 The log loss is: 1.225773759172489
For values of alpha =  0.001 The log loss is: 1.3808866065053826
For values of alpha =  0.01 The log loss is: 1.5161501194990117
For values of alpha =  0.1 The log loss is: 1.6711703351985054
For values of alpha =  1 The log loss is: 1.6969287706393494
```



```
For values of best alpha =  1e-05 The train log loss is: 1.0583946070658943
For values of best alpha =  1e-05 The cross validation log loss is: 1.2250095032736767
For values of best alpha =  1e-05 The test log loss is: 1.2503560443666308
```

Q. Is the Text feature stable across all the data sets (Test, Train, Cross validation)?

Ans. Yes, it seems like!

In [102]:

```
def get_intersec_text(df,y):
    df_text_vec = TfidfVectorizer(min_df=3)
    df_text_fea = df_text_vec.fit_transform(df['TEXT'])
    df_text_fea = df_text_vec.fit_transform(df['TEXT'])
    selector = SelectKBest(k=1000).fit(df_text_fea, y)
    df_text_fea = selector.transform(df_text_fea)
    df_text_features = [df_text_vec.get_feature_names()[i] for i in selector.get_support(indices=True)]

    df_text_fea_counts = df_text_fea.sum(axis=0).A1
    df_text_fea_dict = dict(zip(list(df_text_features),df_text_fea_counts))
    len1 = len(set(df_text_features))
    len2 = len(set(train_text_features) & set(df_text_features))
    return len1,len2
```

In [103]:

```
len1,len2 = get_intersec_text(test_df,y_test)
print(np.round((len2/len1)*100, 3), "% of word of test data appeared in train data")
len1,len2 = get_intersec_text(cv_df,y_cv)
print(np.round((len2/len1)*100, 3), "% of word of Cross Validation appeared in train data")
```

```
44.4 % of word of test data appeared in train data
36.2 % of word of Cross Validation appeared in train data
```

4. Machine Learning Models

In [104]:

```
#Data preparation for ML models.

#Misc. functionns for ML models


def predict_and_plot_confusion_matrix(train_x, train_y,test_x, test_y, clf):
    clf.fit(train_x, train_y)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x, train_y)
    pred_y = sig_clf.predict(test_x)

    # for calculating log_loss we willl provide the array of probabilities belongs to each class
    print("Log loss :",log_loss(test_y, sig_clf.predict_proba(test_x)))
    # calculating the number of data points that are misclassified
    print("Number of mis-classified points :", np.count_nonzero((pred_y- test_y))/test_y.shape[0])
    plot_confusion_matrix(test_y, pred_y)
```

In [105]:

```
def report_log_loss(train_x, train_y, test_x, test_y,  clf):
    clf.fit(train_x, train_y)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x, train_y)
    sig_clf_probs = sig_clf.predict_proba(test_x)
    return log_loss(test_y, sig_clf_probs, eps=1e-15)
```

In [117]:

```
# this function will be used just for naive bayes
# for the given indices, we will print the name of the features
# and we will check whether the feature present in the test point text or not
def get_impfeature_names(indices, text, gene, var, no_features):
    gene_count_vec = CountVectorizer()
    var_count_vec = CountVectorizer()
    text_count_vec = TfidfVectorizer(min_df=3)
    text_vec = text_count_vec.fit_transform(train_df['TEXT'])

    gene_vec = gene_count_vec.fit(train_df['Gene'])
    var_vec  = var_count_vec.fit(train_df['Variation'])
    selector = SelectKBest(k=1000).fit(text_vec,y_train)
    text_fea = [text_count_vec.get_feature_names()[i] for i in selector.get_support(indices=True)]

    all_features = gene_count_vec.get_feature_names() + var_count_vec.get_feature_names() + text_fea
    fea1_len = len(gene_vec.get_feature_names())
    fea2_len = len(var_count_vec.get_feature_names())

    word_present = 0
    for i,v in enumerate(indices):
        if (v < fea1_len):
            word = all_features[v]
            yes_no = True if word == gene else False
            if yes_no:
                word_present += 1
                print(i, "Gene feature [{}] present in test data point [{}]".format(word,yes_no))
        elif (v < fea1_len+fea2_len):
            word = all_features[v]
            yes_no = True if word == var else False
            if yes_no:
                word_present += 1
                print(i, "variation feature [{}] present in test data point [{}]".format(word,yes_no))
        else:
            word = all_features[v]
            yes_no = True if word in text.split() else False
            if yes_no:
                word_present += 1
                print(i, "Text feature [{}] present in test data point [{}]".format(word,yes_no))

    print("Out of the top ",no_features," features ", word_present, "are present in query point")
```

Stacking the three types of features

In [107]:

```
# merging gene, variance and text features

# building train, test and cross validation data sets
# a = [[1, 2],
#      [3, 4]]
# b = [[4, 5],
#      [6, 7]]
# hstack(a, b) = [[1, 2, 4, 5],
#                [ 3, 4, 6, 7]]

train_gene_var_onehotCoding = hstack((train_gene_feature_onehotCoding,train_variation_feature_onehotCod
ing))
test_gene_var_onehotCoding = hstack((test_gene_feature_onehotCoding,test_variation_feature_onehotCoding
))
cv_gene_var_onehotCoding = hstack((cv_gene_feature_onehotCoding,cv_variation_feature_onehotCoding))

train_x_onehotCoding = hstack((train_gene_var_onehotCoding, train_text_feature_onehotCoding)).tocsr()
train_y = np.array(list(train_df['Class']))

test_x_onehotCoding = hstack((test_gene_var_onehotCoding, test_text_feature_onehotCoding)).tocsr()
test_y = np.array(list(test_df['Class']))

cv_x_onehotCoding = hstack((cv_gene_var_onehotCoding, cv_text_feature_onehotCoding)).tocsr()
cv_y = np.array(list(cv_df['Class']))


train_gene_var_responseCoding = np.hstack((train_gene_feature_responseCoding,train_variation_feature_re
sponseCoding))
test_gene_var_responseCoding = np.hstack((test_gene_feature_responseCoding,test_variation_feature_respo
nseCoding))
cv_gene_var_responseCoding = np.hstack((cv_gene_feature_responseCoding,cv_variation_feature_responseCod
ing))

train_x_responseCoding = np.hstack((train_gene_var_responseCoding, train_text_feature_responseCoding))
test_x_responseCoding = np.hstack((test_gene_var_responseCoding, test_text_feature_responseCoding))
cv_x_responseCoding = np.hstack((cv_gene_var_responseCoding, cv_text_feature_responseCoding))
```

In [108]:

```
print("One hot encoding features :")
print("(number of data points * number of features) in train data = ", train_x_onehotCoding.shape)
print("(number of data points * number of features) in train data = ", train_x_onehotCoding.shape)
print("(number of data points * number of features) in test data = ", test_x_onehotCoding.shape)
print("(number of data points * number of features) in cross validation data =", cv_x_onehotCoding.shap
e)
```

```
One hot encoding features :
(number of data points * number of features) in train data =  (2124, 3183)
(number of data points * number of features) in test data =  (665, 3183)
(number of data points * number of features) in cross validation data = (532, 3183)
```

In [109]:

```
print(" Response encoding features :")
print("(number of data points * number of features) in train data = ", train_x_responseCoding.shape)
print("(number of data points * number of features) in test data = ", test_x_responseCoding.shape)
print("(number of data points * number of features) in cross validation data =", cv_x_responseCoding.sh
ape)
```

```
 Response encoding features :
(number of data points * number of features) in train data =  (2124, 27)
(number of data points * number of features) in test data =  (665, 27)
(number of data points * number of features) in cross validation data = (532, 27)
```

4.1. Base Line Model

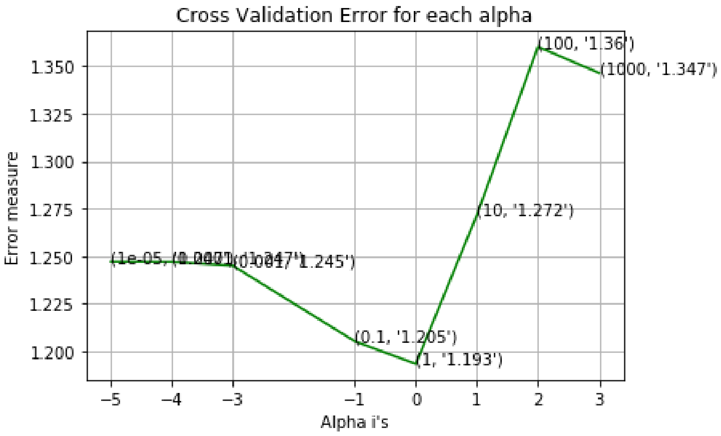
4.1.1. Naive Bayes

4.1.1.1. Hyper parameter tuning

In [110]:

```
# find more about Multinomial Naive base function here http://scikit-learn.org/stable/modules/generated
/sklearn.naive_bayes.MultinomialNB.html
# -------------------------
# default paramters
# sklearn.naive_bayes.MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)

# some of methods of MultinomialNB()
# fit(X, y[, sample_weight]) Fit Naive Bayes classifier according to X, y
# predict(X) Perform classification on an array of test vectors X.
# predict_log_proba(X) Return log-probability estimates for the test vector X.
# -----------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/naive-bayes-algor
ithm-1/
# -----------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklea
rn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
# ----------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/naive-bayes-algor
ithm-1/
# -----------------------


alpha = [0.00001, 0.0001, 0.001, 0.1, 1, 10, 100,1000]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = MultinomialNB(alpha=i)
    clf.fit(train_x_onehotCoding, train_y)

    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x_onehotCoding, train_y)

    sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
    cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    # to avoid rounding error while multiplying probabilites we use log-probability estimates
    print("Log Loss :",log_loss(cv_y, sig_clf_probs))

fig, ax = plt.subplots()
ax.plot(np.log10(alpha), cv_log_error_array,c='g')
for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[i],str(txt)), (np.log10(alpha[i]),cv_log_error_array[i]))
plt.grid()
plt.xticks(np.log10(alpha))
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(cv_log_error_array)
clf = MultinomialNB(alpha=alpha[best_alpha])
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)


predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pred
ict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_
cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predic
t_y, labels=clf.classes_, eps=1e-15))
```

```
for alpha = 1e-05
Log Loss : 1.2470622837318706
for alpha = 0.0001
Log Loss : 1.2469758464822072
for alpha = 0.001
Log Loss : 1.2449805279637534
for alpha = 0.1
Log Loss : 1.20511235500431
for alpha = 1
Log Loss : 1.1932519333426541
for alpha = 10
Log Loss : 1.271712457837876
for alpha = 100
Log Loss : 1.3603363562320845
for alpha = 1000
Log Loss : 1.3465541485505257
```



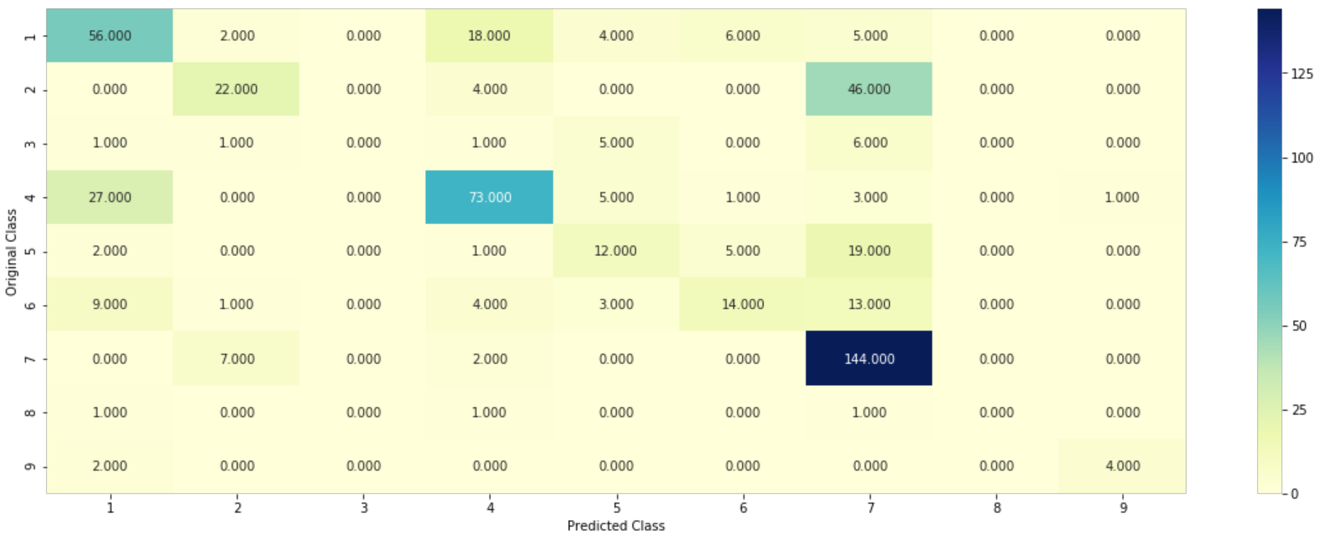
```
For values of best alpha =  1 The train log loss is: 0.8326846653519047
For values of best alpha =  1 The train log loss is: [illegible]
For values of best alpha =  1 The cross validation log loss is: 1.1932519333426541
For values of best alpha =  1 The test log loss is: 1.1779539500664262
```

4.1.1.2. Testing the model with best hyper paramters

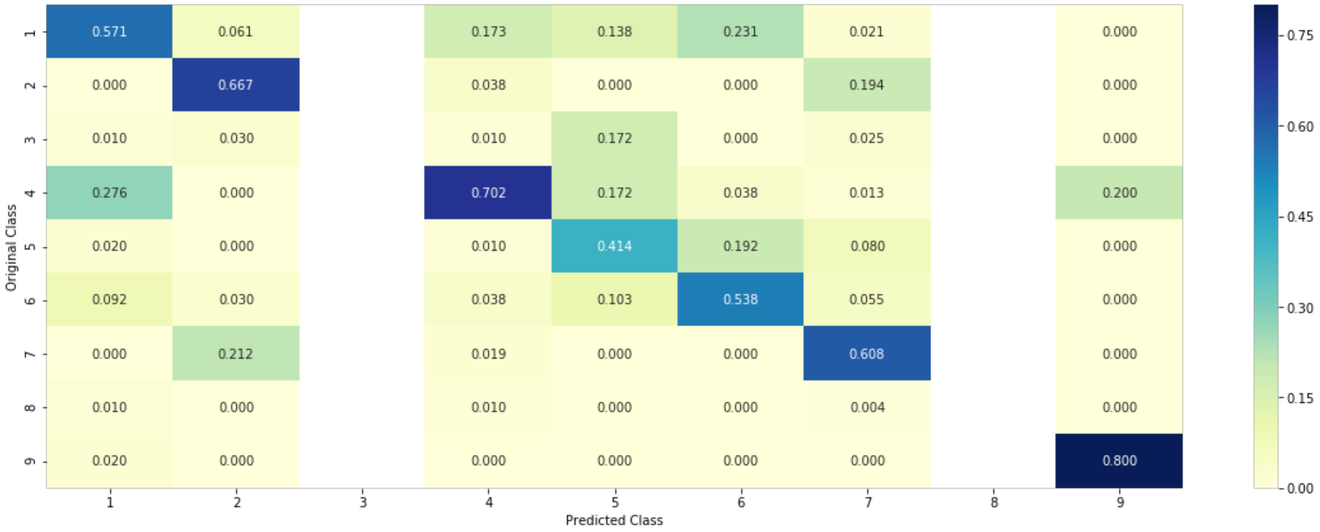
In [111]:

```
# find more about Multinomial Naive base function here http://scikit-learn.org/stable/modules/generated
/sklearn.naive_bayes.MultinomialNB.html
# -------------------------
# default paramters
# sklearn.naive_bayes.MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)

# some of methods of MultinomialNB()
# fit(X, y[, sample_weight]) Fit Naive Bayes classifier according to X, y
# predict(X) Perform classification on an array of test vectors X.
# predict_log_proba(X) Return log-probability estimates for the test vector X.
# -----------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/naive-bayes-algor
ithm-1/
# -----------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklea
rn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
# ----------------------------

clf = MultinomialNB(alpha=alpha[best_alpha])
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)
sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
# to avoid rounding error while multiplying probabilites we use log-probability estimates
print("Log Loss :",log_loss(cv_y, sig_clf_probs))
print("Number of missclassified point :", np.count_nonzero((sig_clf.predict(cv_x_onehotCoding)- cv_y))/
cv_y.shape[0])
plot_confusion_matrix(cv_y, sig_clf.predict(cv_x_onehotCoding.toarray()))
```

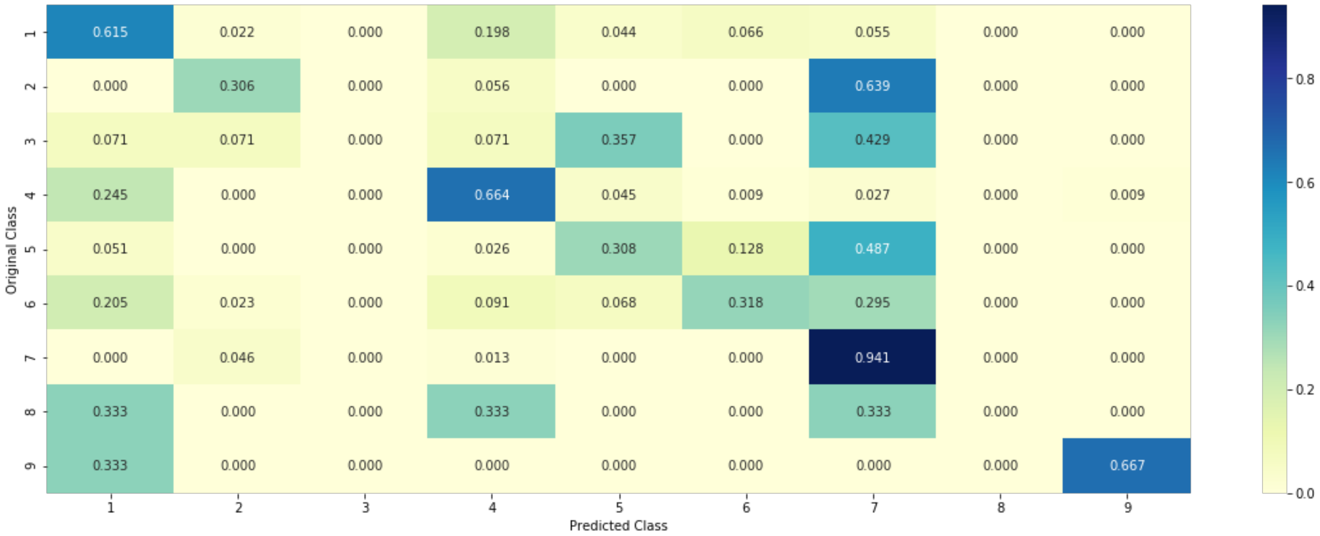
```
Log Loss : 1.1932519333426541
Number of missclassified point : 0.38909774436090223
-------------------- Confusion matrix --------------------
```



```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



```
-------------------- Recall matrix (Row sum=1) --------------------
```



4.1.1.3. Feature Importance, Correctly classified point

In [118]:

```
test_point_index = 100
no_feature = 100
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point
_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 1
Predicted Class Probabilities: [[0.6716 0.031  0.0089 0.188  0.0285 0.0383 0.0268 0.0033 0.0035]]
Actual Class : 1
--------------------------------------------------
9 Text feature [protein] present in test data point [True]
12 Text feature [function] present in test data point [True]
13 Text feature [loss] present in test data point [True]
17 Text feature [based] present in test data point [True]
```

```
18 Text feature [conserved] present in test data point [True]
19 Text feature [likely] present in test data point [True]
20 Text feature [sequence] present in test data point [True]
27 Text feature [functional] present in test data point [True]
29 Text feature [predicted] present in test data point [True]
30 Text feature [missense] present in test data point [True]
32 Text feature [assay] present in test data point [True]
34 Text feature [evidence] present in test data point [True]
35 Text feature [useful] present in test data point [True]
42 Text feature [model] present in test data point [True]
43 Text feature [26] present in test data point [True]
44 Text feature [variants] present in test data point [True]
46 Text feature [risk] present in test data point [True]
48 Text feature [family] present in test data point [True]
49 Text feature [43] present in test data point [True]
50 Text feature [predispose] present in test data point [True]
59 Text feature [basis] present in test data point [True]
60 Text feature [substitutions] present in test data point [True]
61 Text feature [treated] present in test data point [True]
66 Text feature [classified] present in test data point [True]
68 Text feature [49] present in test data point [True]
69 Text feature [brca1] present in test data point [True]
71 Text feature [patients] present in test data point [True]
76 Text feature [000] present in test data point [True]
78 Text feature [carry] present in test data point [True]
79 Text feature [conservation] present in test data point [True]
80 Text feature [57] present in test data point [True]
81 Text feature [treatment] present in test data point [True]
82 Text feature [estimated] present in test data point [True]
Out of the top  100  features  33 are present in query point
```

4.1.1.4. Feature Importance, Incorrectly classified point

In [119]:

```
test_point_index = 2
no_feature = 100
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.0852 0.0936 0.0279 0.1525 0.0598 0.0696 0.5    0.0056 0.0056]]
Actual Class : 6
--------------------------------------------------
19 Text feature [kinase] present in test data point [True]
20 Text feature [activated] present in test data point [True]
21 Text feature [downstream] present in test data point [True]
22 Text feature [inhibitor] present in test data point [True]
23 Text feature [treated] present in test data point [True]
26 Text feature [activating] present in test data point [True]
28 Text feature [treatment] present in test data point [True]
31 Text feature [patients] present in test data point [True]
35 Text feature [protein] present in test data point [True]
39 Text feature [expected] present in test data point [True]
40 Text feature [model] present in test data point [True]
41 Text feature [differential] present in test data point [True]
45 Text feature [based] present in test data point [True]
46 Text feature [26] present in test data point [True]
48 Text feature [43] present in test data point [True]
49 Text feature [likely] present in test data point [True]
56 Text feature [assay] present in test data point [True]
57 Text feature [substitutions] present in test data point [True]
58 Text feature [evidence] present in test data point [True]
59 Text feature [function] present in test data point [True]
61 Text feature [predicted] present in test data point [True]
75 Text feature [loss] present in test data point [True]
78 Text feature [combined] present in test data point [True]
```

```
76 Text feature [combined] present in test data point [True]
80 Text feature [outside] present in test data point [True]
82 Text feature [useful] present in test data point [True]
83 Text feature [assays] present in test data point [True]
Out of the top  100  features  26 are present in query point
```

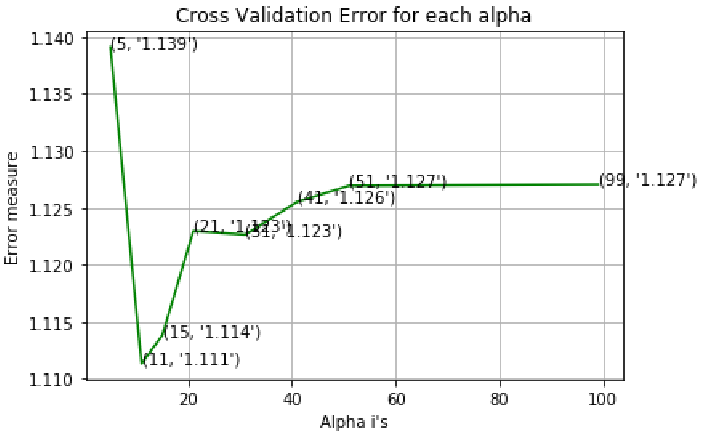
4.2. K Nearest Neighbour Classification

4.2.1. Hyper parameter tuning

In [120]:

```
# find more about KNeighborsClassifier() here http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
# -------------------------
# default parameter
# KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2,
# metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

# methods of
# fit(X, y) : Fit the model using X as training data and y as target values
# predict(X):Predict the class labels for the provided data
# predict_proba(X):Return probability estimates for the test data X.
#--------------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/k-nearest-neighbors-geometric-intuition-with-a-toy-example-1/
#--------------------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
#--------------------------------------
# video link:
#--------------------------------------


alpha = [5, 11, 15, 21, 31, 41, 51, 99]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = KNeighborsClassifier(n_neighbors=i)
    clf.fit(train_x_responseCoding, train_y)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x_responseCoding, train_y)
    sig_clf_probs = sig_clf.predict_proba(cv_x_responseCoding)
    cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    # to avoid rounding error while multiplying probabilites we use log-probability estimates
    print("Log Loss :",log_loss(cv_y, sig_clf_probs))

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],str(txt)), (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()


best_alpha = np.argmin(cv_log_error_array)
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)
sig_clf.fit(train_x_responseCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_responseCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_x_responseCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_responseCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

```
for alpha = 5
Log Loss : 1.139099679856894
for alpha = 11
Log Loss : 1.111344495549769
for alpha = 15
Log Loss : 1.1138335999233595
for alpha = 21
Log Loss : 1.1229670014029776
for alpha = 31
Log Loss : 1.1226455024911766
for alpha = 41
Log Loss : 1.1255304330843283
for alpha = 51
Log Loss : 1.1269446985863898
for alpha = 99
Log Loss : 1.127050761823645
```



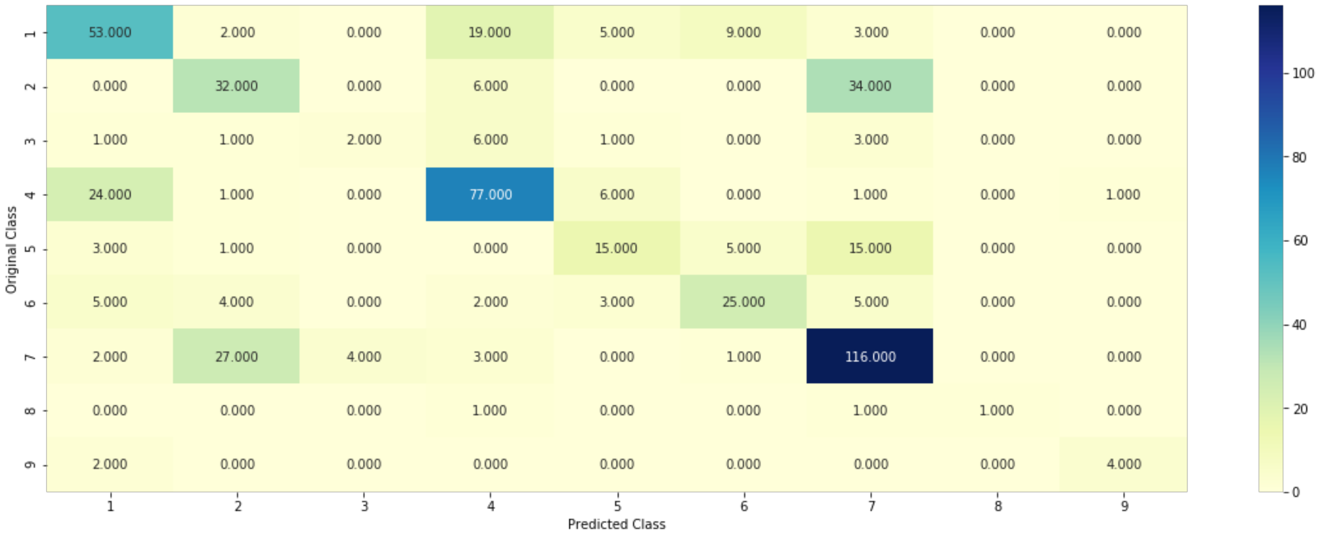
```
For values of best alpha =  11 The train log loss is: 0.6153065208699499
For values of best alpha =  11 The cross validation log loss is: 1.111344495549769
For values of best alpha =  11 The test log loss is: 1.0357413240294981
```

4.2.2. Testing the model with best hyper paramters

In [121]:

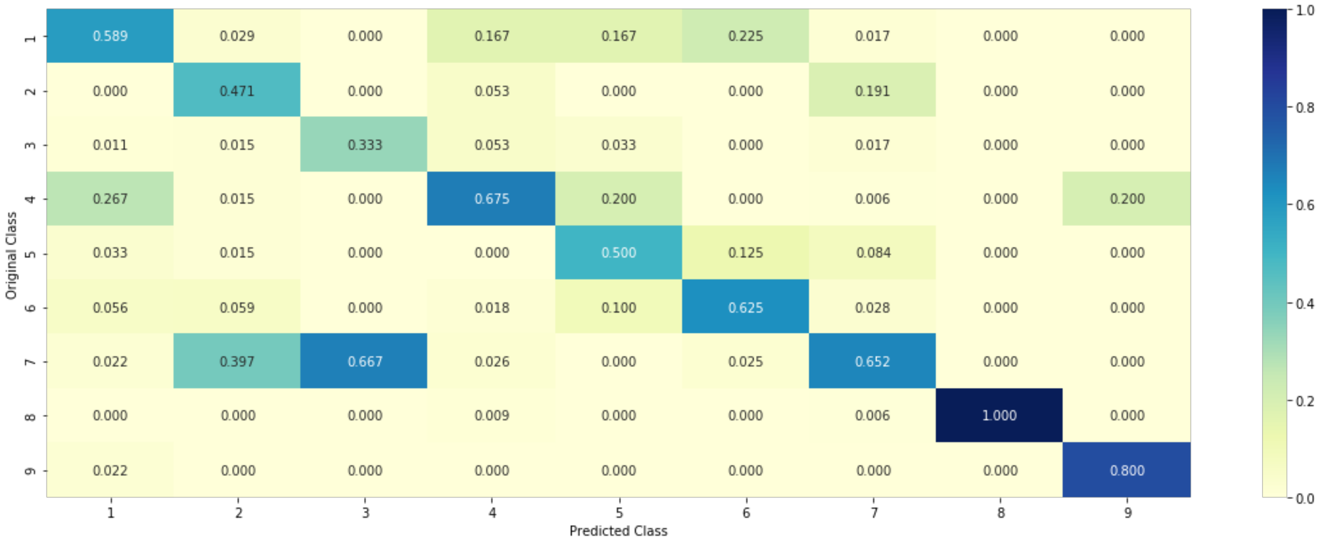
```
# find more about KNeighborsClassifier() here http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
# -------------------------
# default parameter
# KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2,
# metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

# methods of
# fit(X, y) : Fit the model using X as training data and y as target values
# predict(X):Predict the class labels for the provided data
# predict_proba(X):Return probability estimates for the test data X.
#-------------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/k-nearest-neighbors-geometric-intuition-with-a-toy-example-1/
#-------------------------------------
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
predict_and_plot_confusion_matrix(train_x_responseCoding, train_y, cv_x_responseCoding, cv_y, clf)
```

```
Log loss : 1.111344495549769
Number of mis-classified points : 0.3890977443609023
-------------------- Confusion matrix --------------------
```



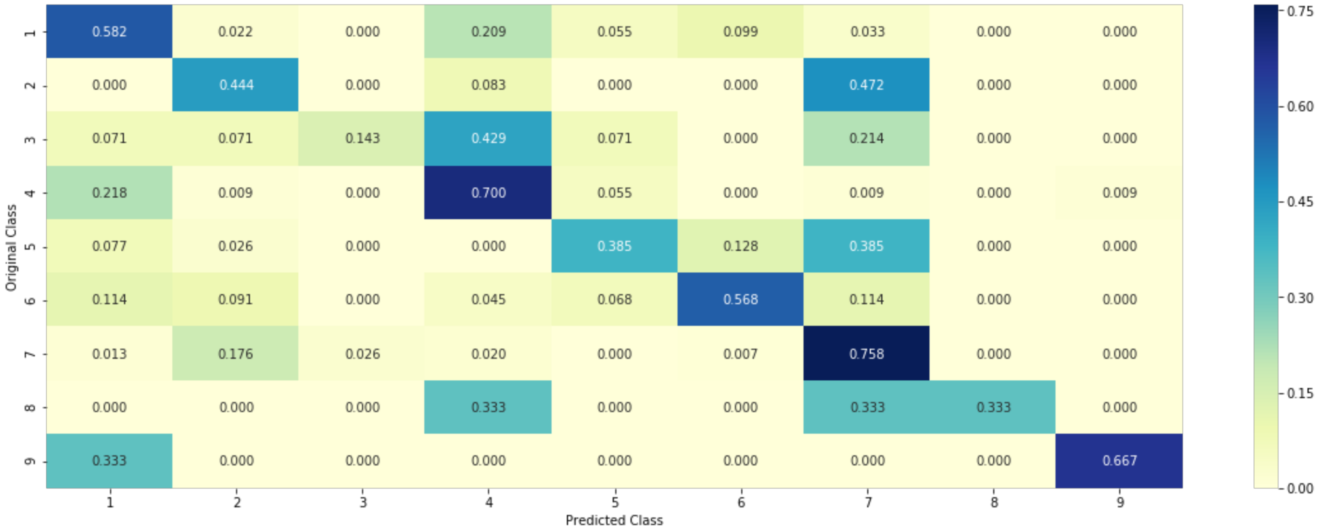
	1	2	3	4	5	6	7	8	9
1	53.000	2.000	0.000	19.000	5.000	9.000	3.000	0.000	0.000
2	0.000	32.000	0.000	6.000	0.000	0.000	34.000	0.000	0.000
3	1.000	1.000	2.000	6.000	1.000	0.000	3.000	0.000	0.000
4	24.000	1.000	0.000	77.000	6.000	0.000	1.000	0.000	1.000
5	3.000	1.000	0.000	0.000	15.000	5.000	15.000	0.000	0.000
6	5.000	4.000	0.000	2.000	3.000	25.000	5.000	0.000	0.000
7	2.000	27.000	4.000	3.000	0.000	1.000	116.000	0.000	0.000
8	0.000	0.000	0.000	1.000	0.000	0.000	1.000	1.000	0.000
9	2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	4.000

```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



	1	2	3	4	5	6	7	8	9
1	0.589	0.029	0.000	0.167	0.167	0.225	0.017	0.000	0.000
2	0.000	0.471	0.000	0.053	0.000	0.000	0.191	0.000	0.000
3	0.011	0.015	0.333	0.053	0.033	0.000	0.017	0.000	0.000
4	0.267	0.015	0.000	0.675	0.200	0.000	0.006	0.000	0.200
5	0.033	0.015	0.000	0.000	0.500	0.125	0.084	0.000	0.000
6	0.056	0.059	0.000	0.018	0.100	0.625	0.028	0.000	0.000
7	0.022	0.397	0.667	0.026	0.000	0.025	0.652	0.000	0.000
8	0.000	0.000	0.000	0.009	0.000	0.000	0.006	1.000	0.000
9	0.022	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.800

```
-------------------- Recall matrix (Row sum=1) --------------------
```



	1	2	3	4	5	6	7	8	9
1	0.582	0.022	0.000	0.209	0.055	0.099	0.033	0.000	0.000
2	0.000	0.444	0.000	0.083	0.000	0.000	0.472	0.000	0.000
3	0.071	0.071	0.143	0.429	0.071	0.000	0.214	0.000	0.000
4	0.218	0.009	0.000	0.700	0.055	0.000	0.009	0.000	0.009
5	0.077	0.026	0.000	0.000	0.385	0.128	0.385	0.000	0.000
6	0.114	0.091	0.000	0.045	0.068	0.568	0.114	0.000	0.000
7	0.013	0.176	0.026	0.020	0.000	0.007	0.758	0.000	0.000
8	0.000	0.000	0.000	0.333	0.000	0.000	0.333	0.333	0.000
9	0.333	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.667

4.2.3.Sample Query point -1

In [122]:

```
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)

test_point_index = 1
predicted_cls = sig_clf.predict(test_x_responseCoding[0].reshape(1,-1))
print("Predicted Class :", predicted_cls[0])
print("Actual Class :", test_y[test_point_index])
neighbors = clf.kneighbors(test_x_responseCoding[test_point_index].reshape(1, -1), alpha[best_alpha])
print("The ",alpha[best_alpha]," nearest neighbours of the test points belongs to classes",train_y[neig
hbors[1][0]])
print("Fequency of nearest points :",Counter(train_y[neighbors[1][0]]))
```

```
Predicted Class : 4
Actual Class : 6
The  11  nearest neighbours of the test points belongs to classes [6 6 6 6 6 6 6 6 6 6 6]
Fequency of nearest points : Counter({6: 11})
```

4.2.4. Sample Query Point-2

In [123]:

```
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)

test_point_index = 100

predicted_cls = sig_clf.predict(test_x_responseCoding[test_point_index].reshape(1,-1))
print("Predicted Class :", predicted_cls[0])
print("Actual Class :", test_y[test_point_index])
neighbors = clf.kneighbors(test_x_responseCoding[test_point_index].reshape(1, -1), alpha[best_alpha])
print("the k value for knn is",alpha[best_alpha],"and the nearest neighbours of the test points belongs
to classes",train_y[neighbors[1][0]])
print("Fequency of nearest points :",Counter(train_y[neighbors[1][0]]))
```

```
Predicted Class : 1
Actual Class : 1
the k value for knn is 11 and the nearest neighbours of the test points belongs to classes [1 1 1 1 4 1
1 1 1 4 1]
Fequency of nearest points : Counter({1: 9, 4: 2})
```

4.3. Logistic Regression

4.3.1. With Class balancing

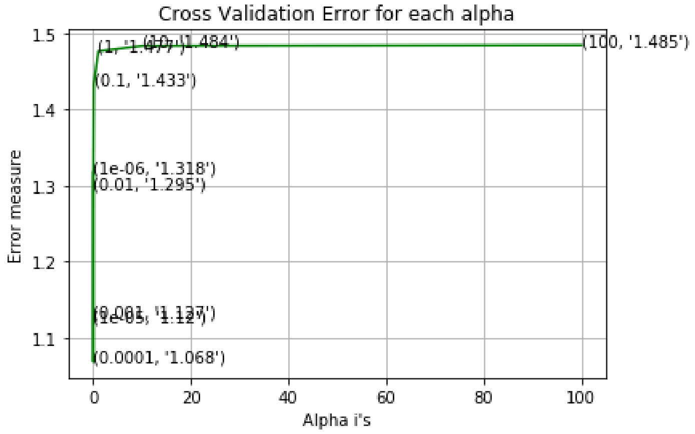
4.3.1.1. Hyper paramter tuning

In [124]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_mo
del.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=N
one, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0,
power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

#-------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geometric-intuiti
on-1/
#------------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklea
rn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
#-------------------------------------
# video link:
#-------------------------------------

alpha = [10 ** x for x in range(-6, 3)]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = SGDClassifier(class_weight='balanced', alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_x_onehotCoding, train_y)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x_onehotCoding, train_y)
    sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
    cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    # to avoid rounding error while multiplying probabilites we use log-probability estimates
    print("Log Loss :",log_loss(cv_y, sig_clf_probs))

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],str(txt)), (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()


best_alpha = np.argmin(cv_log_error_array)
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='log', random_
state=42)
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pred
ict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_
cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predic
t_y, labels=clf.classes_, eps=1e-15))
```

```
for alpha = 1e-06
Log Loss : 1.3178069524763467
for alpha = 1e-05
Log Loss : 1.1203949241613913
for alpha = 0.0001
Log Loss : 1.0679574362247795
for alpha = 0.001
Log Loss : 1.127383160254256
```

```
for alpha = 0.01
Log Loss : 1.2952474441992063
for alpha = 0.1
Log Loss : 1.4331332991514008
for alpha = 1
Log Loss : 1.476987916436993
for alpha = 10
Log Loss : 1.4840537241071972
for alpha = 100
Log Loss : 1.4849056750768814
```



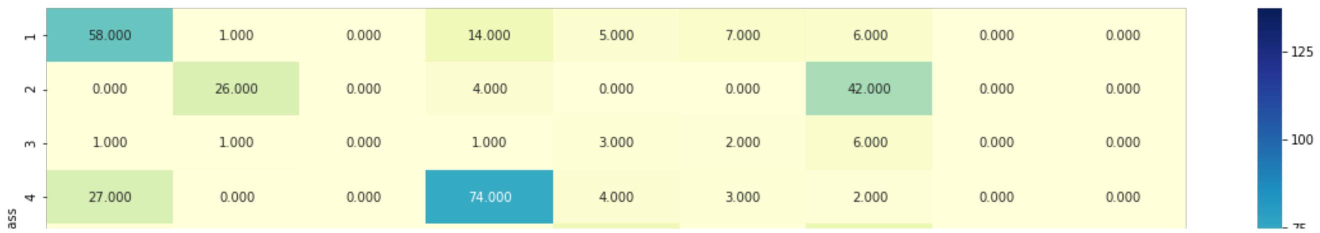
```
For values of best alpha =  0.0001 The train log loss is: 0.5067206993622535
For values of best alpha =  0.0001 The cross validation log loss is: 1.0679574362247795
For values of best alpha =  0.0001 The test log loss is: 1.0581814077799885
```

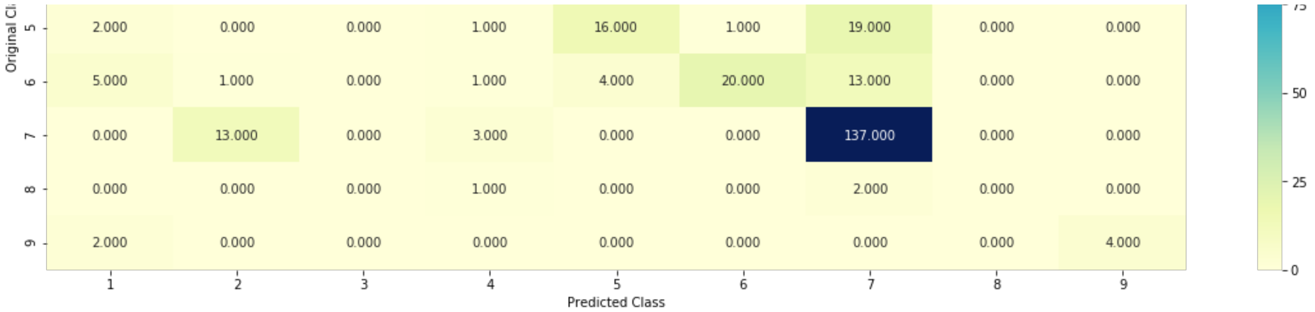
4.3.1.2. Testing the model with best hyper paramters

In [125]:

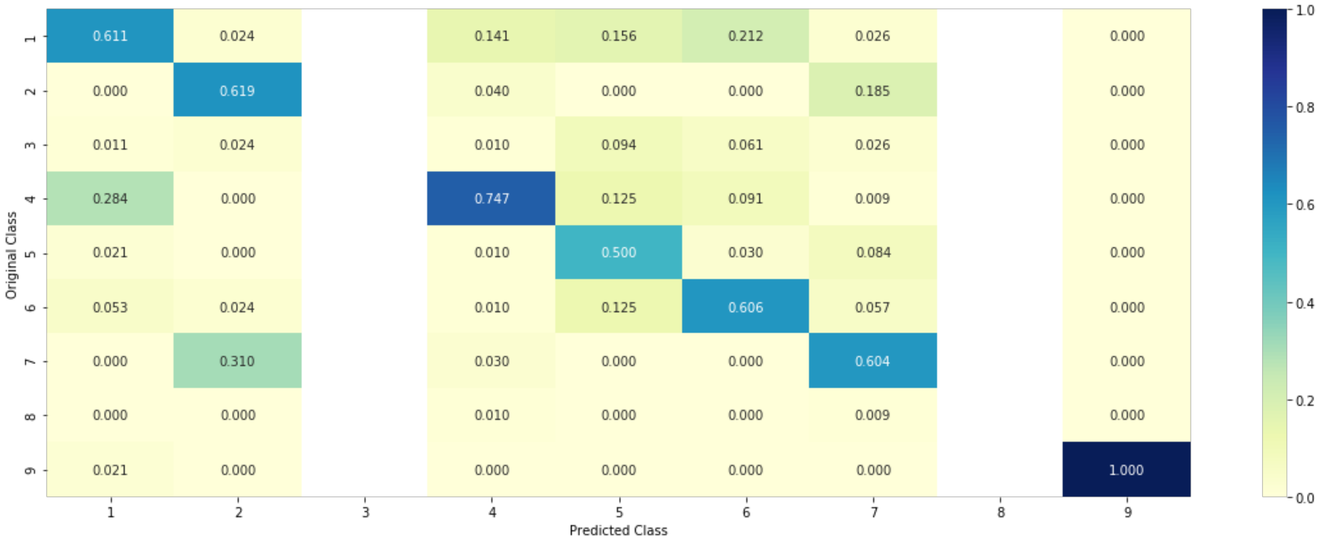
```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_mo
del.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=N
one, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0,
power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

#-------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geometric-intuiti
on-1/
#------------------------------
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='log', random_
state=42)
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y, cv_x_onehotCoding, cv_y, clf)
```

```
Log loss : 1.0679574362247795
Number of mis-classified points : 0.37030075187969924
-------------------- Confusion matrix --------------------
```

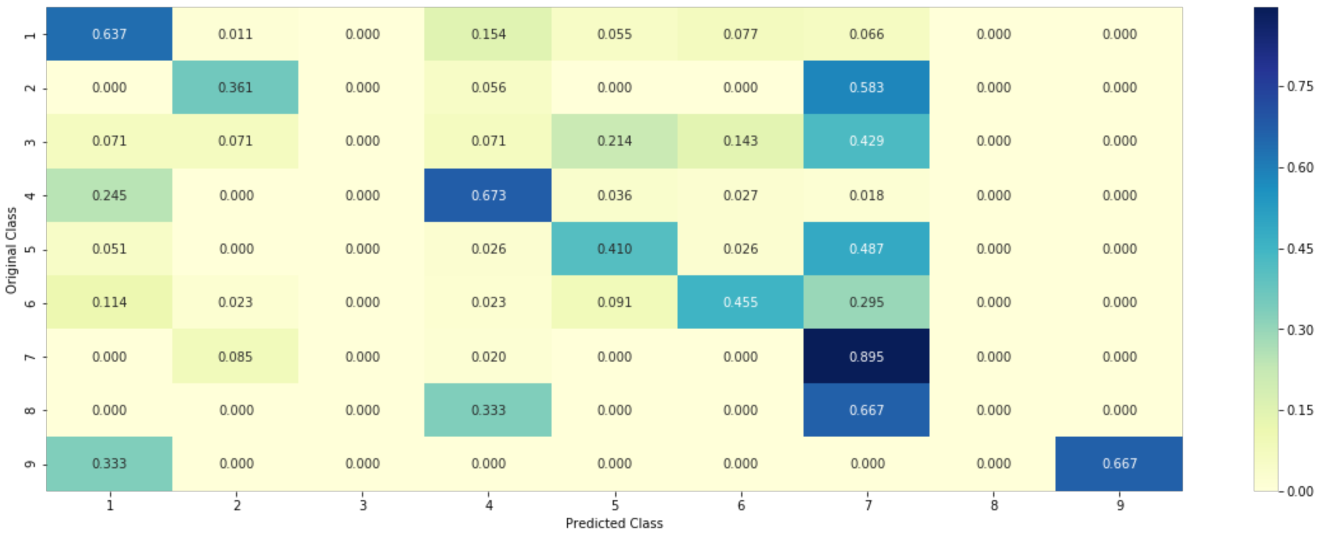




```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



```
-------------------- Recall matrix (Row sum=1) --------------------
```



4.3.1.3. Feature Importance

In [126]:

```
def get_imp_feature_names(text, indices, removed_ind = []):
    word_present = 0
    tabulte_list = []
    incresingorder_ind = 0
    for i in indices:
        if i < train_gene_feature_onehotCoding.shape[1]:
            tabulte_list.append([incresingorder_ind, "Gene", "Yes"])
        elif i< 18:
            tabulte_list.append([incresingorder_ind,"Variation", "Yes"])
        if ((i > 17) & (i not in removed_ind)) :
            word = train_text_features[i]
            yes_no = True if word in text.split() else False
            if yes_no:
                word_present += 1
            tabulte_list.append([incresingorder_ind,train_text_features[i], yes_no])
        incresingorder_ind += 1
    print(word_present, "most importent features are present in our query point")
    print("-"*50)
    print("The features that are most importent of the ",predicted_cls[0]," class:")
    print (tabulate(tabulte_list, headers=["Index",'Feature name', 'Present or Not']))
```

4.3.1.3.1. Correctly Classified point

In [127]:

```
# from tabulate import tabulate
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='log', random_
state=42)
clf.fit(train_x_onehotCoding,train_y)
test_point_index = 1
no_feature = 1000
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point
_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 6
Predicted Class Probabilities: [[1.600e-03 2.000e-04 1.000e-04 2.400e-03 1.423e-01 8.499e-01 1.000e-04
  8.000e-04 2.700e-03]]
Actual Class : 6
--------------------------------------------------
59 Text feature [substitutions] present in test data point [True]
71 Text feature [fell] present in test data point [True]
80 Text feature [i124v] present in test data point [True]
108 Text feature [ovarian] present in test data point [True]
112 Text feature [071] present in test data point [True]
113 Text feature [ivs13] present in test data point [True]
116 Text feature [predicted] present in test data point [True]
120 Text feature [expected] present in test data point [True]
121 Text feature [16c] present in test data point [True]
127 Text feature [ethnic] present in test data point [True]
138 Text feature [i2285v] present in test data point [True]
147 Text feature [loss] present in test data point [True]
155 Text feature [mastectomy] present in test data point [True]
156 Text feature [20c] present in test data point [True]
161 Text feature [brca] present in test data point [True]
162 Text feature [dcis] present in test data point [True]
166 Text feature [000] present in test data point [True]
167 Text feature [bic] present in test data point [True]
170 Text feature [p142h] present in test data point [True]
171 Text feature [y105c] present in test data point [True]
174 Text feature [model] present in test data point [True]
175 Text feature [ivs5] present in test data point [True]
181 Text feature [resource] present in test data point [True]
188 Text feature [ivs8] present in test data point [True]
189 Text feature [practice] present in test data point [True]
191 Text feature [s1172l] present in test data point [True]
192 Text feature [brca1] present in test data point [True]
193 Text feature [7g] present in test data point [True]
195 Text feature [ors] present in test data point [True]
196 Text feature [ivs20] present in test data point [True]
202 Text feature [440] present in test data point [True]
205 Text feature [923] present in test data point [True]
209 Text feature [quantifiable] present in test data point [True]
212 Text feature [threshold] present in test data point [True]
219 Text feature [685] present in test data point [True]
221 Text feature [programming] present in test data point [True]
```

```
221 Text feature [programming] present in test data point [True]
222 Text feature [57] present in test data point [True]
223 Text feature [basis] present in test data point [True]
224 Text feature [multiplying] present in test data point [True]
226 Text feature [ivs18] present in test data point [True]
227 Text feature [k513r] present in test data point [True]
228 Text feature [k2950n] present in test data point [True]
230 Text feature [lrs] present in test data point [True]
231 Text feature [08] present in test data point [True]
232 Text feature [multiplied] present in test data point [True]
233 Text feature [odds] present in test data point [True]
235 Text feature [054] present in test data point [True]
236 Text feature [personal] present in test data point [True]
237 Text feature [deleterious] present in test data point [True]
243 Text feature [3098] present in test data point [True]
244 Text feature [specialists] present in test data point [True]
245 Text feature [hampered] present in test data point [True]
246 Text feature [outside] present in test data point [True]
247 Text feature [74] present in test data point [True]
250 Text feature [logically] present in test data point [True]
251 Text feature [assumption] present in test data point [True]
253 Text feature [abkevich] present in test data point [True]
254 Text feature [79] present in test data point [True]
260 Text feature [trans] present in test data point [True]
261 Text feature [ascertained] present in test data point [True]
262 Text feature [979] present in test data point [True]
264 Text feature [subdivided] present in test data point [True]
265 Text feature [classify] present in test data point [True]
269 Text feature [predictive] present in test data point [True]
270 Text feature [20t] present in test data point [True]
271 Text feature [heretofore] present in test data point [True]
272 Text feature [classified] present in test data point [True]
273 Text feature [specify] present in test data point [True]
274 Text feature [logistic] present in test data point [True]
275 Text feature [heterozygotes] present in test data point [True]
277 Text feature [missense] present in test data point [True]
278 Text feature [lr] present in test data point [True]
279 Text feature [alignments] present in test data point [True]
282 Text feature [discussionusing] present in test data point [True]
283 Text feature [carry] present in test data point [True]
285 Text feature [disciplines] present in test data point [True]
287 Text feature [feel] present in test data point [True]
290 Text feature [histogram] present in test data point [True]
293 Text feature [uncertain] present in test data point [True]
295 Text feature [conserved] present in test data point [True]
298 Text feature [offered] present in test data point [True]
300 Text feature [nonmammalian] present in test data point [True]
301 Text feature [02] present in test data point [True]
302 Text feature [conservation] present in test data point [True]
306 Text feature [latin] present in test data point [True]
307 Text feature [q804h] present in test data point [True]
308 Text feature [vanishingly] present in test data point [True]
311 Text feature [772] present in test data point [True]
312 Text feature [caribbean] present in test data point [True]
313 Text feature [ethnicity] present in test data point [True]
316 Text feature [stata] present in test data point [True]
317 Text feature [evidence] present in test data point [True]
318 Text feature [history] present in test data point [True]
319 Text feature [reassured] present in test data point [True]
320 Text feature [088] present in test data point [True]
321 Text feature [433] present in test data point [True]
322 Text feature [493] present in test data point [True]
323 Text feature [summaries] present in test data point [True]
324 Text feature [antoniou] present in test data point [True]
325 Text feature [1010] present in test data point [True]
326 Text feature [049] present in test data point [True]
328 Text feature [predispose] present in test data point [True]
329 Text feature [favor] present in test data point [True]
330 Text feature [likely] present in test data point [True]
331 Text feature [ivs19] present in test data point [True]
332 Text feature [ifdis] present in test data point [True]
333 Text feature [ifdi] present in test data point [True]
334 Text feature [mvs] present in test data point [True]
335 Text feature [d369n] present in test data point [True]
336 Text feature [n1468h] present in test data point [True]
337 Text feature [n1228d] present in test data point [True]
338 Text feature [d420y] present in test data point [True]
339 Text feature [d648h] present in test data point [True]
```

```
339 Text feature [d642h] present in test data point [True]
340 Text feature [n1102y] present in test data point [True]
341 Text feature [d806h] present in test data point [True]
342 Text feature [g890v] present in test data point [True]
343 Text feature [deletioneuropeana] present in test data point [True]
344 Text feature [sourceturn] present in test data point [True]
345 Text feature [g602r] present in test data point [True]
346 Text feature [e1419q] present in test data point [True]
347 Text feature [g1529r] present in test data point [True]
348 Text feature [e1682k] present in test data point [True]
349 Text feature [e597k] present in test data point [True]
350 Text feature [e842g] present in test data point [True]
351 Text feature [genotypeethnic] present in test data point [True]
352 Text feature [f1524v] present in test data point [True]
353 Text feature [g1194d] present in test data point [True]
354 Text feature [formview] present in test data point [True]
355 Text feature [mutationsto] present in test data point [True]
356 Text feature [f1662s] present in test data point [True]
357 Text feature [g1771d] present in test data point [True]
358 Text feature [i1275v] present in test data point [True]
359 Text feature [i1405v] present in test data point [True]
360 Text feature [h1918y] present in test data point [True]
361 Text feature [r504h] present in test data point [True]
362 Text feature [k862e] present in test data point [True]
363 Text feature [r3052q] present in test data point [True]
364 Text feature [r2888c] present in test data point [True]
365 Text feature [m297i] present in test data point [True]
366 Text feature [r2842h] present in test data point [True]
367 Text feature [r2502c] present in test data point [True]
368 Text feature [l1019v] present in test data point [True]
369 Text feature [m1361l] present in test data point [True]
370 Text feature [l1904v] present in test data point [True]
371 Text feature [l2396f] present in test data point [True]
372 Text feature [r2108h] present in test data point [True]
373 Text feature [r1028h] present in test data point [True]
374 Text feature [r1190w] present in test data point [True]
375 Text feature [r1203q] present in test data point [True]
376 Text feature [llr] present in test data point [True]
377 Text feature [r1495m] present in test data point [True]
378 Text feature [k2411t] present in test data point [True]
379 Text feature [k1690n] present in test data point [True]
380 Text feature [k1109n] present in test data point [True]
381 Text feature [reassurance] present in test data point [True]
382 Text feature [h2074n] present in test data point [True]
383 Text feature [i1044v] present in test data point [True]
384 Text feature [i1349t] present in test data point [True]
385 Text feature [n2048i] present in test data point [True]
386 Text feature [q1396r] present in test data point [True]
387 Text feature [i1858l] present in test data point [True]
388 Text feature [i1929v] present in test data point [True]
389 Text feature [q155e] present in test data point [True]
390 Text feature [gtrag] present in test data point [True]
391 Text feature [i925l] present in test data point [True]
392 Text feature [methodssource] present in test data point [True]
393 Text feature [q2384k] present in test data point [True]
394 Text feature [s1266t] present in test data point [True]
395 Text feature [individualbrca1] present in test data point [True]
396 Text feature [invarianta] present in test data point [True]
397 Text feature [isrk] present in test data point [True]
398 Text feature [isview] present in test data point [True]
399 Text feature [ivs25] present in test data point [True]
400 Text feature [s1733f] present in test data point [True]
401 Text feature [d3170g] present in test data point [True]
402 Text feature [t1685a] present in test data point [True]
403 Text feature [d1280v] present in test data point [True]
404 Text feature [v3079i] present in test data point [True]
405 Text feature [p1819s] present in test data point [True]
406 Text feature [37the] present in test data point [True]
407 Text feature [p168t] present in test data point [True]
408 Text feature [v2969m] present in test data point [True]
409 Text feature [probandsor] present in test data point [True]
410 Text feature [4603g] present in test data point [True]
411 Text feature [v1306i] present in test data point [True]
412 Text feature [4987c] present in test data point [True]
413 Text feature [p1238l] present in test data point [True]
414 Text feature [5181del3] present in test data point [True]
415 Text feature [overstate] present in test data point [True]
```

```
416 Text feature [v1247i] present in test data point [True]
417 Text feature [probandswe] present in test data point [True]
418 Text feature [63mv] present in test data point [True]
419 Text feature [ordiagnosis] present in test data point [True]
420 Text feature [or7] present in test data point [True]
421 Text feature [or5] present in test data point [True]
422 Text feature [or2] present in test data point [True]
423 Text feature [probandsno] present in test data point [True]
424 Text feature [v894i] present in test data point [True]
425 Text feature [2to] present in test data point [True]
426 Text feature [p1859r] present in test data point [True]
427 Text feature [0other] present in test data point [True]
428 Text feature [pedigreeselsewhere] present in test data point [True]
429 Text feature [103splice] present in test data point [True]
430 Text feature [104brca2] present in test data point [True]
431 Text feature [108table] present in test data point [True]
432 Text feature [11delt] present in test data point [True]
433 Text feature [1225del3] present in test data point [True]
434 Text feature [y3098h] present in test data point [True]
435 Text feature [y3092c] present in test data point [True]
436 Text feature [9345g] present in test data point [True]
437 Text feature [16ain] present in test data point [True]
438 Text feature [21table] present in test data point [True]
439 Text feature [24brca2] present in test data point [True]
440 Text feature [2all] present in test data point [True]
441 Text feature [p375s] present in test data point [True]
442 Text feature [p334l] present in test data point [True]
443 Text feature [p3039p] present in test data point [True]
444 Text feature [2del21insa] present in test data point [True]
445 Text feature [w2626c] present in test data point [True]
446 Text feature [2ln] present in test data point [True]
447 Text feature [17g] present in test data point [True]
448 Text feature [d1546y] present in test data point [True]
449 Text feature [optionsoverall] present in test data point [True]
450 Text feature [optionsas] present in test data point [True]
451 Text feature [neutralitybrca1] present in test data point [True]
452 Text feature [c3198r] present in test data point [True]
453 Text feature [c554w] present in test data point [True]
454 Text feature [t630i] present in test data point [True]
455 Text feature [causalitybrca1] present in test data point [True]
456 Text feature [causalityllrgene] present in test data point [True]
457 Text feature [n723d] present in test data point [True]
458 Text feature [t582p] present in test data point [True]
459 Text feature [n56t] present in test data point [True]
460 Text feature [neutralityllrgene] present in test data point [True]
461 Text feature [n473s] present in test data point [True]
462 Text feature [t3349a] present in test data point [True]
463 Text feature [t2250a] present in test data point [True]
464 Text feature [classto] present in test data point [True]
465 Text feature [llrs] present in test data point [True]
466 Text feature [n2436i] present in test data point [True]
467 Text feature [n2113s] present in test data point [True]
468 Text feature [t1354m] present in test data point [True]
469 Text feature [t1349m] present in test data point [True]
470 Text feature [straightforwardly] present in test data point [True]
471 Text feature [characteristicsno] present in test data point [True]
472 Text feature [optionsfrequency] present in test data point [True]
473 Text feature [c1365y] present in test data point [True]
474 Text feature [brca2personal] present in test data point [True]
475 Text feature [onwhere] present in test data point [True]
476 Text feature [a1623g] present in test data point [True]
477 Text feature [a280g] present in test data point [True]
478 Text feature [onthis] present in test data point [True]
479 Text feature [a622v] present in test data point [True]
480 Text feature [a75p] present in test data point [True]
481 Text feature [ontherefore] present in test data point [True]
482 Text feature [aassessed] present in test data point [True]
483 Text feature [onthenview] present in test data point [True]
484 Text feature [onthat] present in test data point [True]
485 Text feature [llrvariant] present in test data point [True]
486 Text feature [age2] present in test data point [True]
487 Text feature [analyzedthe] present in test data point [True]
488 Text feature [onlet] present in test data point [True]
489 Text feature [onfor] present in test data point [True]
490 Text feature [onandview] present in test data point [True]
491 Text feature [andreported] present in test data point [True]
492 Text feature [typehistory] present in test data point [True]
```

```
493 Text feature [aspersonal] present in test data point [True]
494 Text feature [asview] present in test data point [True]
495 Text feature [thereforeview] present in test data point [True]
496 Text feature [byview] present in test data point [True]
497 Text feature [qj] present in test data point [True]
498 Text feature [pj] present in test data point [True]
499 Text feature [collapsing] present in test data point [True]
500 Text feature [10g] present in test data point [True]
501 Text feature [optionstable] present in test data point [True]
502 Text feature [ivs24] present in test data point [True]
503 Text feature [partitions] present in test data point [True]
504 Text feature [12g] present in test data point [True]
505 Text feature [resultswe] present in test data point [True]
506 Text feature [suppose] present in test data point [True]
507 Text feature [counselor] present in test data point [True]
508 Text feature [groupings] present in test data point [True]
509 Text feature [constellations] present in test data point [True]
510 Text feature [cancera] present in test data point [True]
511 Text feature [1012] present in test data point [True]
512 Text feature [ivs15] present in test data point [True]
514 Text feature [ivs12] present in test data point [True]
515 Text feature [mathjax] present in test data point [True]
516 Text feature [ivs6] present in test data point [True]
517 Text feature [age3] present in test data point [True]
519 Text feature [provider] present in test data point [True]
520 Text feature [language] present in test data point [True]
521 Text feature [s186y] present in test data point [True]
528 Text feature [ivs21] present in test data point [True]
529 Text feature [ivs11] present in test data point [True]
530 Text feature [regressions] present in test data point [True]
782 Text feature [df] present in test data point [True]
829 Text feature [liability] present in test data point [True]
874 Text feature [family] present in test data point [True]
964 Text feature [892] present in test data point [True]
978 Text feature [statacorp] present in test data point [True]
Out of the top  1000  features  301 are present in query point
```

4.3.1.3.2. Incorrectly Classified point

In [128]:

```
test_point_index = 2
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point
_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.0742 0.0656 0.0193 0.128  0.0405 0.0342 0.6238 0.0057 0.0086]]
Actual Class : 6
--------------------------------------------------
3 Text feature [activated] present in test data point [True]
6 Text feature [downstream] present in test data point [True]
71 Text feature [inhibitor] present in test data point [True]
273 Text feature [kinase] present in test data point [True]
274 Text feature [activating] present in test data point [True]
279 Text feature [treated] present in test data point [True]
Out of the top  500  features  6 are present in query point
```

4.3.2. Without Class balancing

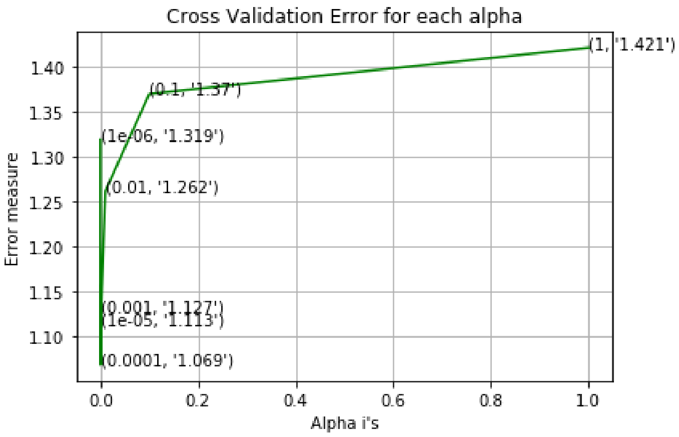
4.3.2.1. Hyper paramter tuning

In [129]:

In [129]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_mo
del.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=N
one, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0,
power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

#-------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geometric-intuiti
on-1/
#------------------------------



# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklea
rn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
#-------------------------------------
# video link:
#-------------------------------------

alpha = [10 ** x for x in range(-6, 1)]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_x_onehotCoding, train_y)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x_onehotCoding, train_y)
    sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
    cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    print("Log Loss :",log_loss(cv_y, sig_clf_probs))

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],str(txt)), (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()


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

predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pred
ict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_
cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predic
t_y, labels=clf.classes_, eps=1e-15))
```

```
for alpha = 1e-06
Log Loss : 1.318736675375108
for alpha = 1e-05
Log Loss : 1.1132728226808046
for alpha = 0.0001
Log Loss : 1.0687923937447215
for alpha = 0.001
Log Loss : 1.1272434090675065
for alpha = 0.01
Log Loss : 1.2616070179516259
for alpha = 0.1
Log Loss : 1.3696274749674393
for alpha = 1
Log Loss : 1.4208495514374617
```



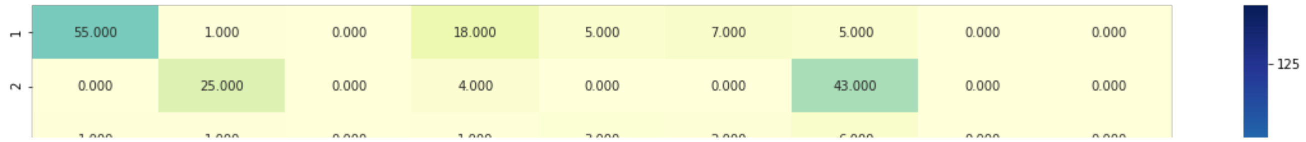
```
For values of best alpha =  0.0001 The train log loss is: 0.4855973138891092
For values of best alpha =  0.0001 The cross validation log loss is: 1.0687923937447215
For values of best alpha =  0.0001 The test log loss is: 1.0548069731945946
```

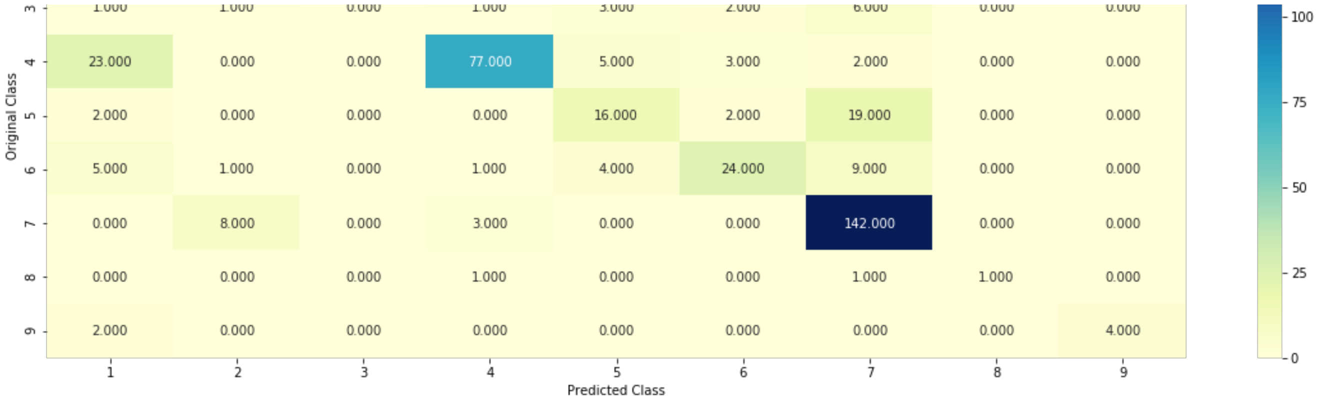
4.3.2.2. Testing model with best hyper parameters

In [130]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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

clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y, cv_x_onehotCoding, cv_y, clf)
```

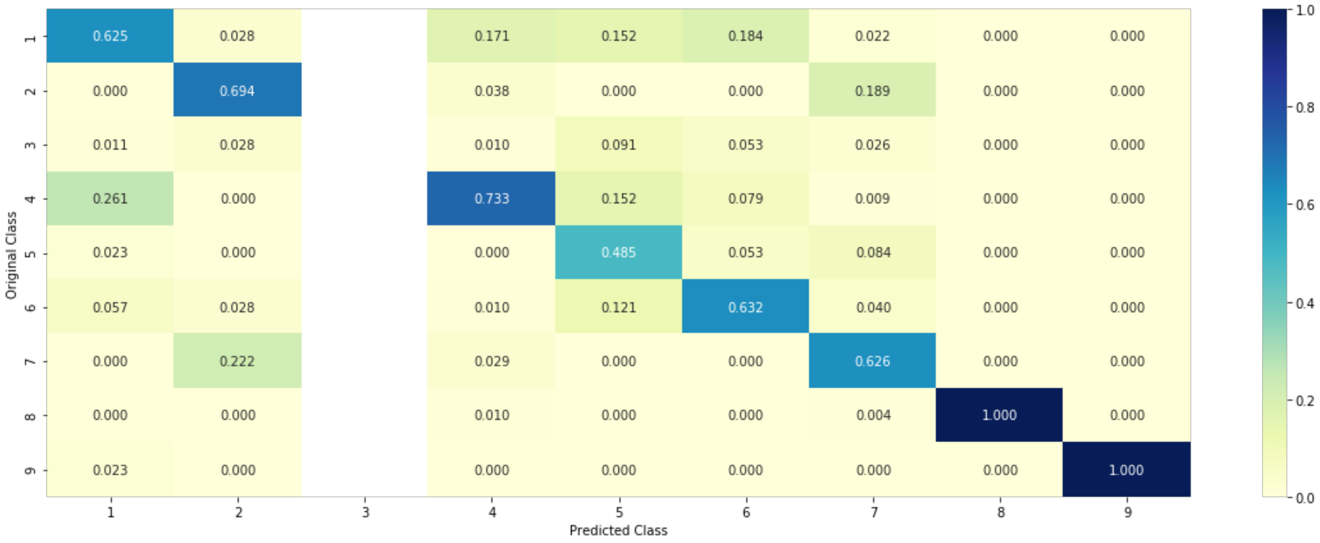
```
Log loss : 1.0687923937447215
Number of mis-classified points : 0.3533834586466165
-------------------- Confusion matrix --------------------
```





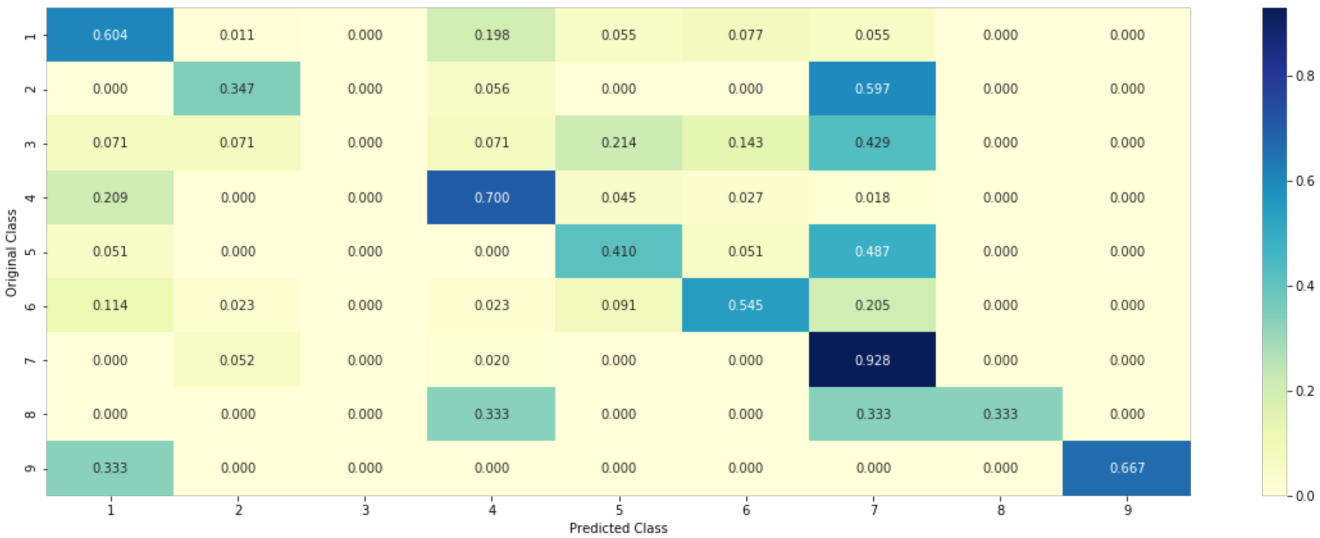
	1	2	3	4	5	6	7	8	9
3	1.000	1.000	0.000	1.000	3.000	2.000	6.000	0.000	0.000
4	23.000	0.000	0.000	77.000	5.000	3.000	2.000	0.000	0.000
5	2.000	0.000	0.000	0.000	16.000	2.000	19.000	0.000	0.000
6	5.000	1.000	0.000	1.000	4.000	24.000	9.000	0.000	0.000
7	0.000	8.000	0.000	3.000	0.000	0.000	142.000	0.000	0.000
8	0.000	0.000	0.000	1.000	0.000	0.000	1.000	1.000	0.000
9	2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	4.000

```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



	1	2	3	4	5	6	7	8	9
1	0.625	0.028		0.171	0.152	0.184	0.022	0.000	0.000
2	0.000	0.694		0.038	0.000	0.000	0.189	0.000	0.000
3	0.011	0.028		0.010	0.091	0.053	0.026	0.000	0.000
4	0.261	0.000		0.733	0.152	0.079	0.009	0.000	0.000
5	0.023	0.000		0.000	0.485	0.053	0.084	0.000	0.000
6	0.057	0.028		0.010	0.121	0.632	0.040	0.000	0.000
7	0.000	0.222		0.029	0.000	0.000	0.626	0.000	0.000
8	0.000	0.000		0.010	0.000	0.000	0.004	1.000	0.000
9	0.023	0.000		0.000	0.000	0.000	0.000	0.000	1.000

```
-------------------- Recall matrix (Row sum=1) --------------------
```



	1	2	3	4	5	6	7	8	9
1	0.604	0.011	0.000	0.198	0.055	0.077	0.055	0.000	0.000
2	0.000	0.347	0.000	0.056	0.000	0.000	0.597	0.000	0.000
3	0.071	0.071	0.000	0.071	0.214	0.143	0.429	0.000	0.000
4	0.209	0.000	0.000	0.700	0.045	0.027	0.018	0.000	0.000
5	0.051	0.000	0.000	0.000	0.410	0.051	0.487	0.000	0.000
6	0.114	0.023	0.000	0.023	0.091	0.545	0.205	0.000	0.000
7	0.000	0.052	0.000	0.020	0.000	0.000	0.928	0.000	0.000
8	0.000	0.000	0.000	0.333	0.000	0.000	0.333	0.333	0.000
9	0.333	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.667

4.3.2.3. Feature Importance, Correctly Classified point

In [131]:

```
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)
clf.fit(train_x_onehotCoding,train_y)
test_point_index = 1
no_feature = 1000
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
```

```
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point
_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 6
Predicted Class Probabilities: [[6.000e-04 3.000e-04 0.000e+00 2.100e-03 6.760e-02 9.288e-01 1.000e-04
  0.000e+00 3.000e-04]]
Actual Class : 6
--------------------------------------------------
60 Text feature [substitutions] present in test data point [True]
69 Text feature [fell] present in test data point [True]
75 Text feature [i124v] present in test data point [True]
108 Text feature [ivs13] present in test data point [True]
110 Text feature [ovarian] present in test data point [True]
117 Text feature [predicted] present in test data point [True]
119 Text feature [expected] present in test data point [True]
120 Text feature [071] present in test data point [True]
121 Text feature [16c] present in test data point [True]
130 Text feature [ethnic] present in test data point [True]
132 Text feature [i2285v] present in test data point [True]
154 Text feature [loss] present in test data point [True]
157 Text feature [20c] present in test data point [True]
158 Text feature [brca] present in test data point [True]
159 Text feature [mastectomy] present in test data point [True]
162 Text feature [bic] present in test data point [True]
163 Text feature [dcis] present in test data point [True]
168 Text feature [000] present in test data point [True]
170 Text feature [p142h] present in test data point [True]
171 Text feature [y105c] present in test data point [True]
172 Text feature [ivs5] present in test data point [True]
177 Text feature [model] present in test data point [True]
178 Text feature [resource] present in test data point [True]
180 Text feature [ivs8] present in test data point [True]
189 Text feature [s1172l] present in test data point [True]
191 Text feature [brca1] present in test data point [True]
193 Text feature [ivs20] present in test data point [True]
194 Text feature [7g] present in test data point [True]
195 Text feature [ors] present in test data point [True]
196 Text feature [practice] present in test data point [True]
202 Text feature [923] present in test data point [True]
204 Text feature [440] present in test data point [True]
208 Text feature [threshold] present in test data point [True]
214 Text feature [685] present in test data point [True]
216 Text feature [quantifiable] present in test data point [True]
218 Text feature [57] present in test data point [True]
221 Text feature [ivs18] present in test data point [True]
223 Text feature [basis] present in test data point [True]
224 Text feature [k513r] present in test data point [True]
225 Text feature [k2950n] present in test data point [True]
226 Text feature [programming] present in test data point [True]
229 Text feature [lrs] present in test data point [True]
230 Text feature [multiplying] present in test data point [True]
232 Text feature [odds] present in test data point [True]
233 Text feature [personal] present in test data point [True]
234 Text feature [multiplied] present in test data point [True]
235 Text feature [deleterious] present in test data point [True]
236 Text feature [054] present in test data point [True]
238 Text feature [08] present in test data point [True]
243 Text feature [3098] present in test data point [True]
245 Text feature [specialists] present in test data point [True]
246 Text feature [logically] present in test data point [True]
247 Text feature [outside] present in test data point [True]
248 Text feature [hampered] present in test data point [True]
249 Text feature [assumption] present in test data point [True]
250 Text feature [abkevich] present in test data point [True]
252 Text feature [74] present in test data point [True]
256 Text feature [79] present in test data point [True]
257 Text feature [trans] present in test data point [True]
258 Text feature [ascertained] present in test data point [True]
260 Text feature [subdivided] present in test data point [True]
261 Text feature [classify] present in test data point [True]
```

```
264 Text feature [979] present in test data point [True]
266 Text feature [classified] present in test data point [True]
267 Text feature [specify] present in test data point [True]
269 Text feature [20t] present in test data point [True]
270 Text feature [predictive] present in test data point [True]
271 Text feature [lr] present in test data point [True]
272 Text feature [heterozygotes] present in test data point [True]
274 Text feature [logistic] present in test data point [True]
275 Text feature [alignments] present in test data point [True]
278 Text feature [heretofore] present in test data point [True]
279 Text feature [missense] present in test data point [True]
281 Text feature [carry] present in test data point [True]
283 Text feature [discussionusing] present in test data point [True]
284 Text feature [disciplines] present in test data point [True]
288 Text feature [offered] present in test data point [True]
289 Text feature [feel] present in test data point [True]
292 Text feature [uncertain] present in test data point [True]
294 Text feature [histogram] present in test data point [True]
297 Text feature [nonmammalian] present in test data point [True]
298 Text feature [02] present in test data point [True]
300 Text feature [conservation] present in test data point [True]
303 Text feature [conserved] present in test data point [True]
305 Text feature [latin] present in test data point [True]
307 Text feature [vanishingly] present in test data point [True]
309 Text feature [caribbean] present in test data point [True]
310 Text feature [ethnicity] present in test data point [True]
311 Text feature [q804h] present in test data point [True]
312 Text feature [stata] present in test data point [True]
313 Text feature [history] present in test data point [True]
314 Text feature [772] present in test data point [True]
316 Text feature [088] present in test data point [True]
317 Text feature [reassured] present in test data point [True]
319 Text feature [433] present in test data point [True]
320 Text feature [493] present in test data point [True]
321 Text feature [summaries] present in test data point [True]
324 Text feature [antoniou] present in test data point [True]
325 Text feature [evidence] present in test data point [True]
326 Text feature [1010] present in test data point [True]
327 Text feature [favor] present in test data point [True]
328 Text feature [ifdis] present in test data point [True]
329 Text feature [ivs19] present in test data point [True]
330 Text feature [g890v] present in test data point [True]
331 Text feature [reassurance] present in test data point [True]
332 Text feature [d642h] present in test data point [True]
333 Text feature [5181del3] present in test data point [True]
334 Text feature [llrvariant] present in test data point [True]
335 Text feature [llrs] present in test data point [True]
336 Text feature [llr] present in test data point [True]
337 Text feature [63mv] present in test data point [True]
338 Text feature [v2969m] present in test data point [True]
339 Text feature [d806h] present in test data point [True]
340 Text feature [v1306i] present in test data point [True]
341 Text feature [straightforwardly] present in test data point [True]
342 Text feature [v1247i] present in test data point [True]
343 Text feature [deletioneuropeana] present in test data point [True]
344 Text feature [t582p] present in test data point [True]
345 Text feature [l2396f] present in test data point [True]
346 Text feature [l1904v] present in test data point [True]
347 Text feature [causalityllrgene] present in test data point [True]
348 Text feature [l1019v] present in test data point [True]
349 Text feature [k862e] present in test data point [True]
350 Text feature [s1266t] present in test data point [True]
351 Text feature [d420y] present in test data point [True]
352 Text feature [d369n] present in test data point [True]
353 Text feature [d3170g] present in test data point [True]
354 Text feature [s1733f] present in test data point [True]
355 Text feature [t1685a] present in test data point [True]
356 Text feature [2to] present in test data point [True]
357 Text feature [37the] present in test data point [True]
358 Text feature [r2108h] present in test data point [True]
359 Text feature [r2502c] present in test data point [True]
360 Text feature [t1354m] present in test data point [True]
361 Text feature [t3349a] present in test data point [True]
362 Text feature [t1349m] present in test data point [True]
363 Text feature [r2842h] present in test data point [True]
364 Text feature [r2888c] present in test data point [True]
365 Text feature [methodssource] present in test data point [True]
```

```
366 Text feature [r3052q] present in test data point [True]
367 Text feature [classto] present in test data point [True]
368 Text feature [d1280v] present in test data point [True]
369 Text feature [r504h] present in test data point [True]
370 Text feature [w2626c] present in test data point [True]
371 Text feature [characteristicsno] present in test data point [True]
372 Text feature [4603g] present in test data point [True]
373 Text feature [m297i] present in test data point [True]
374 Text feature [v894i] present in test data point [True]
375 Text feature [d1546y] present in test data point [True]
376 Text feature [m1361l] present in test data point [True]
377 Text feature [v3079i] present in test data point [True]
378 Text feature [4987c] present in test data point [True]
379 Text feature [causalitybrca1] present in test data point [True]
380 Text feature [k1109n] present in test data point [True]
381 Text feature [k1690n] present in test data point [True]
382 Text feature [e842g] present in test data point [True]
383 Text feature [i1044v] present in test data point [True]
384 Text feature [c1365y] present in test data point [True]
385 Text feature [byview] present in test data point [True]
386 Text feature [age2] present in test data point [True]
387 Text feature [brca2personal] present in test data point [True]
388 Text feature [f1662s] present in test data point [True]
389 Text feature [thereforeview] present in test data point [True]
390 Text feature [sourceturn] present in test data point [True]
391 Text feature [analyzedthe] present in test data point [True]
392 Text feature [2ln] present in test data point [True]
393 Text feature [andreported] present in test data point [True]
394 Text feature [h1918y] present in test data point [True]
395 Text feature [gtrag] present in test data point [True]
396 Text feature [formview] present in test data point [True]
397 Text feature [genotypeethnic] present in test data point [True]
398 Text feature [g1194d] present in test data point [True]
399 Text feature [typehistory] present in test data point [True]
400 Text feature [aspersonal] present in test data point [True]
401 Text feature [g1529r] present in test data point [True]
402 Text feature [asview] present in test data point [True]
403 Text feature [g1771d] present in test data point [True]
404 Text feature [h2074n] present in test data point [True]
405 Text feature [k2411t] present in test data point [True]
406 Text feature [aassessed] present in test data point [True]
407 Text feature [a622v] present in test data point [True]
408 Text feature [g602r] present in test data point [True]
409 Text feature [9345g] present in test data point [True]
410 Text feature [t630i] present in test data point [True]
411 Text feature [e1419q] present in test data point [True]
412 Text feature [a1623g] present in test data point [True]
413 Text feature [ivs25] present in test data point [True]
414 Text feature [t2250a] present in test data point [True]
415 Text feature [c554w] present in test data point [True]
416 Text feature [e1682k] present in test data point [True]
417 Text feature [isview] present in test data point [True]
418 Text feature [a75p] present in test data point [True]
419 Text feature [isrk] present in test data point [True]
420 Text feature [c3198r] present in test data point [True]
421 Text feature [invarianta] present in test data point [True]
422 Text feature [e597k] present in test data point [True]
423 Text feature [individualbrca1] present in test data point [True]
424 Text feature [i925l] present in test data point [True]
425 Text feature [i1929v] present in test data point [True]
426 Text feature [i1858l] present in test data point [True]
427 Text feature [i1405v] present in test data point [True]
428 Text feature [i1349t] present in test data point [True]
429 Text feature [i1275v] present in test data point [True]
430 Text feature [a280g] present in test data point [True]
431 Text feature [2del21insa] present in test data point [True]
432 Text feature [f1524v] present in test data point [True]
433 Text feature [r1495m] present in test data point [True]
434 Text feature [or2] present in test data point [True]
435 Text feature [or5] present in test data point [True]
436 Text feature [pedigreeselsewhere] present in test data point [True]
437 Text feature [or7] present in test data point [True]
438 Text feature [ordiagnosis] present in test data point [True]
439 Text feature [q2384k] present in test data point [True]
440 Text feature [n723d] present in test data point [True]
441 Text feature [overstate] present in test data point [True]
442 Text feature [p1238l] present in test data point [True]
```

```
443 Text feature [n56t] present in test data point [True]
444 Text feature [n473s] present in test data point [True]
445 Text feature [p168t] present in test data point [True]
446 Text feature [neutralitybrca1] present in test data point [True]
447 Text feature [y3098h] present in test data point [True]
448 Text feature [1225del3] present in test data point [True]
449 Text feature [11delt] present in test data point [True]
450 Text feature [n1102y] present in test data point [True]
451 Text feature [p1859r] present in test data point [True]
452 Text feature [n2436i] present in test data point [True]
453 Text feature [n2113s] present in test data point [True]
454 Text feature [n1228d] present in test data point [True]
455 Text feature [n1468h] present in test data point [True]
456 Text feature [p3039p] present in test data point [True]
457 Text feature [21table] present in test data point [True]
458 Text feature [n2048i] present in test data point [True]
459 Text feature [p334l] present in test data point [True]
460 Text feature [p1819s] present in test data point [True]
461 Text feature [p375s] present in test data point [True]
462 Text feature [0other] present in test data point [True]
463 Text feature [optionsfrequency] present in test data point [True]
464 Text feature [103splice] present in test data point [True]
465 Text feature [onandview] present in test data point [True]
466 Text feature [probandswe] present in test data point [True]
467 Text feature [probandsor] present in test data point [True]
468 Text feature [probandsno] present in test data point [True]
469 Text feature [104brca2] present in test data point [True]
470 Text feature [2all] present in test data point [True]
471 Text feature [16ain] present in test data point [True]
472 Text feature [onfor] present in test data point [True]
473 Text feature [onlet] present in test data point [True]
474 Text feature [q1396r] present in test data point [True]
475 Text feature [r1203q] present in test data point [True]
476 Text feature [optionsoverall] present in test data point [True]
477 Text feature [17g] present in test data point [True]
478 Text feature [108table] present in test data point [True]
479 Text feature [mutationsto] present in test data point [True]
480 Text feature [onthenview] present in test data point [True]
481 Text feature [q155e] present in test data point [True]
482 Text feature [r1190w] present in test data point [True]
483 Text feature [r1028h] present in test data point [True]
484 Text feature [ontherefore] present in test data point [True]
485 Text feature [y3092c] present in test data point [True]
486 Text feature [onthis] present in test data point [True]
487 Text feature [onwhere] present in test data point [True]
488 Text feature [optionsas] present in test data point [True]
489 Text feature [neutralityllrgene] present in test data point [True]
490 Text feature [onthat] present in test data point [True]
491 Text feature [24brca2] present in test data point [True]
492 Text feature [ifdi] present in test data point [True]
493 Text feature [mvs] present in test data point [True]
494 Text feature [049] present in test data point [True]
495 Text feature [qj] present in test data point [True]
496 Text feature [pj] present in test data point [True]
497 Text feature [collapsing] present in test data point [True]
498 Text feature [10g] present in test data point [True]
499 Text feature [ivs24] present in test data point [True]
500 Text feature [12g] present in test data point [True]
501 Text feature [partitions] present in test data point [True]
502 Text feature [optionstable] present in test data point [True]
503 Text feature [resultswe] present in test data point [True]
504 Text feature [predispose] present in test data point [True]
505 Text feature [counselor] present in test data point [True]
506 Text feature [suppose] present in test data point [True]
507 Text feature [groupings] present in test data point [True]
508 Text feature [likely] present in test data point [True]
509 Text feature [constellations] present in test data point [True]
510 Text feature [cancera] present in test data point [True]
511 Text feature [1012] present in test data point [True]
512 Text feature [language] present in test data point [True]
513 Text feature [ivs15] present in test data point [True]
515 Text feature [ivs12] present in test data point [True]
516 Text feature [mathjax] present in test data point [True]
517 Text feature [provider] present in test data point [True]
518 Text feature [age3] present in test data point [True]
519 Text feature [ivs6] present in test data point [True]
521 Text feature [ivs21] present in test data point [True]
```

```
522 Text feature [ivs11] present in test data point [True]
528 Text feature [regressions] present in test data point [True]
714 Text feature [df] present in test data point [True]
717 Text feature [s186y] present in test data point [True]
782 Text feature [133] present in test data point [True]
814 Text feature [liability] present in test data point [True]
843 Text feature [892] present in test data point [True]
850 Text feature [statacorp] present in test data point [True]
963 Text feature [family] present in test data point [True]
973 Text feature [r0] present in test data point [True]
979 Text feature [529] present in test data point [True]
Out of the top  1000  features  304 are present in query point
```

4.3.2.4. Feature Importance, Inorrectly Classified point

In [132]:

```
test_point_index = 2
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.07   0.0626 0.0192 0.1255 0.0421 0.0357 0.6323 0.0054 0.0072]]
Actual Class : 6
--------------------------------------------------
4 Text feature [activated] present in test data point [True]
6 Text feature [downstream] present in test data point [True]
87 Text feature [inhibitor] present in test data point [True]
340 Text feature [kinase] present in test data point [True]
345 Text feature [treated] present in test data point [True]
380 Text feature [activating] present in test data point [True]
Out of the top  500  features  6 are present in query point
```

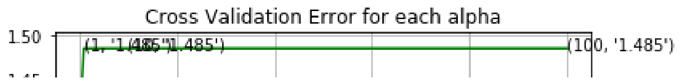
4.4. Linear Support Vector Machines

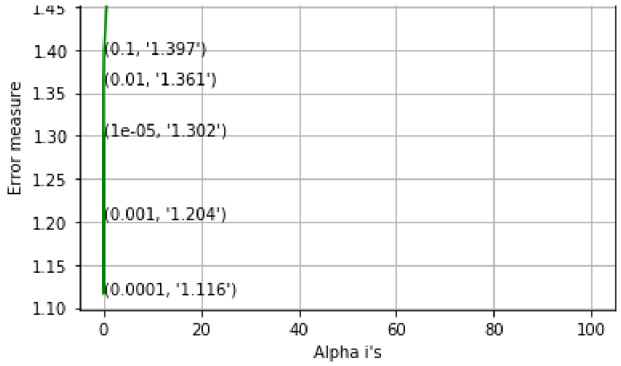
4.4.1. Hyper paramter tuning

In [133]:

```
# read more about support vector machines with linear kernals here http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

# --------------------------------
# default parameters
# SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001,
# cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)

# Some of methods of SVM()
# fit(X, y, [sample_weight]) Fit the SVM model according to the given training data.
# predict(X) Perform classification on samples in X.
# --------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/mathematical-derivation-copy-8/
# --------------------------------



# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
#-------------------------------------
# video link:
#-------------------------------------

alpha = [10 ** x for x in range(-5, 3)]
cv_log_error_array = []
for i in alpha:
    print("for C =", i)
#     clf = SVC(C=i,kernel='linear',probability=True, class_weight='balanced')
    clf = SGDClassifier( class_weight='balanced', alpha=i, penalty='l2', loss='hinge', random_state=42)
    clf.fit(train_x_onehotCoding, train_y)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x_onehotCoding, train_y)
    sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
    cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    print("Log Loss :",log_loss(cv_y, sig_clf_probs))

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],str(txt)), (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()


best_alpha = np.argmin(cv_log_error_array)
# clf = SVC(C=i,kernel='linear',probability=True, class_weight='balanced')
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='hinge', rando
m_state=42)
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train, pred
ict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:",log_loss(y_
cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, predic
t_y, labels=clf.classes_, eps=1e-15))
```

```
for C = 1e-05
Log Loss : 1.30177913153494
for C = 0.0001
Log Loss : 1.116109230941168
for C = 0.001
Log Loss : 1.203770094270132
for C = 0.01
Log Loss : 1.3612572289385194
for C = 0.1
Log Loss : 1.3972542814368751
for C = 1
Log Loss : 1.4849964781985459
for C = 10
Log Loss : 1.4851437992024463
for C = 100
Log Loss : 1.4851437964276049
```





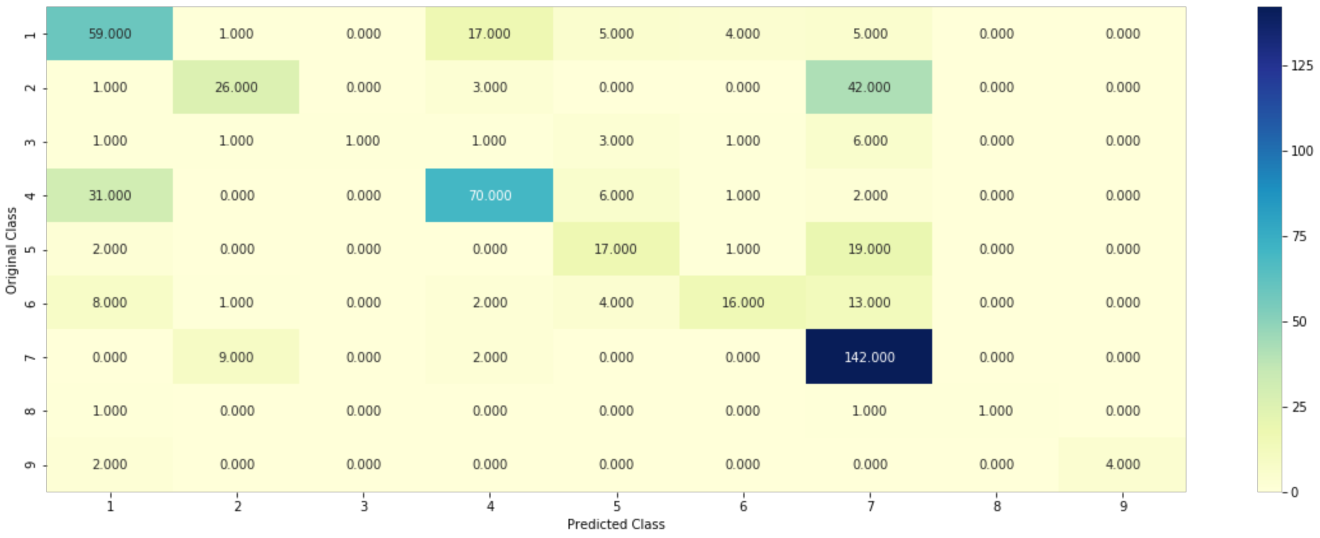
```
For values of best alpha =  0.0001 The train log loss is: 0.45938566660559854
For values of best alpha =  0.0001 The cross validation log loss is: 1.116109230941168
For values of best alpha =  0.0001 The test log loss is: 1.1156627743769747
```

4.4.2. Testing model with best hyper parameters

In [134]:

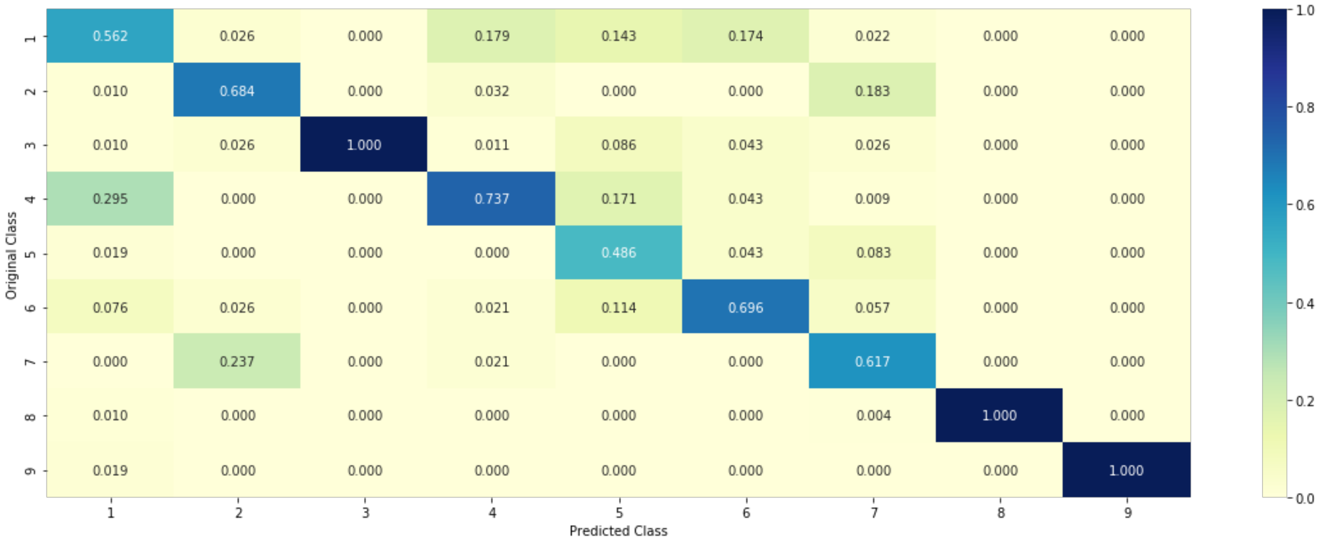
```
# read more about support vector machines with linear kernals here http://scikit-learn.org/stable/modul
es/generated/sklearn.svm.SVC.html

# --------------------------------
# default parameters
# SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.
001,
# cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_
state=None)

# Some of methods of SVM()
# fit(X, y, [sample_weight]) Fit the SVM model according to the given training data.
# predict(X) Perform classification on samples in X.
# --------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/mathematical-deri
vation-copy-8/
# --------------------------------


# clf = SVC(C=alpha[best_alpha],kernel='linear',probability=True, class_weight='balanced')
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='hinge', random_state=42,class_weight='
balanced')
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y,cv_x_onehotCoding,cv_y, clf)
```

```
Log loss : 1.116109230941168
Number of mis-classified points : 0.3684210526315789
-------------------- Confusion matrix --------------------
```

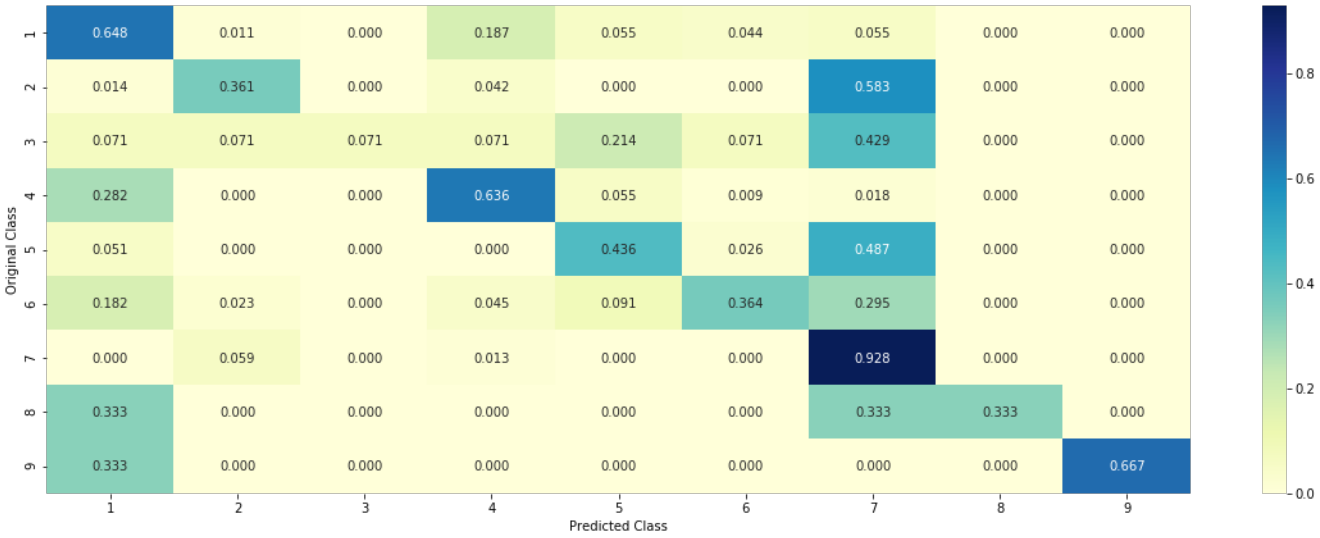


	1	2	3	4	5	6	7	8	9
1	59.000	1.000	0.000	17.000	5.000	4.000	5.000	0.000	0.000
2	1.000	26.000	0.000	3.000	0.000	0.000	42.000	0.000	0.000
3	1.000	1.000	1.000	1.000	3.000	1.000	6.000	0.000	0.000
4	31.000	0.000	0.000	70.000	6.000	1.000	2.000	0.000	0.000
5	2.000	0.000	0.000	0.000	17.000	1.000	19.000	0.000	0.000
6	8.000	1.000	0.000	2.000	4.000	16.000	13.000	0.000	0.000
7	0.000	9.000	0.000	2.000	0.000	0.000	142.000	0.000	0.000
8	1.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
9	2.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	4.000

```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



```
-------------------- Recall matrix (Row sum=1) --------------------
```



4.3.3. Feature Importance

4.3.3.1. For Correctly classified point

In [135]:

```
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='hinge', random_state=42)
clf.fit(train_x_onehotCoding,train_y)
test_point_index = 1
# test_point_index = 100
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point
_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 6
Predicted Class Probabilities: [[0.0128 0.0011 0.0059 0.0118 0.0297 0.9028 0.0289 0.002  0.0049]]
Predicted Class Probabilities: [[0.0128 0.0011 0.0059 0.0118 0.0297 0.9028 0.0289 0.002  0.0049]]
Actual Class : 6
--------------------------------------------------
77 Text feature [substitutions] present in test data point [True]
88 Text feature [fell] present in test data point [True]
109 Text feature [expected] present in test data point [True]
110 Text feature [predicted] present in test data point [True]
111 Text feature [16c] present in test data point [True]
112 Text feature [i124v] present in test data point [True]
113 Text feature [071] present in test data point [True]
114 Text feature [ovarian] present in test data point [True]
115 Text feature [ivs13] present in test data point [True]
116 Text feature [ethnic] present in test data point [True]
127 Text feature [000] present in test data point [True]
128 Text feature [loss] present in test data point [True]
131 Text feature [mastectomy] present in test data point [True]
134 Text feature [20c] present in test data point [True]
135 Text feature [multiplying] present in test data point [True]
136 Text feature [i2285v] present in test data point [True]
173 Text feature [brca] present in test data point [True]
174 Text feature [dcis] present in test data point [True]
175 Text feature [model] present in test data point [True]
176 Text feature [practice] present in test data point [True]
177 Text feature [y105c] present in test data point [True]
178 Text feature [quantifiable] present in test data point [True]
179 Text feature [440] present in test data point [True]
180 Text feature [programming] present in test data point [True]
181 Text feature [p142h] present in test data point [True]
182 Text feature [ivs8] present in test data point [True]
183 Text feature [ivs5] present in test data point [True]
184 Text feature [basis] present in test data point [True]
185 Text feature [7g] present in test data point [True]
186 Text feature [hampered] present in test data point [True]
188 Text feature [ors] present in test data point [True]
213 Text feature [bic] present in test data point [True]
214 Text feature [s1172l] present in test data point [True]
215 Text feature [ivs20] present in test data point [True]
216 Text feature [multiplied] present in test data point [True]
217 Text feature [resource] present in test data point [True]
219 Text feature [abkevich] present in test data point [True]
220 Text feature [685] present in test data point [True]
221 Text feature [assumption] present in test data point [True]
222 Text feature [heretofore] present in test data point [True]
223 Text feature [923] present in test data point [True]
224 Text feature [57] present in test data point [True]
225 Text feature [deleterious] present in test data point [True]
226 Text feature [08] present in test data point [True]
227 Text feature [specialists] present in test data point [True]
228 Text feature [brca1] present in test data point [True]
229 Text feature [threshold] present in test data point [True]
231 Text feature [ivs18] present in test data point [True]
234 Text feature [k2950n] present in test data point [True]
235 Text feature [k513r] present in test data point [True]
237 Text feature [alignments] present in test data point [True]
238 Text feature [054] present in test data point [True]
239 Text feature [74] present in test data point [True]
240 Text feature [lrs] present in test data point [True]
241 Text feature [classify] present in test data point [True]
243 Text feature [heterozygotes] present in test data point [True]
244 Text feature [odds] present in test data point [True]
245 Text feature [979] present in test data point [True]
246 Text feature [classified] present in test data point [True]
268 Text feature [predictive] present in test data point [True]
269 Text feature [discussionusing] present in test data point [True]
270 Text feature [3098] present in test data point [True]
271 Text feature [personal] present in test data point [True]
272 Text feature [scores] present in test data point [True]
273 Text feature [79] present in test data point [True]
276 Text feature [likely] present in test data point [True]
277 Text feature [uncertain] present in test data point [True]
278 Text feature [subdivided] present in test data point [True]
279 Text feature [logistic] present in test data point [True]
280 Text feature [conserved] present in test data point [True]
281 Text feature [20t] present in test data point [True]
282 Text feature [latin] present in test data point [True]
283 Text feature [histogram] present in test data point [True]
284 Text feature [logically] present in test data point [True]
285 Text feature [missense] present in test data point [True]
```

```
285 Text feature [missense] present in test data point [True]
286 Text feature [lr] present in test data point [True]
287 Text feature [trans] present in test data point [True]
288 Text feature [02] present in test data point [True]
289 Text feature [outside] present in test data point [True]
290 Text feature [vanishingly] present in test data point [True]
291 Text feature [feel] present in test data point [True]
292 Text feature [disciplines] present in test data point [True]
293 Text feature [nonmammalian] present in test data point [True]
294 Text feature [evidence] present in test data point [True]
295 Text feature [carry] present in test data point [True]
297 Text feature [493] present in test data point [True]
298 Text feature [q804h] present in test data point [True]
299 Text feature [neutral] present in test data point [True]
300 Text feature [caribbean] present in test data point [True]
302 Text feature [history] present in test data point [True]
303 Text feature [ascertained] present in test data point [True]
304 Text feature [reassured] present in test data point [True]
305 Text feature [433] present in test data point [True]
306 Text feature [antoniou] present in test data point [True]
307 Text feature [conservation] present in test data point [True]
308 Text feature [772] present in test data point [True]
309 Text feature [088] present in test data point [True]
310 Text feature [gvgd] present in test data point [True]
311 Text feature [summaries] present in test data point [True]
312 Text feature [problematic] present in test data point [True]
313 Text feature [stata] present in test data point [True]
314 Text feature [regressions] present in test data point [True]
316 Text feature [l668f] present in test data point [True]
317 Text feature [s1101n] present in test data point [True]
318 Text feature [n810y] present in test data point [True]
319 Text feature [e1214k] present in test data point [True]
320 Text feature [v191i] present in test data point [True]
321 Text feature [r866c] present in test data point [True]
322 Text feature [n132k] present in test data point [True]
323 Text feature [1010] present in test data point [True]
324 Text feature [ifdis] present in test data point [True]
325 Text feature [ivs19] present in test data point [True]
326 Text feature [mvs] present in test data point [True]
327 Text feature [ifdi] present in test data point [True]
328 Text feature [n2113s] present in test data point [True]
329 Text feature [r3052q] present in test data point [True]
330 Text feature [g1194d] present in test data point [True]
331 Text feature [characteristicsno] present in test data point [True]
332 Text feature [0other] present in test data point [True]
333 Text feature [sourceturn] present in test data point [True]
334 Text feature [n2048i] present in test data point [True]
335 Text feature [a622v] present in test data point [True]
336 Text feature [s1733f] present in test data point [True]
337 Text feature [n1468h] present in test data point [True]
338 Text feature [r1495m] present in test data point [True]
339 Text feature [q2384k] present in test data point [True]
340 Text feature [w2626c] present in test data point [True]
341 Text feature [11delt] present in test data point [True]
342 Text feature [c3198r] present in test data point [True]
343 Text feature [e1419q] present in test data point [True]
344 Text feature [i925l] present in test data point [True]
345 Text feature [isrk] present in test data point [True]
346 Text feature [g602r] present in test data point [True]
347 Text feature [formview] present in test data point [True]
348 Text feature [onandview] present in test data point [True]
349 Text feature [individualbrca1] present in test data point [True]
350 Text feature [onfor] present in test data point [True]
351 Text feature [onlet] present in test data point [True]
352 Text feature [s1266t] present in test data point [True]
353 Text feature [a1623g] present in test data point [True]
354 Text feature [n2436i] present in test data point [True]
355 Text feature [k2411t] present in test data point [True]
356 Text feature [2to] present in test data point [True]
357 Text feature [108table] present in test data point [True]
358 Text feature [optionsoverall] present in test data point [True]
359 Text feature [isview] present in test data point [True]
360 Text feature [q1396r] present in test data point [True]
361 Text feature [q155e] present in test data point [True]
362 Text feature [y3092c] present in test data point [True]
363 Text feature [onthat] present in test data point [True]
364 Text feature [onthenview] present in test data point [True]
365 Text feature [[illegible]] present in test data point [True]
```

```
365 Text feature [e1682k] present in test data point [True]
366 Text feature [v2969m] present in test data point [True]
367 Text feature [k1109n] present in test data point [True]
368 Text feature [24brca2] present in test data point [True]
369 Text feature [onwhere] present in test data point [True]
370 Text feature [n56t] present in test data point [True]
371 Text feature [onthis] present in test data point [True]
372 Text feature [r1190w] present in test data point [True]
373 Text feature [classto] present in test data point [True]
374 Text feature [1225del3] present in test data point [True]
375 Text feature [21table] present in test data point [True]
376 Text feature [r1203q] present in test data point [True]
377 Text feature [v3079i] present in test data point [True]
378 Text feature [a280g] present in test data point [True]
379 Text feature [genotypeethnic] present in test data point [True]
380 Text feature [f1524v] present in test data point [True]
381 Text feature [brca2personal] present in test data point [True]
382 Text feature [aassessed] present in test data point [True]
383 Text feature [n1102y] present in test data point [True]
384 Text feature [103splice] present in test data point [True]
385 Text feature [9345g] present in test data point [True]
386 Text feature [gtrag] present in test data point [True]
387 Text feature [f1662s] present in test data point [True]
388 Text feature [h2074n] present in test data point [True]
389 Text feature [e597k] present in test data point [True]
390 Text feature [104brca2] present in test data point [True]
391 Text feature [2ln] present in test data point [True]
392 Text feature [e842g] present in test data point [True]
393 Text feature [n1228d] present in test data point [True]
394 Text feature [n473s] present in test data point [True]
395 Text feature [mutationsto] present in test data point [True]
396 Text feature [ontherefore] present in test data point [True]
397 Text feature [optionsfrequency] present in test data point [True]
398 Text feature [2del21insa] present in test data point [True]
399 Text feature [k1690n] present in test data point [True]
400 Text feature [2all] present in test data point [True]
401 Text feature [n723d] present in test data point [True]
402 Text feature [a75p] present in test data point [True]
403 Text feature [ivs25] present in test data point [True]
404 Text feature [r1028h] present in test data point [True]
405 Text feature [h1918y] present in test data point [True]
406 Text feature [optionsas] present in test data point [True]
407 Text feature [g890v] present in test data point [True]
408 Text feature [probandsno] present in test data point [True]
409 Text feature [v1247i] present in test data point [True]
410 Text feature [d806h] present in test data point [True]
411 Text feature [l1019v] present in test data point [True]
412 Text feature [5181del3] present in test data point [True]
413 Text feature [t630i] present in test data point [True]
414 Text feature [probandsor] present in test data point [True]
415 Text feature [d1280v] present in test data point [True]
416 Text feature [i1349t] present in test data point [True]
417 Text feature [neutralitybrca1] present in test data point [True]
418 Text feature [t582p] present in test data point [True]
419 Text feature [p1238l] present in test data point [True]
420 Text feature [4603g] present in test data point [True]
421 Text feature [age2] present in test data point [True]
422 Text feature [overstate] present in test data point [True]
423 Text feature [deletioneuropeana] present in test data point [True]
424 Text feature [pedigreeselsewhere] present in test data point [True]
425 Text feature [invarianta] present in test data point [True]
426 Text feature [byview] present in test data point [True]
427 Text feature [v1306i] present in test data point [True]
428 Text feature [causalityllrgene] present in test data point [True]
429 Text feature [p1859r] present in test data point [True]
430 Text feature [reassurance] present in test data point [True]
431 Text feature [neutralityllrgene] present in test data point [True]
432 Text feature [l2396f] present in test data point [True]
433 Text feature [m1361l] present in test data point [True]
434 Text feature [d3170g] present in test data point [True]
435 Text feature [p168t] present in test data point [True]
436 Text feature [17g] present in test data point [True]
437 Text feature [l1904v] present in test data point [True]
438 Text feature [i1929v] present in test data point [True]
439 Text feature [analyzedthe] present in test data point [True]
440 Text feature [m297i] present in test data point [True]
441 Text feature [p1819s] present in test data point [True]
```

```
442 Text feature [d420y] present in test data point [True]
443 Text feature [v894i] present in test data point [True]
444 Text feature [r2842h] present in test data point [True]
445 Text feature [d642h] present in test data point [True]
446 Text feature [g1529r] present in test data point [True]
447 Text feature [i1405v] present in test data point [True]
448 Text feature [d1546y] present in test data point [True]
449 Text feature [4987c] present in test data point [True]
450 Text feature [t3349a] present in test data point [True]
451 Text feature [d369n] present in test data point [True]
452 Text feature [probandswe] present in test data point [True]
453 Text feature [or7] present in test data point [True]
454 Text feature [or5] present in test data point [True]
455 Text feature [methodssource] present in test data point [True]
456 Text feature [g1771d] present in test data point [True]
457 Text feature [causalitybrca1] present in test data point [True]
458 Text feature [k862e] present in test data point [True]
459 Text feature [63mv] present in test data point [True]
460 Text feature [straightforwardly] present in test data point [True]
461 Text feature [or2] present in test data point [True]
462 Text feature [thereforeview] present in test data point [True]
463 Text feature [r2108h] present in test data point [True]
464 Text feature [p334l] present in test data point [True]
465 Text feature [c554w] present in test data point [True]
466 Text feature [aspersonal] present in test data point [True]
467 Text feature [typehistory] present in test data point [True]
468 Text feature [p375s] present in test data point [True]
469 Text feature [r504h] present in test data point [True]
470 Text feature [andreported] present in test data point [True]
471 Text feature [llr] present in test data point [True]
472 Text feature [ordiagnosis] present in test data point [True]
473 Text feature [p3039p] present in test data point [True]
474 Text feature [t1349m] present in test data point [True]
475 Text feature [i1858l] present in test data point [True]
476 Text feature [llrvariant] present in test data point [True]
477 Text feature [r2502c] present in test data point [True]
478 Text feature [r2888c] present in test data point [True]
479 Text feature [i1275v] present in test data point [True]
480 Text feature [t1354m] present in test data point [True]
481 Text feature [y3098h] present in test data point [True]
482 Text feature [llrs] present in test data point [True]
483 Text feature [i1044v] present in test data point [True]
484 Text feature [t2250a] present in test data point [True]
485 Text feature [16ain] present in test data point [True]
486 Text feature [37the] present in test data point [True]
487 Text feature [asview] present in test data point [True]
488 Text feature [c1365y] present in test data point [True]
489 Text feature [t1685a] present in test data point [True]
490 Text feature [pj] present in test data point [True]
491 Text feature [qj] present in test data point [True]
492 Text feature [10g] present in test data point [True]
493 Text feature [provider] present in test data point [True]
494 Text feature [alike] present in test data point [True]
495 Text feature [ivs24] present in test data point [True]
496 Text feature [partitions] present in test data point [True]
497 Text feature [ivs12] present in test data point [True]
498 Text feature [639] present in test data point [True]
Out of the top  500  features  285 are present in query point
```

4.3.3.2. For Incorrectly classified point

In [136]:

```
test_point_index = 2
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point
_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.052  0.0534 0.024  0.0988 0.045  0.0375 0.677  0.0048 0.0075]]
Actual Class : 6
--------------------------------------------------
5 Text feature [activated] present in test data point [True]
23 Text feature [downstream] present in test data point [True]
402 Text feature [useful] present in test data point [True]
403 Text feature [inhibitor] present in test data point [True]
407 Text feature [activating] present in test data point [True]
408 Text feature [treated] present in test data point [True]
Out of the top  500  features  6 are present in query point
```

4.5 Random Forest Classifier

4.5.1. Hyper paramter tuning (With One hot Encoding)

In [137]:

```python
# --------------------------------
# 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/
# --------------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
#-------------------------------------
# video link:
#-------------------------------------

alpha = [100,200,500,1000,2000]
max_depth = [5, 10]
cv_log_error_array = []
for i in alpha:
    for j in max_depth:
        print("for n_estimators =", i,"and max depth = ", j)
        clf = RandomForestClassifier(n_estimators=i, criterion='gini', max_depth=j, random_state=42, n_jobs=-1)
        clf.fit(train_x_onehotCoding, train_y)
        sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
        sig_clf.fit(train_x_onehotCoding, train_y)
        sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
        cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.classes_, eps=1e-15))
        print("Log Loss :",log_loss(cv_y, sig_clf_probs))
        print("Log Loss :",log_loss(cv_y, sig_clf_probs))

'''fig, ax = plt.subplots()
features = np.dot(np.array(alpha)[:,None],np.array(max_depth)[None]).ravel()
ax.plot(features, cv_log_error_array,c='g')
for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[int(i/2)],max_depth[int(i%2)],str(txt)), (features[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()
'''

best_alpha = np.argmin(cv_log_error_array)
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_depth=max_dep
th[int(best_alpha%2)], random_state=42, n_jobs=-1)
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best estimator = ', alpha[int(best_alpha/2)], "The train log loss is:",log_loss(y_
train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best estimator = ', alpha[int(best_alpha/2)], "The cross validation log loss is:",
log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best estimator = ', alpha[int(best_alpha/2)], "The test log loss is:",log_loss(y_t
est, predict_y, labels=clf.classes_, eps=1e-15))
```

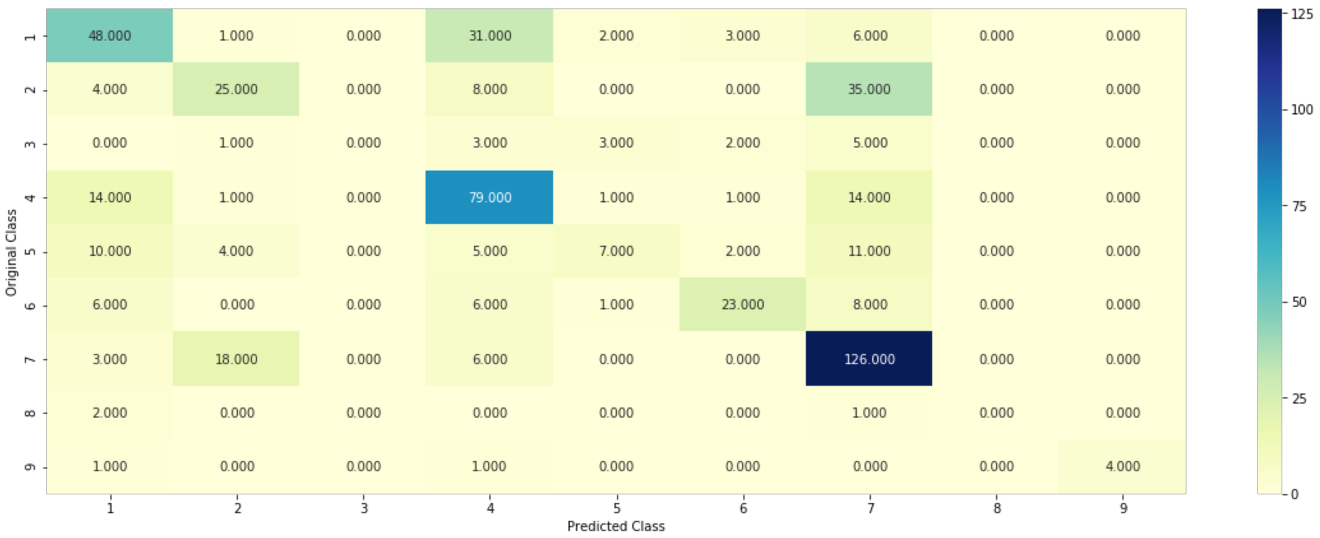
```
for n_estimators = 100 and max depth =  5
Log Loss : 1.2901813108719098
for n_estimators = 100 and max depth =  10
Log Loss : 1.2041171344002752
for n_estimators = 200 and max depth =  5
Log Loss : 1.279543059942447
for n_estimators = 200 and max depth =  10
Log Loss : 1.1953662271781487
for n_estimators = 500 and max depth =  5
Log Loss : 1.2688548147691339
for n_estimators = 500 and max depth =  10
Log Loss : 1.1920693136092997
for n_estimators = 1000 and max depth =  5
Log Loss : 1.263706601382646
for n_estimators = 1000 and max depth =  10
Log Loss : 1.1908752815473447
for n_estimators = 2000 and max depth =  5
Log Loss : 1.2616185903524488
for n_estimators = 2000 and max depth =  10
Log Loss : 1.189238145561043
For values of best estimator =  2000 The train log loss is: 0.7728556863323316
For values of best estimator =  2000 The cross validation log loss is: 1.189238145561043
For values of best estimator =  2000 The test log loss is: 1.1840677562504702
```

4.5.2. Testing model with best hyper parameters (One Hot Encoding)

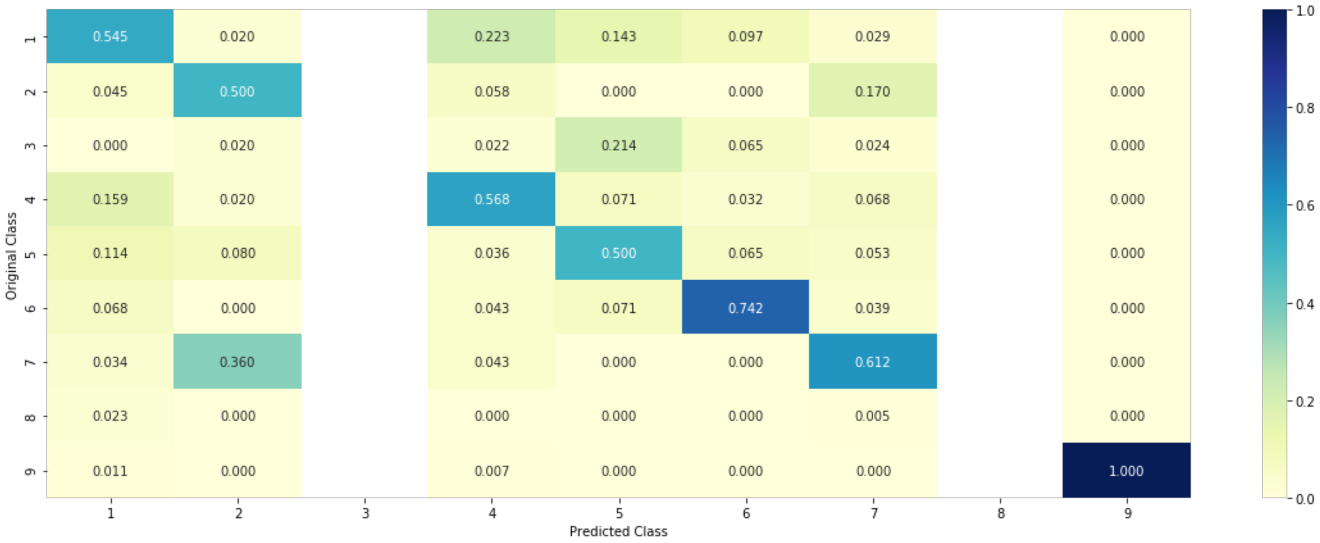
In [138]:

```
# --------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_sample
s_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impur
ity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, war
m_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/
# --------------------------------

clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_depth=max_dep
th[int(best_alpha%2)], random_state=42, n_jobs=-1)
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y,cv_x_onehotCoding,cv_y, clf)
```

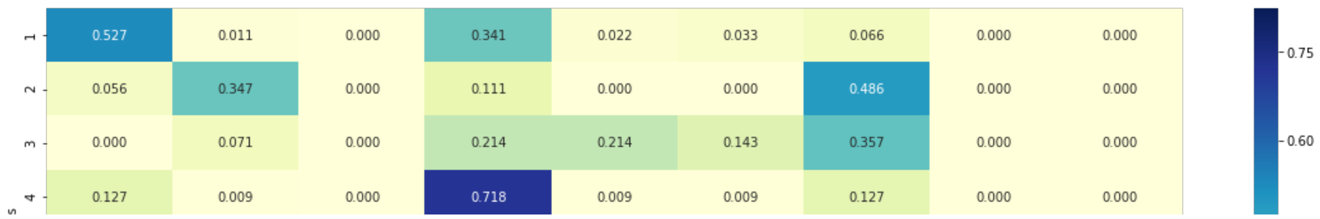
```
Log loss : 1.189238145561043
Number of mis-classified points : 0.41353383458646614
-------------------- Confusion matrix --------------------
```

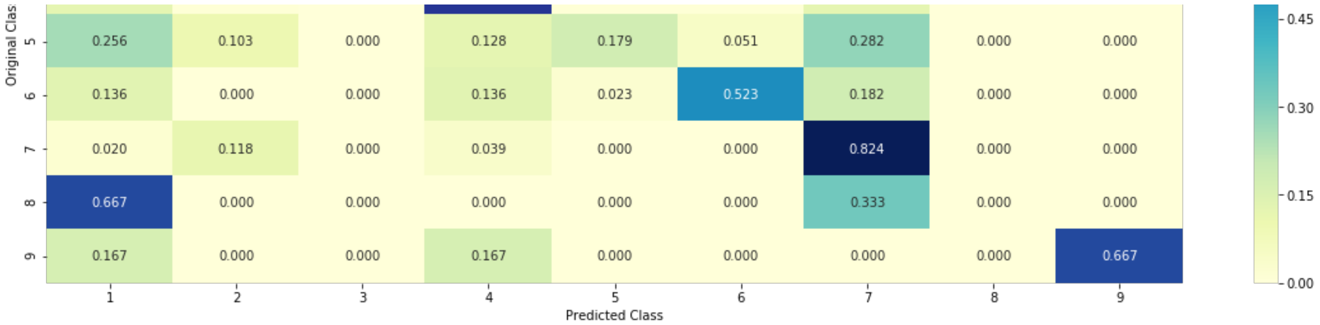


```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



```
-------------------- Recall matrix (Row sum=1) --------------------
```





4.5.3. Feature Importance

4.5.3.1. Correctly Classified point

In [139]:

```
# test_point_index = 10
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_depth=max_dep
th[int(best_alpha%2)], random_state=42, n_jobs=-1)
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)

test_point_index = 1
no_feature = 100
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.feature_importances_)
print("-"*50)
get_impfeature_names(indices[:no_feature], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[
test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 6
Predicted Class Probabilities: [[0.0118 0.002  0.0111 0.0106 0.0863 0.8659 0.0039 0.0038 0.0046]]
Actual Class : 6
--------------------------------------------------
4 Text feature [function] present in test data point [True]
7 Text feature [loss] present in test data point [True]
8 Text feature [missense] present in test data point [True]
10 Text feature [functional] present in test data point [True]
11 Text feature [protein] present in test data point [True]
12 Text feature [patients] present in test data point [True]
15 Text feature [variants] present in test data point [True]
17 Text feature [predicted] present in test data point [True]
18 Text feature [variant] present in test data point [True]
21 Text feature [deleterious] present in test data point [True]
23 Text feature [brca1] present in test data point [True]
25 Text feature [likely] present in test data point [True]
26 Text feature [substitutions] present in test data point [True]
27 Text feature [assays] present in test data point [True]
28 Text feature [conserved] present in test data point [True]
29 Text feature [based] present in test data point [True]
30 Text feature [sequence] present in test data point [True]
31 Text feature [expected] present in test data point [True]
33 Text feature [classified] present in test data point [True]
34 Text feature [useful] present in test data point [True]
35 Text feature [family] present in test data point [True]
36 Text feature [neutral] present in test data point [True]
37 Text feature [26] present in test data point [True]
38 Text feature [likelihood] present in test data point [True]
40 Text feature [risk] present in test data point [True]
41 Text feature [classify] present in test data point [True]
42 Text feature [evidence] present in test data point [True]
43 Text feature [model] present in test data point [True]
44 Text feature [ovarian] present in test data point [True]
45 Text feature [conservation] present in test data point [True]
```

```
46 Text feature [probability] present in test data point [True]
47 Text feature [alignments] present in test data point [True]
48 Text feature [brca2] present in test data point [True]
49 Text feature [basis] present in test data point [True]
51 Text feature [carry] present in test data point [True]
52 Text feature [bic] present in test data point [True]
53 Text feature [43] present in test data point [True]
54 Text feature [49] present in test data point [True]
55 Text feature [estimated] present in test data point [True]
56 Text feature [uncertain] present in test data point [True]
57 Text feature [57] present in test data point [True]
58 Text feature [article] present in test data point [True]
59 Text feature [splicing] present in test data point [True]
61 Text feature [000] present in test data point [True]
62 Text feature [combined] present in test data point [True]
63 Text feature [history] present in test data point [True]
64 Text feature [p142h] present in test data point [True]
65 Text feature [laboratories] present in test data point [True]
66 Text feature [cosegregation] present in test data point [True]
67 Text feature [odds] present in test data point [True]
68 Text feature [counseling] present in test data point [True]
69 Text feature [evolutionarily] present in test data point [True]
70 Text feature [104] present in test data point [True]
71 Text feature [outside] present in test data point [True]
72 Text feature [probabilities] present in test data point [True]
73 Text feature [sufficiently] present in test data point [True]
74 Text feature [00] present in test data point [True]
75 Text feature [personal] present in test data point [True]
76 Text feature [predictive] present in test data point [True]
77 Text feature [brca] present in test data point [True]
78 Text feature [occurrence] present in test data point [True]
79 Text feature [threshold] present in test data point [True]
80 Text feature [scores] present in test data point [True]
81 Text feature [56] present in test data point [True]
82 Text feature [resource] present in test data point [True]
84 Text feature [practice] present in test data point [True]
85 Text feature [favor] present in test data point [True]
87 Text feature [79] present in test data point [True]
89 Text feature [74] present in test data point [True]
90 Text feature [gvgd] present in test data point [True]
93 Text feature [99] present in test data point [True]
94 Text feature [align] present in test data point [True]
95 Text feature [sources] present in test data point [True]
96 Text feature [trans] present in test data point [True]
97 Text feature [i124v] present in test data point [True]
98 Text feature [powerful] present in test data point [True]
99 Text feature [predispose] present in test data point [True]
Out of the top  100  features  77 are present in query point
```

4.5.3.2. Incorrectly Classified point

In [140]:

```
test_point_index = 2
no_feature = 100
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_onehotCoding[test_point_i
ndex]),4))
print("Actuall Class :", test_y[test_point_index])
indices = np.argsort(-clf.feature_importances_)
print("-"*50)
get_impfeature_names(indices[:no_feature], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[
test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.0736 0.1522 0.0305 0.2177 0.0565 0.0459 0.411  0.0059 0.0069]]
Actuall Class : 6
--------------------------------------------------
0 Text feature [kinase] present in test data point [True]
1 Text feature [activating] present in test data point [True]
2 Text feature [inhibitor] present in test data point [True]
3 Text feature [activated] present in test data point [True]
4 Text feature [function] present in test data point [True]
```

```
4 Text feature [function] present in test data point [True]
5 Text feature [treatment] present in test data point [True]
7 Text feature [loss] present in test data point [True]
9 Text feature [downstream] present in test data point [True]
10 Text feature [functional] present in test data point [True]
11 Text feature [protein] present in test data point [True]
12 Text feature [patients] present in test data point [True]
13 Text feature [treated] present in test data point [True]
15 Text feature [variants] present in test data point [True]
16 Text feature [pten] present in test data point [True]
17 Text feature [predicted] present in test data point [True]
24 Text feature [assay] present in test data point [True]
25 Text feature [likely] present in test data point [True]
26 Text feature [substitutions] present in test data point [True]
27 Text feature [assays] present in test data point [True]
29 Text feature [based] present in test data point [True]
31 Text feature [expected] present in test data point [True]
33 Text feature [classified] present in test data point [True]
34 Text feature [useful] present in test data point [True]
37 Text feature [26] present in test data point [True]
40 Text feature [risk] present in test data point [True]
42 Text feature [evidence] present in test data point [True]
43 Text feature [model] present in test data point [True]
51 Text feature [carry] present in test data point [True]
53 Text feature [43] present in test data point [True]
54 Text feature [49] present in test data point [True]
60 Text feature [differential] present in test data point [True]
62 Text feature [combined] present in test data point [True]
71 Text feature [outside] present in test data point [True]
98 Text feature [powerful] present in test data point [True]
Out of the top  100  features  34 are present in query point
```

4.5.3. Hyper paramter tuning (With Response Coding)

In [141]:

```
# --------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_sample
s_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impur
ity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, war
m_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/
# --------------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklea
rn.calibration.CalibratedClassifierCV.html
# ----------------------------
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, method='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight]) Fit the calibrated model
# get_params([deep]) Get parameters for this estimator.
# predict(X) Predict the target of new samples.
# predict_proba(X) Posterior probabilities of classification
#-------------------------------------
# video link:
#-------------------------------------
```

```
alpha = [10,50,100,200,500,1000]
max_depth = [2,3,5,10]
cv_log_error_array = []
for i in alpha:
    for j in max_depth:
        print("for n_estimators =", i,"and max depth = ", j)
        clf = RandomForestClassifier(n_estimators=i, criterion='gini', max_depth=j, random_state=42, n_jobs=-1)
        clf.fit(train_x_responseCoding, train_y)
        sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
        sig_clf.fit(train_x_responseCoding, train_y)
        sig_clf_probs = sig_clf.predict_proba(cv_x_responseCoding)
        cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.classes_, eps=1e-15))
        print("Log Loss :",log_loss(cv_y, sig_clf_probs))
'''
fig, ax = plt.subplots()
features = np.dot(np.array(alpha)[:,None],np.array(max_depth)[None]).ravel()
ax.plot(features, cv_log_error_array,c='g')
for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[int(i/4)],max_depth[int(i%4)],str(txt)), (features[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()
'''

best_alpha = np.argmin(cv_log_error_array)
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/4)], criterion='gini', max_depth=max_depth[int(best_alpha%4)], random_state=42, n_jobs=-1)
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_responseCoding)
print('For values of best alpha = ', alpha[int(best_alpha/4)], "The train log loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(cv_x_responseCoding)
print('For values of best alpha = ', alpha[int(best_alpha/4)], "The cross validation log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_responseCoding)
print('For values of best alpha = ', alpha[int(best_alpha/4)], "The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

```
for n_estimators = 10 and max depth =  2
Log Loss : 2.115807468495449
for n_estimators = 10 and max depth =  3
Log Loss : 1.880987021470951
for n_estimators = 10 and max depth =  5
Log Loss : 1.5208114591939843
for n_estimators = 10 and max depth =  10
Log Loss : 2.159416839044755
for n_estimators = 50 and max depth =  2
Log Loss : 1.7606914108165852
for n_estimators = 50 and max depth =  3
Log Loss : 1.4684144886744233
for n_estimators = 50 and max depth =  5
Log Loss : 1.4441650609842154
for n_estimators = 50 and max depth =  10
Log Loss : 1.7537827827013042
for n_estimators = 100 and max depth =  2
Log Loss : 1.5783248728005936
for n_estimators = 100 and max depth =  3
Log Loss : 1.453351834023957
for n_estimators = 100 and max depth =  5
Log Loss : 1.406642649745641
for n_estimators = 100 and max depth =  10
Log Loss : 1.8049496864596726
for n_estimators = 200 and max depth =  2
Log Loss : 1.6190726280020316
for n_estimators = 200 and max depth =  3
Log Loss : 1.4438681510920974
for n_estimators = 200 and max depth =  5
Log Loss : 1.4527470472663297
for n_estimators = 200 and max depth =  10
```

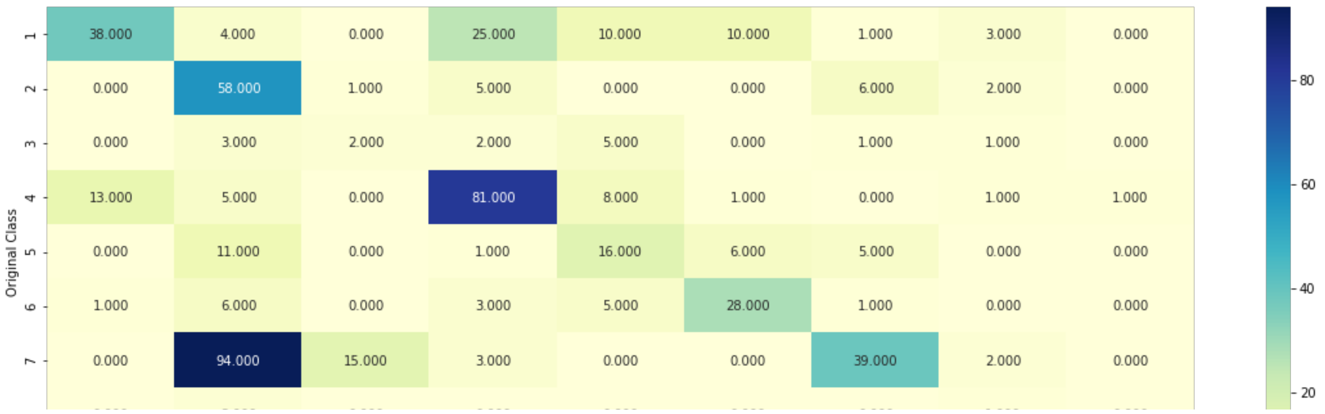
```
Log Loss : 1.7743624814385526
for n_estimators = 500 and max depth =  2
Log Loss : 1.6889838306117022
for n_estimators = 500 and max depth =  3
Log Loss : 1.5091203518929073
for n_estimators = 500 and max depth =  5
Log Loss : 1.4477524281617966
for n_estimators = 500 and max depth =  10
Log Loss : 1.7963611293720925
for n_estimators = 1000 and max depth =  2
Log Loss : 1.639699143823005
for n_estimators = 1000 and max depth =  3
Log Loss : 1.518971124246746
for n_estimators = 1000 and max depth =  5
Log Loss : 1.4586401883886473
for n_estimators = 1000 and max depth =  10
Log Loss : 1.7604238462483834
For values of best alpha =  100 The train log loss is: 0.0555113394763 61984
For values of best alpha =  100 The cross validation log loss is: 1.406642649745641
For values of best alpha =  100 The test log loss is: 1.3875321091912167
```

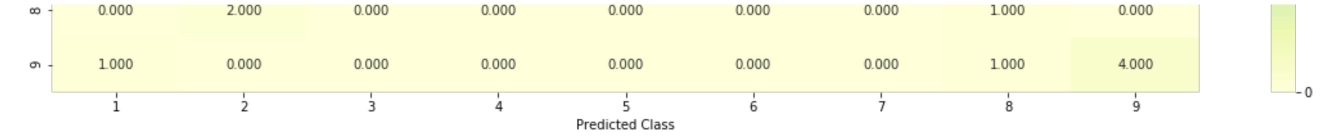
4.5.4. Testing model with best hyper parameters (Response Coding)

In [142]:

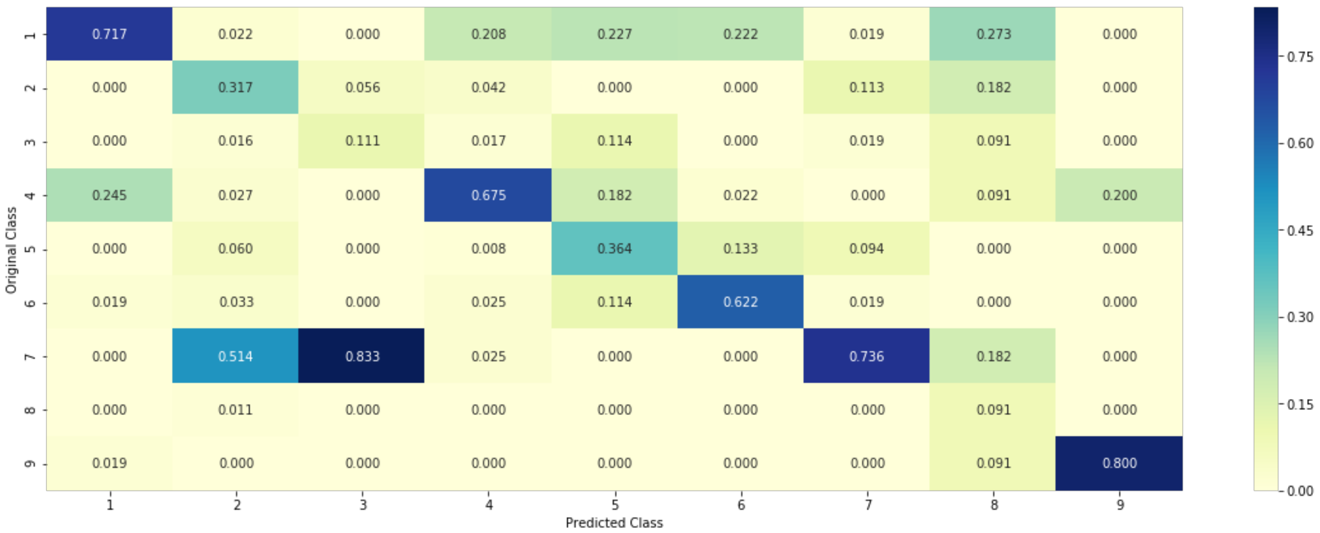
```
# --------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_sample
s_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impur
ity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, war
m_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/
# --------------------------------

clf = RandomForestClassifier(max_depth=max_depth[int(best_alpha%4)], n_estimators=alpha[int(best_alpha/
4)], criterion='gini', max_features='auto',random_state=42)
predict_and_plot_confusion_matrix(train_x_responseCoding, train_y,cv_x_responseCoding,cv_y, clf)
```

```
Log loss : 1.4066426497456408
Number of mis-classified points : 0.4981203007518797
-------------------- Confusion matrix --------------------
```

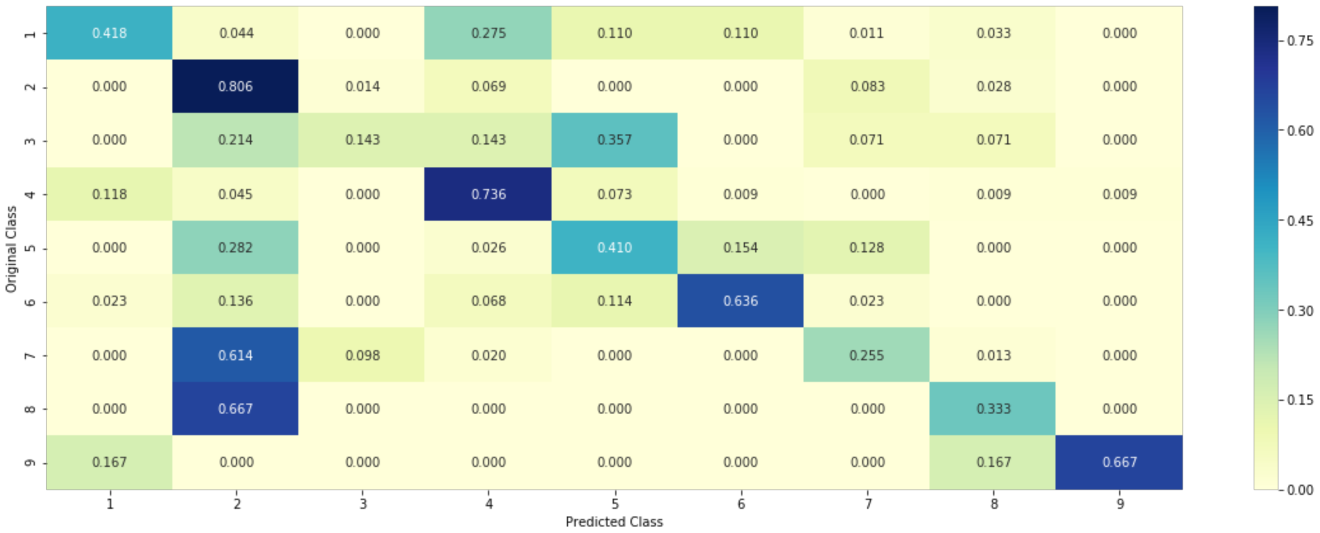




```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



```
-------------------- Recall matrix (Row sum=1) --------------------
```



4.5.5. Feature Importance

4.5.5.1. Correctly Classified point

In [143]:

```
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/4)], criterion='gini', max_depth=max_dep
th[int(best_alpha%4)], random_state=42, n_jobs=-1)
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)


test_point_index = 1
no_feature = 27
predicted_cls = sig_clf.predict(test_x_responseCoding[test_point_index].reshape(1,-1))
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_responseCoding[test_point
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_responseCoding[test_point
_index].reshape(1,-1)),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.feature_importances_)
print("-"*50)
for i in indices:
    if i<9:
        print("Gene is important feature")
    elif i<18:
        print("Variation is important feature")
    else:
        print("Text is important feature")
```

```
Predicted Class : 6
Predicted Class Probabilities: [[0.0056 0.0038 0.0065 0.0075 0.1295 0.8358 0.0021 0.0045 0.0046]]
Actual Class : 6
--------------------------------------------------
Variation is important feature
Variation is important feature
Variation is important feature
Variation is important feature
Gene is important feature
Variation is important feature
Variation is important feature
Text is important feature
Text is important feature
Gene is important feature
Text is important feature
Text is important feature
Text is important feature
Variation is important feature
Gene is important feature
Gene is important feature
Text is important feature
Gene is important feature
Gene is important feature
Variation is important feature
Text is important feature
Text is important feature
Text is important feature
Variation is important feature
Gene is important feature
Gene is important feature
Gene is important feature
```

4.5.5.2. Incorrectly Classified point

In [144]:

```
test_point_index = 2
predicted_cls = sig_clf.predict(test_x_responseCoding[test_point_index].reshape(1,-1))
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_responseCoding[test_point
_index].reshape(1,-1)),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.feature_importances_)
print("-"*50)
for i in indices:
    if i<9:
        print("Gene is important feature")
    elif i<18:
        print("Variation is important feature")
    else:
        print("Text is important feature")
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.0337 0.1545 0.2188 0.0868 0.036  0.0534 0.3519 0.0414 0.0234]]
Actual Class : 6
--------------------------------------------------
Variation is important feature
Variation is important feature
Variation is important feature
Variation is important feature
Variation is important feature
Gene is important feature
Variation is important feature
Variation is important feature
Text is important feature
Text is important feature
Gene is important feature
Text is important feature
Text is important feature
Text is important feature
Variation is important feature
Gene is important feature
Gene is important feature
Text is important feature
Gene is important feature
Gene is important feature
Variation is important feature
Text is important feature
Text is important feature
Text is important feature
Variation is important feature
Gene is important feature
Gene is important feature
Gene is important feature
```

4.7 Stack the models

4.7.1 testing with hyper parameter tuning

In [146]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_mo
del.SGDClassifier.html
# ------------------------------
# default parameters
# SGDClassifier(loss=’hinge’, penalty=’l2’, alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=N
one, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate=’optimal’, eta0=0.0,
power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

#-------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geometric-intuiti
on-1/
#------------------------------


# read more about support vector machines with linear kernals here http://scikit-learn.org/stable/modul
es/generated/sklearn.svm.SVC.html
# --------------------------------
# default parameters
# SVC(C=1.0, kernel=’rbf’, degree=3, gamma=’auto’, coef0=0.0, shrinking=True, probability=False, tol=0.
001,
# cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape=’ovr’, random_
state=None)

# Some of methods of SVM()
# fit(X, y, [sample_weight]) Fit the SVM model according to the given training data.
# predict(X) Perform classification on samples in X.
# --------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/mathematical-deri
vation-copy-8/
# --------------------------------


# read more about support vector machines with linear kernals here http://scikit-learn.org/stable/modul
es/generated/sklearn.ensemble.RandomForestClassifier.html
# --------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion=’gini’, max_depth=None, min_sample
# 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/
# --------------------------------


clf1 = SGDClassifier(alpha=0.0001, penalty='l2', loss='log', class_weight='balanced', random_state=0)
clf1.fit(train_x_onehotCoding, train_y)
sig_clf1 = CalibratedClassifierCV(clf1, method="sigmoid")

clf2 = SGDClassifier(alpha=0.0001, penalty='l2', loss='hinge', class_weight='balanced', random_state=0)
clf2.fit(train_x_onehotCoding, train_y)
sig_clf2 = CalibratedClassifierCV(clf2, method="sigmoid")


clf3 = MultinomialNB(alpha=1)
clf3.fit(train_x_onehotCoding, train_y)
sig_clf3 = CalibratedClassifierCV(clf3, method="sigmoid")

sig_clf1.fit(train_x_onehotCoding, train_y)
print("Logistic Regression :  Log Loss: %0.2f" % (log_loss(cv_y, sig_clf1.predict_proba(cv_x_onehotCoding))))
sig_clf2.fit(train_x_onehotCoding, train_y)
print("Support vector machines : Log Loss: %0.2f" % (log_loss(cv_y, sig_clf2.predict_proba(cv_x_onehotCoding))))
sig_clf3.fit(train_x_onehotCoding, train_y)
print("Naive Bayes : Log Loss: %0.2f" % (log_loss(cv_y, sig_clf3.predict_proba(cv_x_onehotCoding))))
print("-"*50)
alpha = [0.0001,0.001,0.01,0.1,1,10]
best_alpha = 999
for i in alpha:
    lr = LogisticRegression(C=i)
    sclf = StackingClassifier(classifiers=[sig_clf1, sig_clf2, sig_clf3], meta_classifier=lr, use_probas=True)
    sclf.fit(train_x_onehotCoding, train_y)
    print("Stacking Classifer : for the value of alpha: %f Log Loss: %0.3f" % (i, log_loss(cv_y, sclf.predict_proba(cv_x_onehotCoding))))
    log_error =log_loss(cv_y, sclf.predict_proba(cv_x_onehotCoding))
    if best_alpha > log_error:
        best_alpha = log_error
```

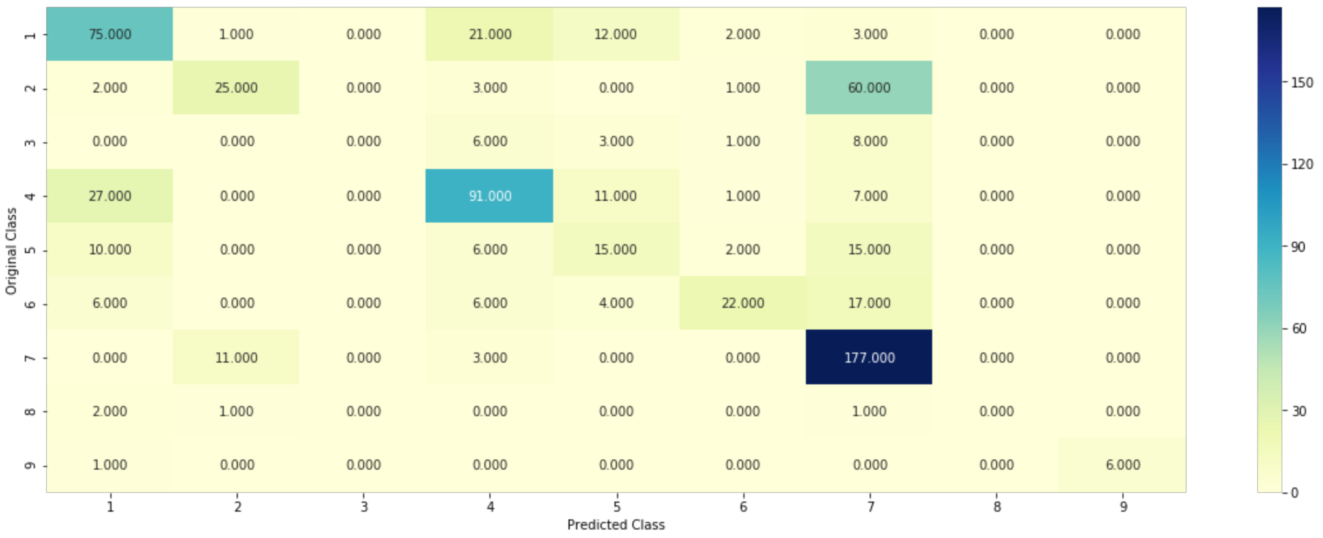
```
Logistic Regression :  Log Loss: 1.07
Support vector machines : Log Loss: 1.14
Naive Bayes : Log Loss: 1.19
--------------------------------------------------
Stacking Classifer : for the value of alpha: 0.000100 Log Loss: 2.174
Stacking Classifer : for the value of alpha: 0.001000 Log Loss: 2.004
Stacking Classifer : for the value of alpha: 0.010000 Log Loss: 1.451
Stacking Classifer : for the value of alpha: 0.100000 Log Loss: 1.140
Stacking Classifer : for the value of alpha: 1.000000 Log Loss: 1.292
Stacking Classifer : for the value of alpha: 10.000000 Log Loss: 1.625
```

4.7.2 testing the model with the best hyper parameters

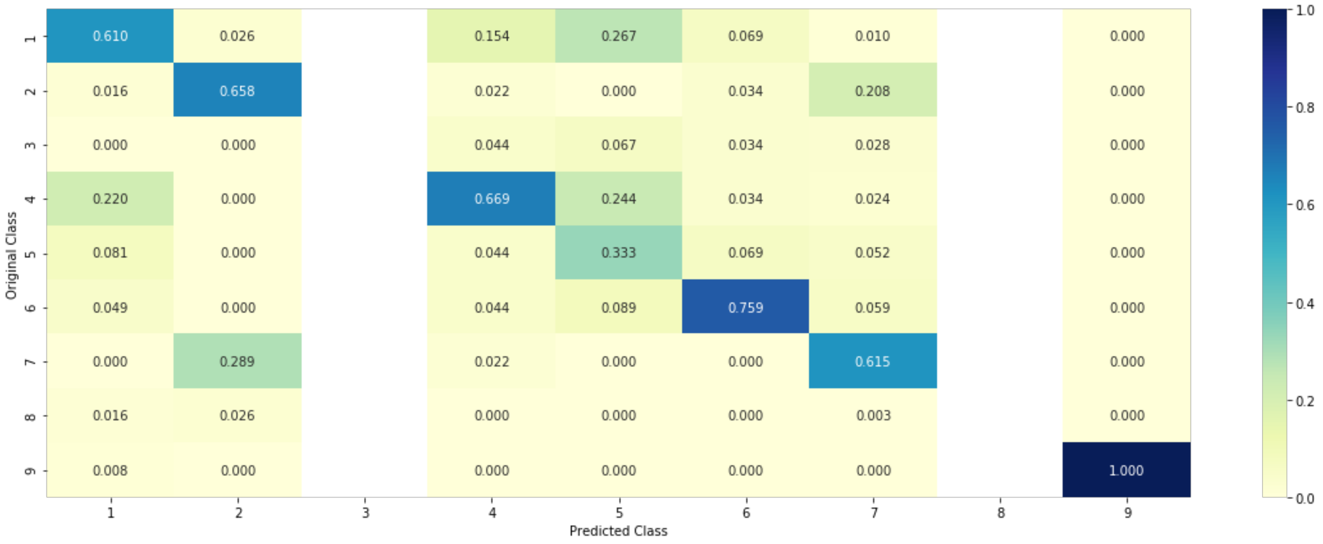
In [147]:

```
lr = LogisticRegression(C=0.1)
lr = LogisticRegression(C=0.1)
sclf = StackingClassifier(classifiers=[sig_clf1, sig_clf2, sig_clf3], meta_classifier=lr, use_probas=True)
sclf.fit(train_x_onehotCoding, train_y)

log_error = log_loss(train_y, sclf.predict_proba(train_x_onehotCoding))
print("Log loss (train) on the stacking classifier :",log_error)

log_error = log_loss(cv_y, sclf.predict_proba(cv_x_onehotCoding))
print("Log loss (CV) on the stacking classifier :",log_error)

log_error = log_loss(test_y, sclf.predict_proba(test_x_onehotCoding))
print("Log loss (test) on the stacking classifier :",log_error)

print("Number of missclassified point :", np.count_nonzero((sclf.predict(test_x_onehotCoding)- test_y))/test_y.shape[0])
plot_confusion_matrix(test_y=test_y, predict_y=sclf.predict(test_x_onehotCoding))
```

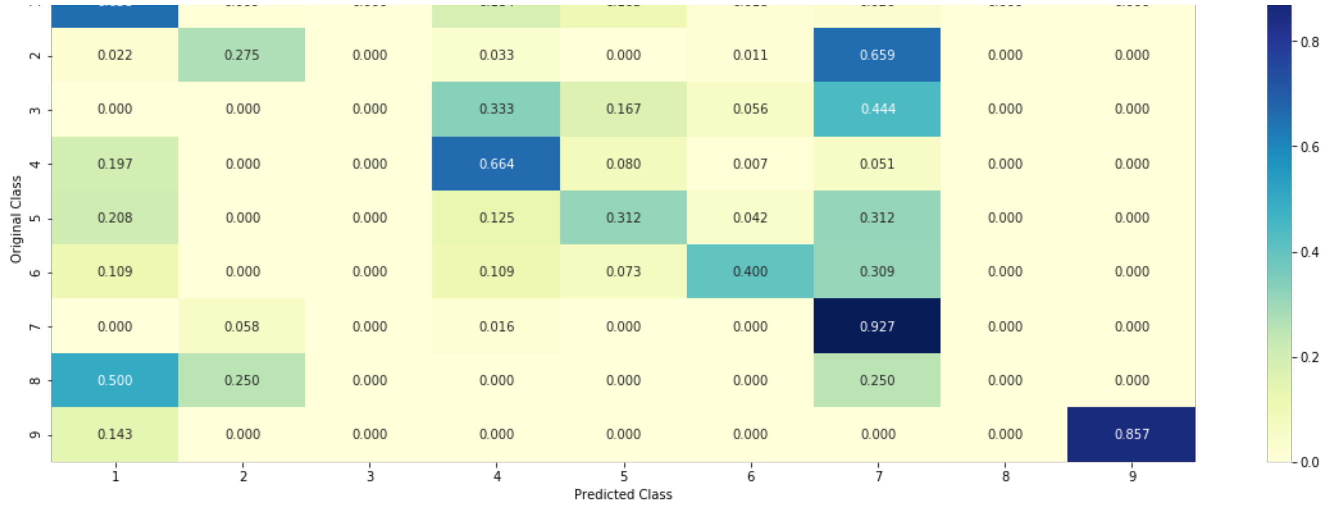
```
Log loss (train) on the stacking classifier : 0.4105101200455324
Log loss (CV) on the stacking classifier : 1.1399724185689657
Log loss (test) on the stacking classifier : 1.1544389208227244
Number of missclassified point : 0.3819548872180451
-------------------- Confusion matrix --------------------
```



```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



```
-------------------- Recall matrix (Row sum=1) --------------------
```



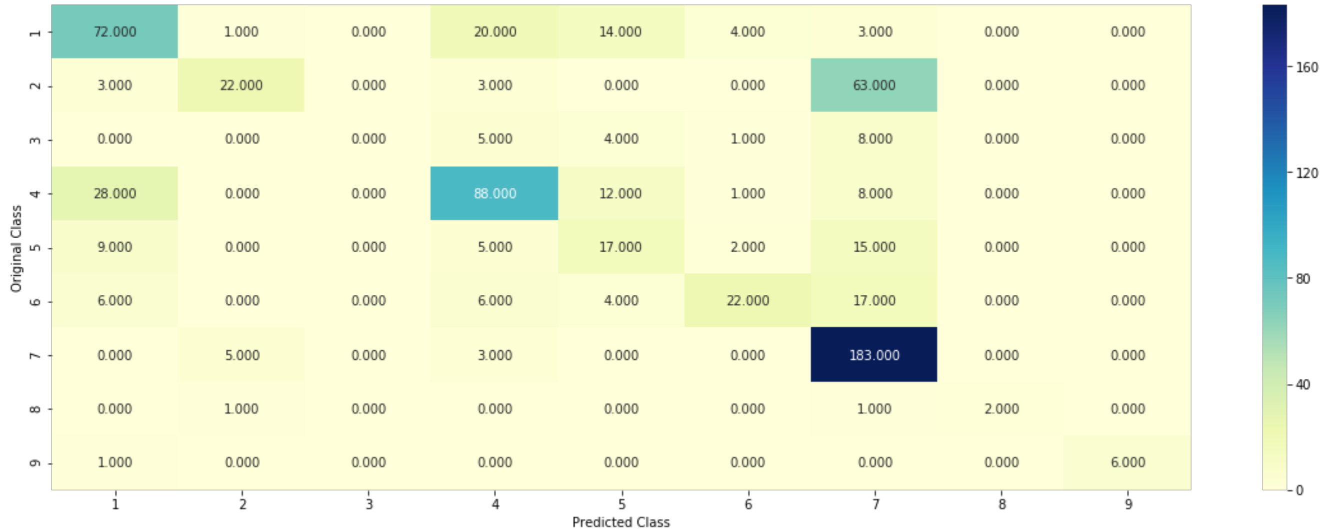
	1	2	3	4	5	6	7	8	9
2	0.022	0.275	0.000	0.033	0.000	0.011	0.659	0.000	0.000
3	0.000	0.000	0.000	0.333	0.167	0.056	0.444	0.000	0.000
4	0.197	0.000	0.000	0.664	0.080	0.007	0.051	0.000	0.000
5	0.208	0.000	0.000	0.125	0.312	0.042	0.312	0.000	0.000
6	0.109	0.000	0.000	0.109	0.073	0.400	0.309	0.000	0.000
7	0.000	0.058	0.000	0.016	0.000	0.000	0.927	0.000	0.000
8	0.500	0.250	0.000	0.000	0.000	0.000	0.250	0.000	0.000
9	0.143	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.857

4.7.3 Maximum Voting classifier

In [148]:

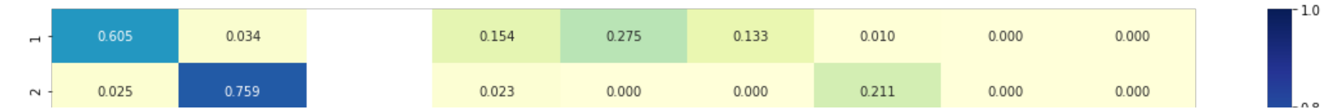
```
#Refer:http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html
from sklearn.ensemble import VotingClassifier
vclf = VotingClassifier(estimators=[('lr', sig_clf1), ('svc', sig_clf2), ('rf', sig_clf3)], voting='sof
t')
vclf.fit(train_x_onehotCoding, train_y)
print("Log loss (train) on the VotingClassifier :", log_loss(train_y, vclf.predict_proba(train_x_onehot
Coding)))
print("Log loss (CV) on the VotingClassifier :", log_loss(cv_y, vclf.predict_proba(cv_x_onehotCoding)))
print("Log loss (test) on the VotingClassifier :", log_loss(test_y, vclf.predict_proba(test_x_onehotCod
ing)))
print("Number of missclassified point :", np.count_nonzero((vclf.predict(test_x_onehotCoding)- test_y))
/test_y.shape[0])
plot_confusion_matrix(test_y=test_y, predict_y=vclf.predict(test_x_onehotCoding))
```

```
Log loss (train) on the VotingClassifier : 0.5567225904996682
Log loss (CV) on the VotingClassifier : 1.1043246257097739
Log loss (test) on the VotingClassifier : 1.0977705999342517
Number of missclassified point : 0.3804511278195489
-------------------- Confusion matrix --------------------
```

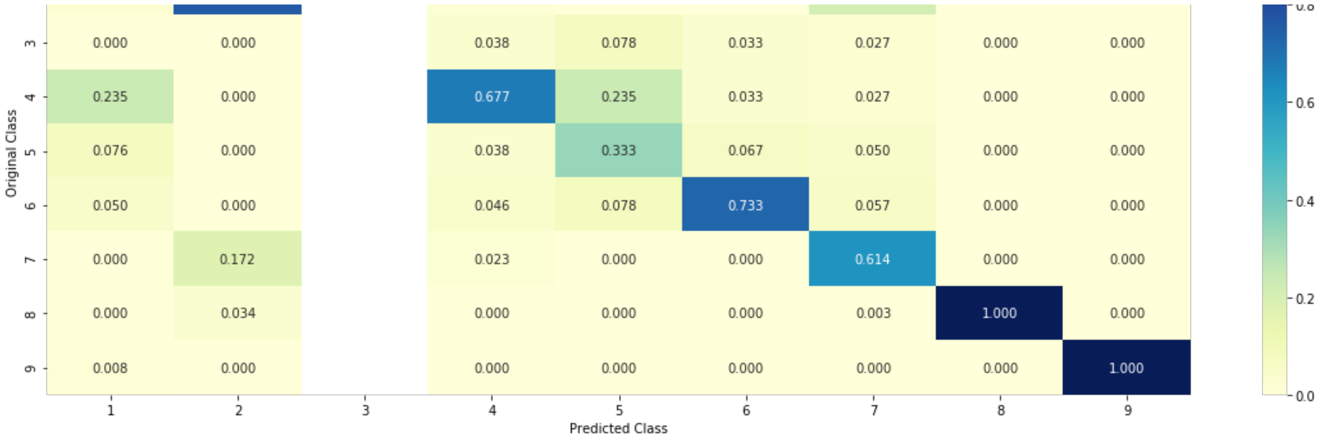


	1	2	3	4	5	6	7	8	9
1	72.000	1.000	0.000	20.000	14.000	4.000	3.000	0.000	0.000
2	3.000	22.000	0.000	3.000	0.000	0.000	63.000	0.000	0.000
3	0.000	0.000	0.000	5.000	4.000	1.000	8.000	0.000	0.000
4	28.000	0.000	0.000	88.000	12.000	1.000	8.000	0.000	0.000
5	9.000	0.000	0.000	5.000	17.000	2.000	15.000	0.000	0.000
6	6.000	0.000	0.000	6.000	4.000	22.000	17.000	0.000	0.000
7	0.000	5.000	0.000	3.000	0.000	0.000	183.000	0.000	0.000
8	0.000	1.000	0.000	0.000	0.000	0.000	1.000	2.000	0.000
9	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	6.000

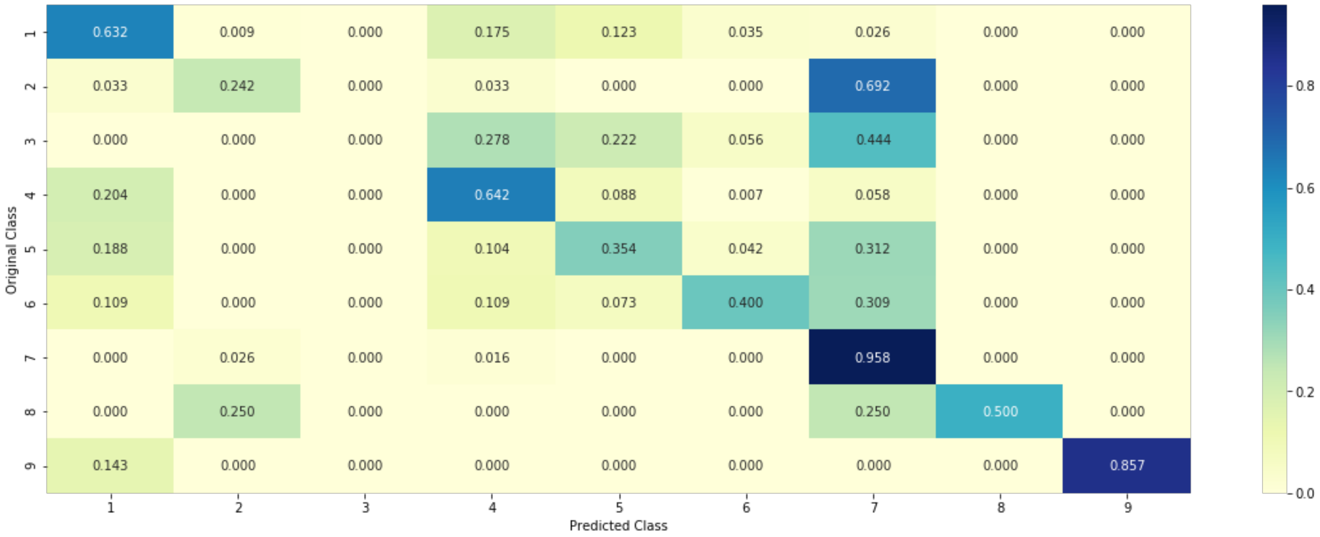
```
-------------------- Precision matrix (Columm Sum=1) --------------------
```



	1	2	3	4	5	6	7	8	9
1	0.605	0.034		0.154	0.275	0.133	0.010	0.000	0.000
2	0.025	0.759		0.023	0.000	0.000	0.211	0.000	0.000



```
-------------------- Recall matrix (Row sum=1) --------------------
```



Conclusion

1. EDA
   - Read training data 'Gene and Variation'
   - Read training data 'Text'
   - Preprocessing 'Text' data
   - Handling missing 'Text' data and merge all features
2. Split data into 3 parts in ratio 64:20:16
3. See the distribution of each class in train, cv and test data
4. Train the Random Model (worst/dump) to find the upperbound logloss
5. Univariant analysis on Gene and Variant
   - What type of feature is?
   - How many categories are present
   - How they are distributed
   - How to featurize the feature: responsecoding with Laplace smoothing and onehotencoding (CountVectorizer)
   - How good is this feature in prediction y?
6. Univariant Analysis on Text data
   - Extract the number of words from each class
   - Featurized the text with onhotencoding(TdidfVectorizer) and select top 1000 features and responsecoding with Laplace smoothing
   - How good is this feature in predicting y?
7. Stacked all features
8. Finally , train model and plot and observe confusion matrix, precision and recall
   - Naive Bayes (NB) and get feature importance
   - Logistic Regr (LR) with class balance and get feature importance
   - Logistic Regr (LR) without class balance and get feature importance
   - Linear SVM and get feature importance
   - Random Forest (RF) and get feature importance

- Random Forest (RF) and get feature importance
- StackModel (Logistic Regr, Linear SVM, Naive Bayes)+Logistic Regr(Final layer)
- Maximum Voting Classifier

In [2]:

```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Feature", "Model", "Train logloss", "CV logloss", "Test logloss", "# of Misclassified
pts"]

print('OHE -> OneHotEncoding\nRC-> ResponseCoding with Laplace smoothing')

feature = ['OHE','RC','OHE','OHE','OHE','OHE','RC','OHE','OHE']
model_ = ['NB','KNN','LR (class balance)','LR (no class balance)','Linear SVM','RF','RF','StackModel+LR
','Maximum Voting Classifier']
train_loss = [0.84,0.61,0.51,0.48,0.48,0.77,0.05,0.41,0.55]
cv_loss = [1.19,1.11,1.07,1.07,1.12,1.19,1.41,1.14,1.1]
test_loss = [1.17,1.03,1.06,1.05,1.11,1.18,1.39,1.15,1.1]
mis_class = [0.389,0.389,0.37,0.353,0.368,0.413,0.498,0.382,0.38]

for i in range(0,9):
    x.add_row([feature[i],model_[i],train_loss[i],cv_loss[i],test_loss[i],mis_class[i]])

print(x)

print('\n\n1. From the observation above, LR(class balance) are perform better than others.')
print('2. LR(no class balance) tends to have more overfitting than LR(class balance)')
```

```
OHE -> OneHotEncoding
RC-> ResponseCoding with Laplace smoothing
+---------+---------------------------+---------------+------------+--------------+--------------------
----+
| Feature |           Model           | Train logloss | CV logloss | Test logloss | # of Misclassified
pts |
+---------+---------------------------+---------------+------------+--------------+--------------------
----+
|   OHE   |             NB            |      0.84     |    1.19    |     1.17     |         0.389
|
|    RC   |            KNN            |      0.61     |    1.11    |     1.03     |         0.389
|
|   OHE   |     LR (class balance)    |      0.51     |    1.07    |     1.06     |          0.37
|
|   OHE   |   LR (no class balance)   |      0.48     |    1.07    |     1.05     |         0.353
|
|   OHE   |         Linear SVM        |      0.48     |    1.12    |     1.11     |         0.368
|
|   OHE   |             RF            |      0.77     |    1.19    |     1.18     |         0.413
|
|    RC   |             RF            |      0.05     |    1.41    |     1.39     |         0.498
|
|   OHE   |       StackModel+LR       |      0.41     |    1.14    |     1.15     |         0.382
|
|   OHE   | Maximum Voting Classifier |      0.55     |    1.1     |     1.1      |          0.38
|
+---------+---------------------------+---------------+------------+--------------+--------------------
----+


1. From the observation above, LR(class balance) are perform better than others.
2. LR(no class balance) tends to have more overfitting than LR(class balance)
```

In [ ]: