Personalized cancer diagnosis

1. Business Problem

1.1. Description

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

- List item
- List item

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 [3]:

```
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 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.model_selection import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
```

```
from sklearn.ensemble import RandomForestClassifier
warnings.filterwarnings("ignore")
import six
import sys
sys.modules['sklearn.externals.six'] = six
from mlxtend.classifier import StackingClassifier
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
```

In [7]:
```
# !gdown --id 1RmX5_q6D7rzoXD7nPUM_s8rKEf1KVMDi #training_text.zip download
# !gdown --id 1bSQrw5WmDqqI8hBcr8Pflzatx4xCT0Ex #training_variants.zip download
```

In [8]:
```
# !unzip training_text.zip
```

In [9]:
```
# !unzip training_variants.zip
```

3.1. Reading Data

3.1.1. Reading Gene and Variation Data

In [10]:
```
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[10]:

	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 [11]:
```
# note the seprator in this file
data_text =pd.read_csv("training_text",sep="\|\|",engine="python",names=["ID","TEXT"],skip
```

```
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[11]:

	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 [12]:

```
import nltk
nltk.download('stopwords')
```

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

Out[12]:

```
True
```

In [13]:

```
# 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 [15]:

```
#text processing stage.
start_time = time.time()
for index, row in 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.time() - 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 : 37.20944118499756 seconds
```

In [16]:
```
#merging both gene_variations and text data based on ID
result = pd.merge(data, data_text,on='ID', how='left')
result.head()
```

Out[16]:

	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 [17]:
```
result[result.isnull().any(axis=1)]
```

Out[17]:

	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 [18]:
```
result.loc[result['TEXT'].isnull(),'TEXT'] = result['Gene'] +' '+result['Variation']
```

In [19]:
```
result[result['ID']==1109]
```

Out[19]:

	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 [20]:
```
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
X_train, test_df, y_train, y_test = train_test_split(result, y_true, stratify=y_true, test
# split the train data into train and cross validation by maintaining same distribution o
train_df, cv_df, y_train, y_cv = train_test_split(X_train, y_train, stratify=y_train, test
```

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

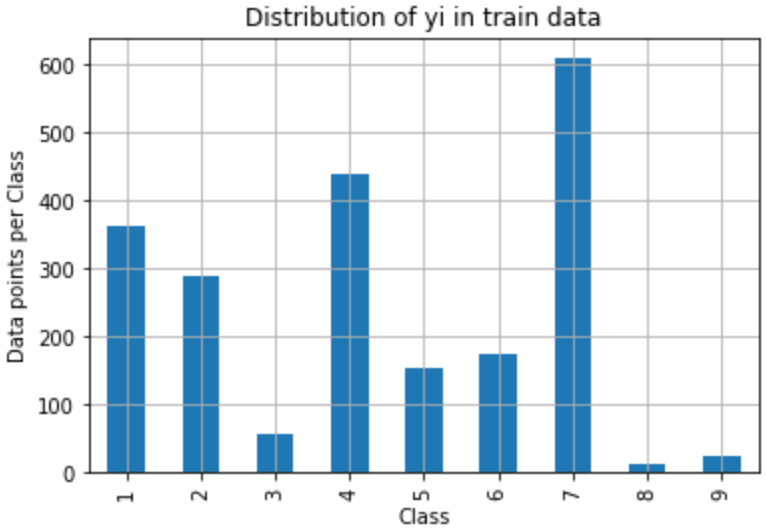
original data set

In [21]:
```
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])
```

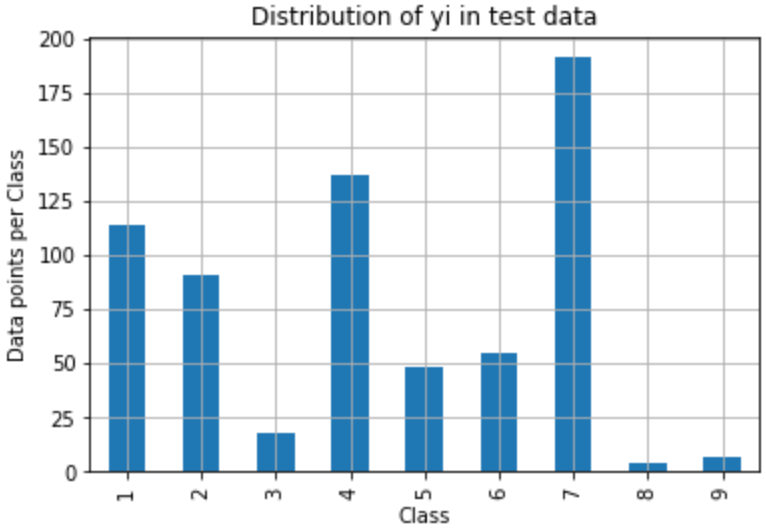
```
Number of data points in train data: 2124
Number of data points in test data: 665
Number of data points in cross validation data: 532
```

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

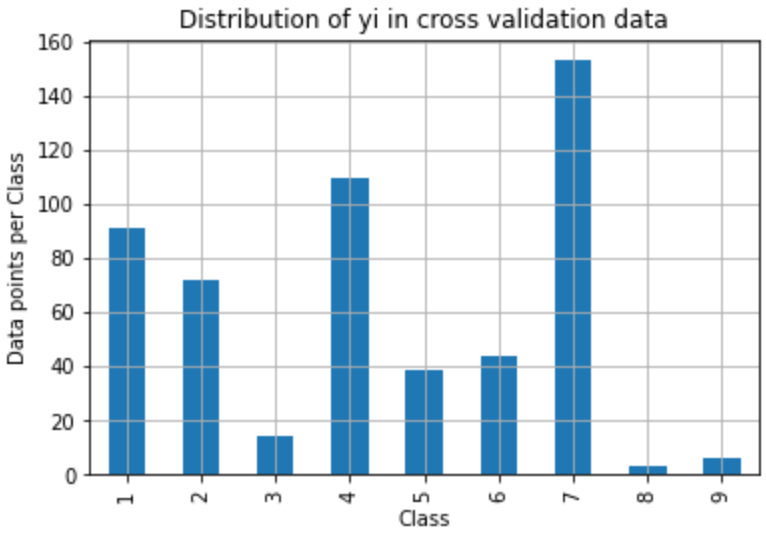
In [22]:
```
# it returns a dict, keys as class labels and values as the number of data points in that
train_class_distribution = train_df['Class'].value_counts().sort_index()
test_class_distribution = test_df['Class'].value_counts().sort_index()
cv_class_distribution = cv_df['Class'].value_counts().sort_index()

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], '


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], '

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], '('
```




```
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 [23]:

```
# 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 predicte

    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
    # 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
    # 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
```

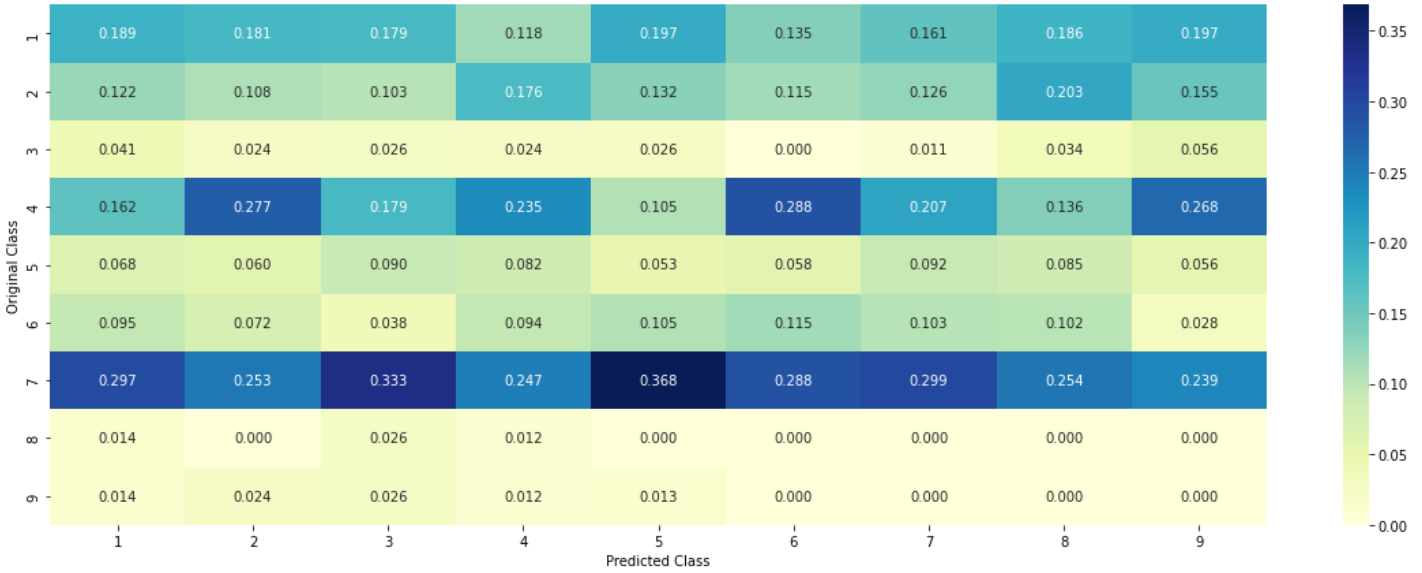
```
    print("-"*20, "Confusion matrix", "-"*20)
    plt.figure(figsize=(20,7))
    sns.heatmap(C, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=l
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    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=l
    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=l
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
```

In [24]:

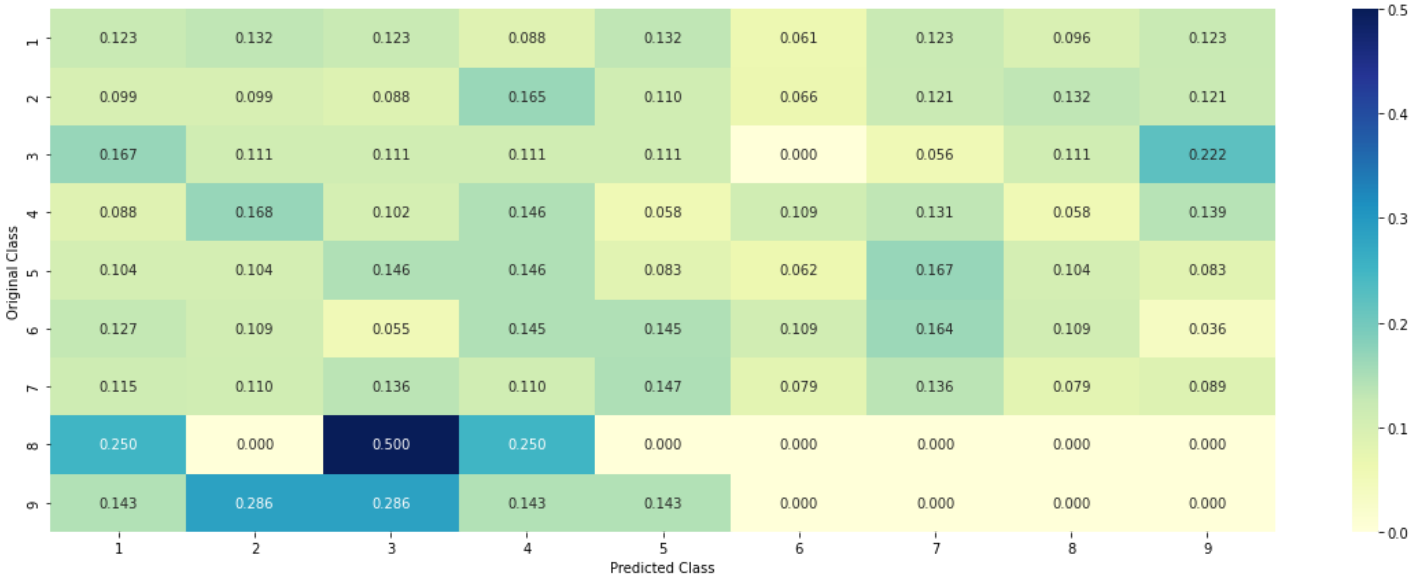
```
# we need to generate 9 numbers and the sum of numbers should be 1
# one solution is to genarate 9 numbers and divide each of the numbers by their sum
# ref: https://stackoverflow.com/a/18662466/4084039
test_data_len = test_df.shape[0]
cv_data_len = cv_df.shape[0]

# 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,


# 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-

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 2.5090819230750654
Log loss on Test Data using Random Model 2.4293904615197515
-------------------- Confusion matrix --------------------
```

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



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



3.3 Univariate Analysis

In [25]:

```
# 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
# build a vector (1*9) , the first element = (number of times it occured in class1 + 10*a
# 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/va
    gv_dict = dict()

    # denominator will contain the number of time that particular feature occured in whole
    for i, denominator in value_count.items():
        # vec will contain (p(yi==1/Gi) probability of gene/variation belongs to perticula
        # 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
            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):
    # print(gv_dict)
    #     {'BRCA1': [0.20075757575757575, 0.03787878787878788, 0.06818181818181817, 0.13
    #      'TP53': [0.32142857142857145, 0.06122448979591836, 0.06122448979591836, 0.27
    #      'EGFR': [0.05681818181818181, 0.21590909090909091, 0.0625, 0.06818181818181
    #      'BRCA2': [0.13333333333333333, 0.060606060606060608, 0.060606060606060608, 0.0
    #      'PTEN': [0.069182389937106917, 0.062893081761006289, 0.069182389937106917, 0.4
    #      'KIT': [0.066225165562913912, 0.25165562913907286, 0.072847682119205295, 0.072
    #      'BRAF': [0.06666666666666666, 0.17999999999999999, 0.073333333333333334, 0.07
    #      ...
    #     }
    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 i
    gv_fea = []
    # for every feature values in the given data frame we will check if it is there in the
    # 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 [26]:

```
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 : 241
BRCA1     168
TP53       99
EGFR       91
PTEN       88
BRCA2      80
KIT        61
BRAF       60
ERBB2      44
```

```
ALK         41
CDKN2A      37
Name: Gene, dtype: int64
```

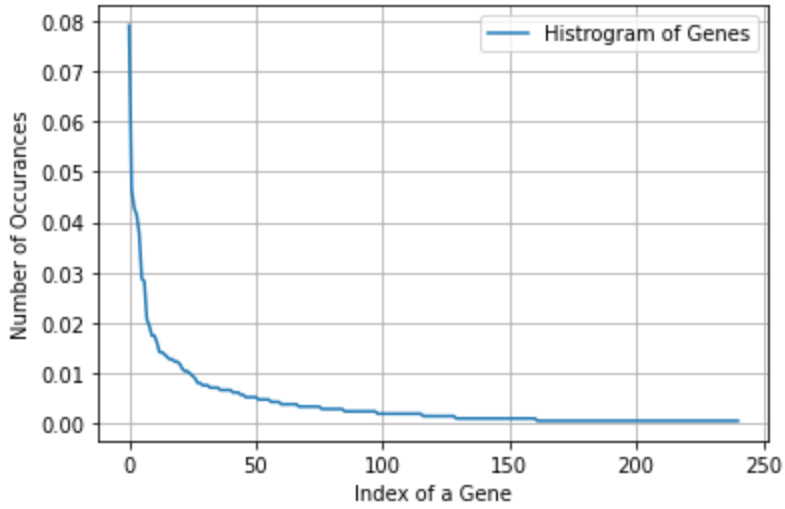
In [27]:

```
print("Ans: There are", unique_genes.shape[0] ,"different categories of genes in the trai
```

```
Ans: There are 241 different categories of genes in the train data, and they are distibute
d as follows
```

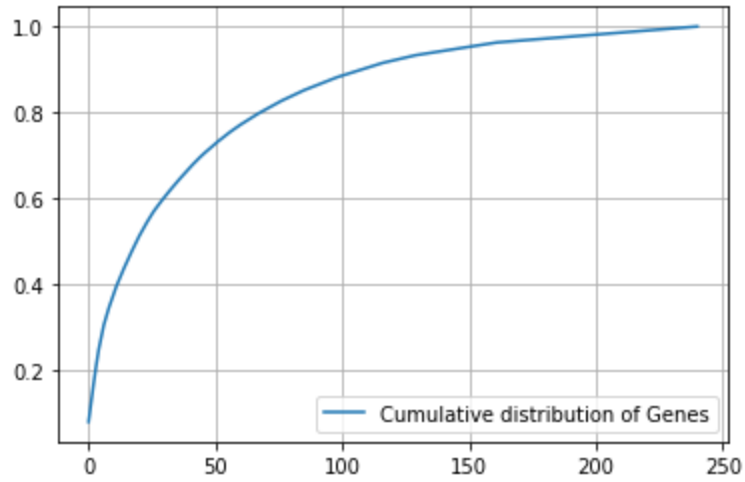
In [28]:

```
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 [29]:

```
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 [30]:
```
#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 [31]:
```
print("train_gene_feature_responseCoding is converted feature using respone coding method.
```

```
train_gene_feature_responseCoding is converted feature using respone coding method. The sh
ape of gene feature: (2124, 9)
```

In [32]:
```
# 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 [33]:
```
train_df['Gene'].head()
```

Out[33]:
```
1190    PIK3CA
782      ERBB3
3130      KRAS
2302      JAK1
969       ESR1
Name: Gene, dtype: object
```

In [34]:
```
gene_vectorizer.get_feature_names()
```

Out[34]:
```
['abl1',
 'acvr1',
 'ago2',
 'akt1',
 'akt2',
 'akt3',
 'alk',
 'apc',
 'ar',
 'araf',
 'arid1a',
 'arid1b',
 'arid2',
 'arid5b',
 'asxl1',
 'asxl2',
 'atm',
```

```
'atr',
'atrx',
'aurka',
'aurkb',
'axin1',
'b2m',
'bap1',
'bard1',
'bcl10',
'bcl2',
'bcor',
'braf',
'brca1',
'brca2',
'brd4',
'brip1',
'btk',
'card11',
'carm1',
'casp8',
'cbl',
'ccnd1',
'ccnd2',
'ccnd3',
'ccne1',
'cdh1',
'cdk12',
'cdk4',
'cdkn1a',
'cdkn1b',
'cdkn2a',
'cdkn2b',
'cdkn2c',
'chek2',
'cic',
'crebbp',
'ctcf',
'ctla4',
'ctnnb1',
'ddr2',
'dicer1',
'dnmt3a',
'dnmt3b',
'dusp4',
'egfr',
'eif1ax',
'elf3',
'ep300',
'epas1',
'epcam',
'erbb2',
'erbb3',
'erbb4',
'ercc2',
'ercc3',
'ercc4',
'erg',
'esr1',
'etv1',
'etv6',
'ewsr1',
'ezh2',
'fam58a',
'fanca',
'fancc',
'fat1',
```

```
'fbxw7',
'fgf19',
'fgf4',
'fgfr1',
'fgfr2',
'fgfr3',
'fgfr4',
'flt1',
'flt3',
'foxa1',
'foxo1',
'foxp1',
'fubp1',
'gata3',
'gli1',
'gna11',
'gnaq',
'gnas',
'h3f3a',
'hist1h1c',
'hla',
'hnf1a',
'hras',
'idh1',
'idh2',
'igf1r',
'ikbke',
'jak1',
'jak2',
'jun',
'kdm5a',
'kdm5c',
'kdr',
'keap1',
'kit',
'klf4',
'kmt2a',
'kmt2c',
'kmt2d',
'knstrn',
'kras',
'map2k1',
'map2k2',
'map2k4',
'map3k1',
'mapk1',
'mdm2',
'med12',
'mef2b',
'men1',
'met',
'mga',
'mlh1',
'mpl',
'msh2',
'msh6',
'mtor',
'myc',
'mycn',
'myd88',
'myod1',
'ncor1',
'nf1',
'nf2',
'nfe2l2',
'nfkbia',
```

```
'nkx2',
'notch1',
'notch2',
'npm1',
'nras',
'nsd1',
'ntrk1',
'ntrk2',
'ntrk3',
'nup93',
'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',
'rad51c',
'rad51d',
'rad54l',
'raf1',
'rara',
'rasa1',
'rb1',
'rbm10',
'ret',
'rheb',
'rhoa',
'rictor',
'rit1',
'ros1',
'rras2',
'runx1',
'rxra',
'rybp',
'sdhb',
'setd2',
'sf3b1',
'smad2',
'smad3',
'smad4',
'smarca4',
'smarcb1',
'smo',
'sos1',
'sox9',
```

```
'spop',
'src',
'srsf2',
'stag2',
'stat3',
'stk11',
'tcf3',
'tcf7l2',
'tert',
'tet1',
'tet2',
'tgfbr1',
'tgfbr2',
'tmprss2',
'tp53',
'tp53bp1',
'tsc1',
'tsc2',
'u2af1',
'vhl',
'whsc1',
'whsc1l1',
'xpo1',
'xrcc2',
'yap1']
```

In [35]:

```
print("train_gene_feature_onehotCoding is converted feature using one-hot encoding method.
```

```
train_gene_feature_onehotCoding is converted feature using one-hot encoding method. The sh
ape of gene feature: (2124, 240)
```

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

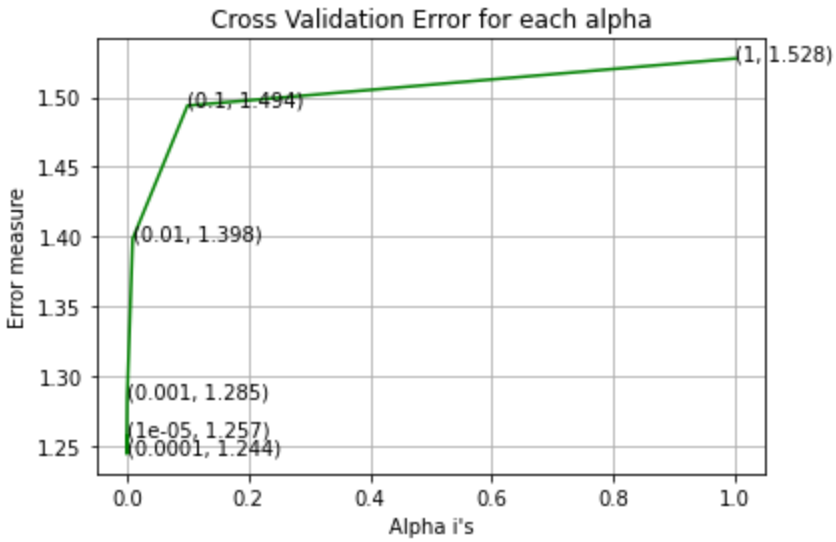
There are many ways to estimate how good a feature is, in predicting y_i. One of the good methods is to build a proper ML model using just this feature. In this case, we will build a logistic regression model using only Gene feature (one hot encoded) to predict y_i.

In [36]:

```
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/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
# 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
# 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

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
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:
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
```

```
For values of alpha =  1e-05 The log loss is: 1.256710975987449
For values of alpha =  0.0001 The log loss is: 1.244102418037804
For values of alpha =  0.001 The log loss is: 1.2845457041850366
For values of alpha =  0.01 The log loss is: 1.398362088999798
For values of alpha =  0.1 The log loss is: 1.493890111890319
For values of alpha =  1 The log loss is: 1.5276978857137333
```




```
For values of best alpha =  0.0001 The train log loss is: 0.9990784061542896
For values of best alpha =  0.0001 The cross validation log loss is: 1.244102418037804
For values of best alpha =  0.0001 The test log loss is: 1.1393041262782093
```

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 [37]:

```
print("Q6. How many data points in Test and CV datasets are covered by the ", unique_genes

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/
print('2. In cross validation data',cv_coverage, 'out of ',cv_df.shape[0],":" ,(cv_coverag
```

```
Q6. How many data points in Test and CV datasets are covered by the  241  genes in train d
ataset?
Ans
1. In test data 653 out of 665 : 98.19548872180451
2. In cross validation data 517 out of  532 : 97.18045112781954
```

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?

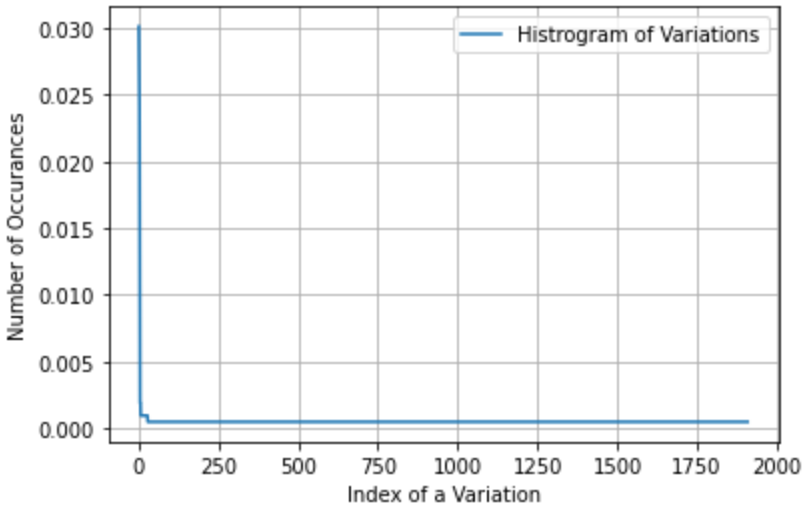
In [38]:
```
unique_variations = train_df['Variation'].value_counts()
print('Number of Unique Variations :', unique_variations.shape[0])
# the top 10 variations that occured most
print(unique_variations.head(10))
```

```
Number of Unique Variations : 1910
Truncating_Mutations    64
Deletion                58
Amplification           51
Fusions                 17
Overexpression           4
G12V                     4
Q61L                     2
S308A                    2
T73I                     2
Q61R                     2
Name: Variation, dtype: int64
```

In [39]:
```
print("Ans: There are", unique_variations.shape[0] ,"different categories of variations in
```

```
Ans: There are 1910 different categories of variations in the train data, and they are dis
tibuted as follows
```

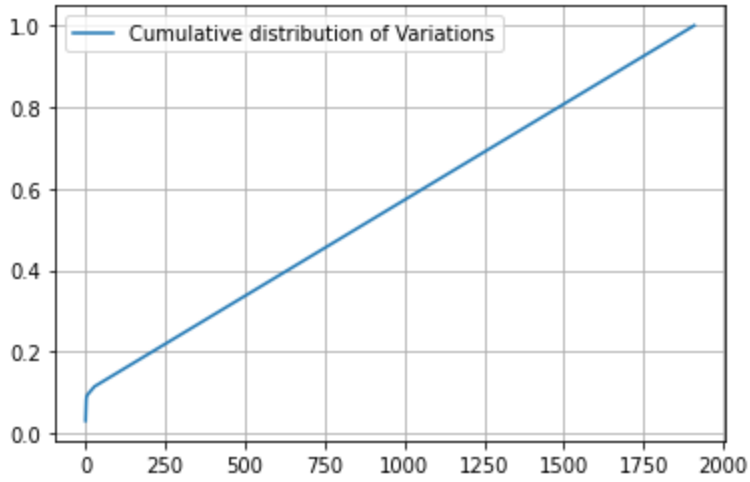
In [40]:
```
s = sum(unique_variations.values);
h = unique_variations.values/s;
plt.plot(h, label="Histrogram of Variations")
plt.xlabel('Index of a Variation')
plt.ylabel('Number of Occurances')
plt.legend()
plt.grid()
plt.show()
```



In [41]:

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

```
[0.03013183 0.05743879 0.08145009 ... 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 [42]:

```
# alpha is used for laplace smoothing
alpha = 1
# train gene feature
train_variation_feature_responseCoding = np.array(get_gv_feature(alpha, "Variation", train
# test gene feature
test_variation_feature_responseCoding = np.array(get_gv_feature(alpha, "Variation", test_
# cross validation gene feature
cv_variation_feature_responseCoding = np.array(get_gv_feature(alpha, "Variation", cv_df))
```

In [43]:
```
print("train_variation_feature_responseCoding is a converted feature using the response co
```

```
train_variation_feature_responseCoding is a converted feature using the response coding me
thod. The shape of Variation feature: (2124, 9)
```

In [44]:
```
# one-hot encoding of variation feature.
variation_vectorizer = CountVectorizer()
train_variation_feature_onehotCoding = variation_vectorizer.fit_transform(train_df['Variat
test_variation_feature_onehotCoding = variation_vectorizer.transform(test_df['Variation'])
cv_variation_feature_onehotCoding = variation_vectorizer.transform(cv_df['Variation'])
```

In [ ]:
```

```

In [45]:
```
print("train_variation_feature_onehotEncoded is converted feature using the onne-hot encoc
```

```
train_variation_feature_onehotEncoded is converted feature using the onne-hot encoding met
hod. The shape of Variation feature: (2124, 1940)
```

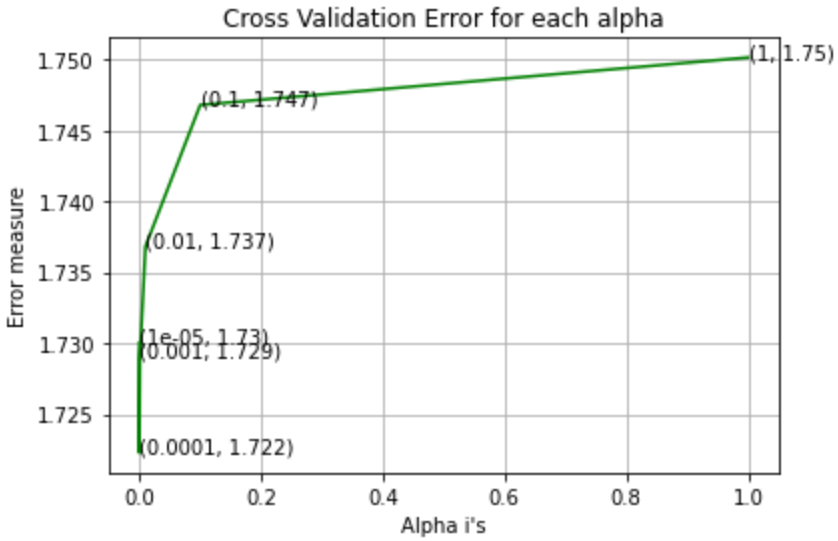
Q10. How good is this Variation feature in predicting y_i?

Let's build a model just like the earlier!

In [46]:
```
alpha = [10 ** x for x in range(-5, 1)]

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
# 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
# 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))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_cv, predict_y, labels

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
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:
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
```

```
For values of alpha =  1e-05 The log loss is: 1.7300295227831646
For values of alpha =  0.0001 The log loss is: 1.7222667390100297
For values of alpha =  0.001 The log loss is: 1.729008449646787
For values of alpha =  0.01 The log loss is: 1.7368181168627341
For values of alpha =  0.1 The log loss is: 1.746797142482326
For values of alpha =  1 The log loss is: 1.750147201831202
```




```
For values of best alpha =  0.0001 The train log loss is: 0.7226732957024683
For values of best alpha =  0.0001 The cross validation log loss is: 1.7222667390100297
For values of best alpha =  0.0001 The test log loss is: 1.7234476452034517
```

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 [47]:

```
print("Q12. How many data points are covered by total ", unique_variations.shape[0], " ge
test_coverage=test_df[test_df['Variation'].isin(list(set(train_df['Variation'])))].shape[
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/
print('2. In cross validation data',cv_coverage, 'out of ',cv_df.shape[0],":" ,(cv_coverag
```

```
Q12. How many data points are covered by total  1910  genes in test and cross validation d
ata sets?
Ans
1. In test data 62 out of 665 : 9.323308270676693
2. In cross validation data 46 out of  532 : 8.646616541353383
```

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 [48]:
```
# 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 [49]:
```
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.get(word,
            text_feature_responseCoding[row_index][i] = math.exp(sum_prob/len(row['TEXT']).
            row_index += 1
    return text_feature_responseCoding
```

In [50]:
```
# building a CountVectorizer with all the words that occured minimum 3 times in train data
text_vectorizer = CountVectorizer(min_df=3)
train_text_feature_onehotCoding = text_vectorizer.fit_transform(train_df['TEXT'])
# getting all the feature names (words)
train_text_features= text_vectorizer.get_feature_names()

# train_text_feature_onehotCoding.sum(axis=0).A1 will sum every row and returns (1*number
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 oc
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 : 53084
```

In [51]:
```
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 = extract_dictionary_paddle(train_df)


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[i]+90))
    confuse_array.append(ratios)
confuse_array = np.array(confuse_array)
```

In [52]:

```
#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 [53]:

```
# 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_featur
test_text_feature_responseCoding = (test_text_feature_responseCoding.T/test_text_feature_r
cv_text_feature_responseCoding = (cv_text_feature_responseCoding.T/cv_text_feature_respons
```

In [54]:

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

In [55]:

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

In [56]:

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

```
Counter({3: 5409, 4: 3457, 6: 2638, 7: 2616, 5: 2598, 8: 2167, 9: 1736, 10: 1319, 12: 114
0, 11: 1126, 14: 1045, 16: 1008, 18: 852, 15: 840, 13: 818, 17: 632, 20: 532, 24: 511, 21:
492, 19: 487, 22: 465, 28: 431, 25: 386, 42: 351, 30: 337, 23: 335, 32: 334, 26: 323, 27:
320, 48: 315, 36: 305, 34: 284, 33: 277, 35: 256, 29: 255, 31: 244, 40: 229, 39: 208, 37:
199, 38: 197, 41: 194, 50: 190, 46: 190, 54: 176, 47: 174, 43: 161, 44: 159, 51: 157, 49:
156, 45: 152, 56: 151, 52: 149, 66: 147, 57: 142, 53: 131, 55: 130, 64: 118, 61: 116, 58:
111, 59: 108, 72: 107, 63: 107, 70: 105, 68: 104, 60: 104, 65: 101, 62: 101, 67: 100, 96:
89, 71: 88, 73: 86, 69: 86, 75: 85, 90: 80, 77: 79, 84: 77, 74: 77, 81: 75, 92: 74, 83: 7
4, 93: 72, 76: 72, 80: 69, 85: 67, 91: 66, 98: 63, 86: 63, 78: 61, 108: 60, 94: 60, 89: 6
0, 87: 60, 79: 60, 101: 59, 99: 59, 88: 59, 82: 59, 104: 58, 95: 55, 103: 54, 106: 53, 11
1: 52, 97: 52, 116: 51, 105: 49, 100: 49, 136: 48, 112: 48, 110: 47, 144: 46, 107: 46, 12
1: 45, 140: 44, 138: 43, 128: 43, 124: 43, 115: 43, 113: 43, 102: 43, 118: 42, 132: 41, 12
6: 41, 120: 40, 122: 39, 157: 38, 135: 38, 125: 38, 119: 38, 141: 36, 139: 36, 133: 36, 12
7: 36, 114: 36, 143: 35, 130: 35, 123: 34, 148: 33, 146: 33, 137: 33, 131: 33, 129: 33, 16
9: 32, 162: 32, 161: 31, 159: 31, 198: 30, 173: 30, 196: 29, 177: 29, 152: 29, 109: 29, 18
```

0: 28, 166: 28, 154: 28, 151: 28, 149: 28, 134: 28, 248: 27, 210: 27, 187: 27, 168: 27, 16
4: 27, 160: 27, 191: 26, 142: 26, 240: 25, 199: 25, 155: 25, 153: 25, 147: 25, 145: 25, 11
7: 25, 229: 24, 201: 24, 192: 24, 181: 24, 167: 24, 273: 23, 202: 23, 200: 23, 186: 23, 17
9: 23, 171: 23, 156: 23, 226: 22, 195: 22, 244: 21, 222: 21, 204: 21, 203: 21, 189: 21, 18
3: 21, 182: 21, 178: 21, 163: 21, 231: 20, 223: 20, 221: 20, 211: 20, 188: 20, 176: 20, 17
5: 20, 165: 20, 150: 20, 276: 19, 220: 19, 217: 19, 209: 19, 206: 19, 205: 19, 190: 19, 17
0: 19, 158: 19, 271: 18, 266: 18, 256: 18, 252: 18, 235: 18, 225: 18, 193: 18, 322: 17, 29
3: 17, 277: 17, 264: 17, 241: 17, 230: 17, 228: 17, 184: 17, 174: 17, 329: 16, 302: 16, 28
8: 16, 275: 16, 251: 16, 243: 16, 236: 16, 234: 16, 212: 16, 207: 16, 185: 16, 172: 16, 39
7: 15, 384: 15, 336: 15, 321: 15, 298: 15, 294: 15, 282: 15, 258: 15, 257: 15, 224: 15, 21
8: 15, 197: 15, 352: 14, 325: 14, 323: 14, 309: 14, 297: 14, 296: 14, 267: 14, 250: 14, 24
5: 14, 238: 14, 237: 14, 215: 14, 214: 14, 208: 14, 438: 13, 399: 13, 386: 13, 356: 13, 33
2: 13, 307: 13, 292: 13, 289: 13, 254: 13, 249: 13, 246: 13, 239: 13, 233: 13, 219: 13, 45
0: 12, 340: 12, 326: 12, 280: 12, 269: 12, 260: 12, 247: 12, 242: 12, 227: 12, 213: 12, 51
3: 11, 434: 11, 369: 11, 368: 11, 348: 11, 328: 11, 318: 11, 316: 11, 310: 11, 272: 11, 26
3: 11, 262: 11, 255: 11, 232: 11, 449: 10, 414: 10, 406: 10, 395: 10, 338: 10, 333: 10, 32
4: 10, 313: 10, 312: 10, 305: 10, 274: 10, 259: 10, 194: 10, 680: 9, 486: 9, 458: 9, 444:
9, 440: 9, 436: 9, 419: 9, 394: 9, 380: 9, 376: 9, 367: 9, 359: 9, 351: 9, 346: 9, 345: 9,
343: 9, 341: 9, 335: 9, 334: 9, 330: 9, 320: 9, 315: 9, 306: 9, 304: 9, 303: 9, 299: 9, 29
5: 9, 285: 9, 281: 9, 279: 9, 270: 9, 265: 9, 733: 8, 596: 8, 576: 8, 530: 8, 515: 8, 507:
8, 492: 8, 480: 8, 463: 8, 435: 8, 421: 8, 417: 8, 412: 8, 408: 8, 389: 8, 382: 8, 378: 8,
366: 8, 363: 8, 360: 8, 358: 8, 349: 8, 347: 8, 314: 8, 308: 8, 291: 8, 290: 8, 286: 8, 27
8: 8, 261: 8, 216: 8, 991: 7, 985: 7, 774: 7, 711: 7, 643: 7, 611: 7, 599: 7, 589: 7, 562:
7, 554: 7, 539: 7, 533: 7, 490: 7, 472: 7, 441: 7, 433: 7, 430: 7, 428: 7, 423: 7, 415: 7,
411: 7, 409: 7, 401: 7, 392: 7, 374: 7, 372: 7, 365: 7, 350: 7, 342: 7, 337: 7, 331: 7, 31
7: 7, 301: 7, 287: 7, 283: 7, 268: 7, 1165: 6, 853: 6, 846: 6, 809: 6, 771: 6, 724: 6, 68
1: 6, 674: 6, 665: 6, 664: 6, 656: 6, 639: 6, 637: 6, 594: 6, 579: 6, 578: 6, 563: 6, 557:
6, 549: 6, 545: 6, 538: 6, 527: 6, 526: 6, 501: 6, 499: 6, 495: 6, 475: 6, 469: 6, 467: 6,
431: 6, 427: 6, 426: 6, 407: 6, 398: 6, 393: 6, 379: 6, 375: 6, 371: 6, 364: 6, 344: 6, 33
9: 6, 253: 6, 1539: 5, 1405: 5, 1331: 5, 1327: 5, 1268: 5, 1044: 5, 897: 5, 890: 5, 877:
5, 820: 5, 797: 5, 787: 5, 783: 5, 780: 5, 773: 5, 766: 5, 764: 5, 754: 5, 744: 5, 730: 5,
703: 5, 702: 5, 675: 5, 671: 5, 655: 5, 653: 5, 649: 5, 640: 5, 634: 5, 632: 5, 626: 5, 61
9: 5, 606: 5, 595: 5, 573: 5, 568: 5, 565: 5, 564: 5, 537: 5, 536: 5, 535: 5, 528: 5, 520:
5, 517: 5, 511: 5, 503: 5, 498: 5, 493: 5, 489: 5, 488: 5, 487: 5, 482: 5, 481: 5, 473: 5,
468: 5, 456: 5, 455: 5, 453: 5, 451: 5, 448: 5, 446: 5, 429: 5, 425: 5, 420: 5, 416: 5, 40
5: 5, 403: 5, 390: 5, 385: 5, 377: 5, 370: 5, 362: 5, 357: 5, 353: 5, 300: 5, 2439: 4, 217
6: 4, 1923: 4, 1869: 4, 1851: 4, 1785: 4, 1454: 4, 1448: 4, 1313: 4, 1296: 4, 1267: 4, 119
8: 4, 1187: 4, 1184: 4, 1169: 4, 1077: 4, 1014: 4, 973: 4, 972: 4, 930: 4, 929: 4, 919: 4,
894: 4, 869: 4, 858: 4, 839: 4, 836: 4, 812: 4, 799: 4, 798: 4, 792: 4, 789: 4, 786: 4, 78
1: 4, 763: 4, 761: 4, 758: 4, 757: 4, 743: 4, 736: 4, 735: 4, 734: 4, 731: 4, 728: 4, 709:
4, 705: 4, 690: 4, 684: 4, 682: 4, 654: 4, 641: 4, 628: 4, 625: 4, 624: 4, 622: 4, 616: 4,
610: 4, 591: 4, 590: 4, 585: 4, 581: 4, 580: 4, 575: 4, 574: 4, 570: 4, 553: 4, 552: 4, 54
4: 4, 540: 4, 531: 4, 524: 4, 512: 4, 510: 4, 508: 4, 506: 4, 505: 4, 504: 4, 500: 4, 494:
4, 491: 4, 485: 4, 484: 4, 479: 4, 477: 4, 476: 4, 470: 4, 462: 4, 461: 4, 460: 4, 452: 4,
443: 4, 437: 4, 418: 4, 410: 4, 404: 4, 402: 4, 388: 4, 387: 4, 381: 4, 373: 4, 361: 4, 35
5: 4, 354: 4, 319: 4, 284: 4, 3008: 3, 2487: 3, 2444: 3, 2089: 3, 1919: 3, 1838: 3, 1816:
3, 1788: 3, 1781: 3, 1761: 3, 1705: 3, 1699: 3, 1682: 3, 1653: 3, 1647: 3, 1561: 3, 1558:
3, 1549: 3, 1530: 3, 1519: 3, 1514: 3, 1506: 3, 1498: 3, 1473: 3, 1453: 3, 1450: 3, 1440:
3, 1430: 3, 1422: 3, 1416: 3, 1350: 3, 1338: 3, 1309: 3, 1278: 3, 1259: 3, 1255: 3, 1249:
3, 1244: 3, 1243: 3, 1219: 3, 1213: 3, 1204: 3, 1203: 3, 1195: 3, 1190: 3, 1185: 3, 1182:
3, 1155: 3, 1154: 3, 1151: 3, 1150: 3, 1132: 3, 1118: 3, 1108: 3, 1104: 3, 1095: 3, 1094:
3, 1091: 3, 1088: 3, 1086: 3, 1084: 3, 1064: 3, 1061: 3, 1060: 3, 1056: 3, 1050: 3, 1048:
3, 1013: 3, 1009: 3, 999: 3, 993: 3, 990: 3, 987: 3, 969: 3, 967: 3, 961: 3, 950: 3, 949:
3, 946: 3, 944: 3, 941: 3, 939: 3, 938: 3, 935: 3, 932: 3, 926: 3, 923: 3, 922: 3, 915: 3,
914: 3, 912: 3, 905: 3, 900: 3, 899: 3, 892: 3, 884: 3, 870: 3, 866: 3, 864: 3, 862: 3, 85
7: 3, 849: 3, 845: 3, 841: 3, 838: 3, 828: 3, 822: 3, 817: 3, 813: 3, 796: 3, 782: 3, 778:
3, 776: 3, 772: 3, 769: 3, 759: 3, 756: 3, 752: 3, 748: 3, 747: 3, 746: 3, 732: 3, 729: 3,
725: 3, 722: 3, 719: 3, 718: 3, 713: 3, 708: 3, 707: 3, 699: 3, 698: 3, 693: 3, 692: 3, 69
1: 3, 683: 3, 679: 3, 678: 3, 673: 3, 672: 3, 667: 3, 659: 3, 657: 3, 651: 3, 648: 3, 647:
3, 638: 3, 636: 3, 630: 3, 615: 3, 612: 3, 608: 3, 607: 3, 604: 3, 603: 3, 602: 3, 601: 3,
600: 3, 597: 3, 588: 3, 586: 3, 583: 3, 577: 3, 572: 3, 569: 3, 560: 3, 559: 3, 558: 3, 55
6: 3, 555: 3, 550: 3, 547: 3, 543: 3, 542: 3, 534: 3, 532: 3, 521: 3, 519: 3, 518: 3, 516:
3, 509: 3, 496: 3, 471: 3, 466: 3, 459: 3, 457: 3, 454: 3, 447: 3, 442: 3, 439: 3, 432: 3,
422: 3, 413: 3, 400: 3, 396: 3, 383: 3, 327: 3, 24560: 2, 12262: 2, 7235: 2, 7075: 2, 698
0: 2, 6241: 2, 5195: 2, 5080: 2, 4812: 2, 4768: 2, 4513: 2, 4496: 2, 4452: 2, 4409: 2, 413
4: 2, 4087: 2, 4009: 2, 3930: 2, 3885: 2, 3866: 2, 3809: 2, 3795: 2, 3794: 2, 3691: 2, 367

4: 2, 3667: 2, 3660: 2, 3613: 2, 3610: 2, 3568: 2, 3553: 2, 3551: 2, 3533: 2, 3467: 2, 346
4: 2, 3433: 2, 3408: 2, 3358: 2, 3327: 2, 3299: 2, 3277: 2, 3252: 2, 3232: 2, 3189: 2, 316
0: 2, 3111: 2, 3047: 2, 3032: 2, 2977: 2, 2936: 2, 2935: 2, 2906: 2, 2883: 2, 2863: 2, 285
8: 2, 2779: 2, 2736: 2, 2689: 2, 2665: 2, 2663: 2, 2619: 2, 2618: 2, 2608: 2, 2583: 2, 252
1: 2, 2479: 2, 2457: 2, 2424: 2, 2417: 2, 2409: 2, 2373: 2, 2354: 2, 2347: 2, 2341: 2, 234
0: 2, 2320: 2, 2303: 2, 2294: 2, 2265: 2, 2250: 2, 2220: 2, 2171: 2, 2149: 2, 2141: 2, 213
5: 2, 2126: 2, 2121: 2, 2099: 2, 2094: 2, 2076: 2, 2040: 2, 2026: 2, 2019: 2, 2014: 2, 199
8: 2, 1988: 2, 1980: 2, 1951: 2, 1949: 2, 1943: 2, 1917: 2, 1905: 2, 1894: 2, 1882: 2, 186
6: 2, 1856: 2, 1852: 2, 1850: 2, 1847: 2, 1835: 2, 1828: 2, 1815: 2, 1794: 2, 1777: 2, 176
2: 2, 1759: 2, 1758: 2, 1757: 2, 1756: 2, 1753: 2, 1748: 2, 1728: 2, 1723: 2, 1721: 2, 170
6: 2, 1690: 2, 1678: 2, 1677: 2, 1668: 2, 1649: 2, 1644: 2, 1643: 2, 1636: 2, 1635: 2, 163
4: 2, 1626: 2, 1625: 2, 1619: 2, 1615: 2, 1614: 2, 1612: 2, 1611: 2, 1610: 2, 1598: 2, 157
5: 2, 1570: 2, 1564: 2, 1563: 2, 1560: 2, 1555: 2, 1554: 2, 1551: 2, 1543: 2, 1536: 2, 153
3: 2, 1527: 2, 1526: 2, 1524: 2, 1517: 2, 1511: 2, 1510: 2, 1508: 2, 1489: 2, 1487: 2, 147
7: 2, 1460: 2, 1441: 2, 1425: 2, 1419: 2, 1417: 2, 1414: 2, 1407: 2, 1406: 2, 1401: 2, 139
0: 2, 1389: 2, 1388: 2, 1385: 2, 1384: 2, 1381: 2, 1374: 2, 1366: 2, 1356: 2, 1353: 2, 134
2: 2, 1330: 2, 1328: 2, 1325: 2, 1322: 2, 1312: 2, 1310: 2, 1302: 2, 1301: 2, 1295: 2, 129
1: 2, 1290: 2, 1289: 2, 1287: 2, 1284: 2, 1281: 2, 1280: 2, 1277: 2, 1276: 2, 1275: 2, 127
4: 2, 1271: 2, 1266: 2, 1265: 2, 1262: 2, 1257: 2, 1251: 2, 1247: 2, 1237: 2, 1236: 2, 123
3: 2, 1231: 2, 1229: 2, 1226: 2, 1218: 2, 1214: 2, 1209: 2, 1207: 2, 1201: 2, 1199: 2, 119
6: 2, 1180: 2, 1174: 2, 1170: 2, 1167: 2, 1162: 2, 1159: 2, 1156: 2, 1145: 2, 1143: 2, 114
0: 2, 1129: 2, 1127: 2, 1125: 2, 1120: 2, 1114: 2, 1113: 2, 1112: 2, 1110: 2, 1107: 2, 110
2: 2, 1099: 2, 1097: 2, 1093: 2, 1092: 2, 1090: 2, 1087: 2, 1080: 2, 1079: 2, 1078: 2, 107
6: 2, 1070: 2, 1069: 2, 1067: 2, 1066: 2, 1063: 2, 1055: 2, 1052: 2, 1042: 2, 1041: 2, 103
8: 2, 1036: 2, 1031: 2, 1029: 2, 1027: 2, 1026: 2, 1025: 2, 1024: 2, 1023: 2, 1006: 2, 100
0: 2, 997: 2, 989: 2, 984: 2, 982: 2, 980: 2, 976: 2, 975: 2, 974: 2, 971: 2, 965: 2, 962:
2, 958: 2, 952: 2, 951: 2, 948: 2, 947: 2, 943: 2, 942: 2, 934: 2, 931: 2, 928: 2, 925: 2,
917: 2, 916: 2, 911: 2, 908: 2, 904: 2, 902: 2, 896: 2, 895: 2, 893: 2, 891: 2, 889: 2, 88
7: 2, 882: 2, 868: 2, 863: 2, 859: 2, 856: 2, 855: 2, 854: 2, 850: 2, 844: 2, 834: 2, 831:
2, 829: 2, 823: 2, 821: 2, 819: 2, 815: 2, 814: 2, 810: 2, 805: 2, 804: 2, 791: 2, 790: 2,
784: 2, 777: 2, 767: 2, 762: 2, 760: 2, 750: 2, 749: 2, 741: 2, 740: 2, 738: 2, 737: 2, 72
3: 2, 717: 2, 701: 2, 700: 2, 694: 2, 689: 2, 687: 2, 670: 2, 669: 2, 668: 2, 662: 2, 661:
2, 660: 2, 658: 2, 652: 2, 650: 2, 646: 2, 642: 2, 633: 2, 631: 2, 629: 2, 627: 2, 623: 2,
621: 2, 620: 2, 618: 2, 617: 2, 614: 2, 613: 2, 609: 2, 605: 2, 584: 2, 571: 2, 567: 2, 56
6: 2, 561: 2, 551: 2, 546: 2, 529: 2, 525: 2, 523: 2, 502: 2, 497: 2, 483: 2, 445: 2, 391:
2, 311: 2, 153780: 1, 119387: 1, 81567: 1, 67767: 1, 67565: 1, 66943: 1, 66116: 1, 63898:
1, 62380: 1, 53974: 1, 53772: 1, 49650: 1, 48843: 1, 47194: 1, 45981: 1, 45230: 1, 42400:
1, 42144: 1, 41313: 1, 40787: 1, 40452: 1, 39981: 1, 39798: 1, 39298: 1, 38589: 1, 37899:
1, 36855: 1, 36170: 1, 35542: 1, 34236: 1, 33977: 1, 33878: 1, 33527: 1, 32887: 1, 31499:
1, 31293: 1, 29257: 1, 28033: 1, 26833: 1, 26289: 1, 26266: 1, 25939: 1, 25930: 1, 25907:
1, 25110: 1, 24788: 1, 24649: 1, 24288: 1, 24075: 1, 23715: 1, 22633: 1, 22399: 1, 22156:
1, 21986: 1, 21962: 1, 21747: 1, 21687: 1, 21674: 1, 20866: 1, 20488: 1, 20178: 1, 20177:
1, 19989: 1, 19322: 1, 19183: 1, 19128: 1, 19016: 1, 18901: 1, 18612: 1, 18589: 1, 18561:
1, 18481: 1, 18386: 1, 18169: 1, 18044: 1, 18019: 1, 17998: 1, 17915: 1, 17906: 1, 17781:
1, 17488: 1, 17378: 1, 17248: 1, 17234: 1, 17185: 1, 17097: 1, 17073: 1, 16986: 1, 16969:
1, 16771: 1, 16689: 1, 16673: 1, 16559: 1, 16416: 1, 16414: 1, 16249: 1, 15838: 1, 15814:
1, 15809: 1, 15774: 1, 15734: 1, 15559: 1, 15478: 1, 15452: 1, 15275: 1, 15247: 1, 15207:
1, 15154: 1, 14822: 1, 14805: 1, 14663: 1, 14606: 1, 14587: 1, 14501: 1, 14491: 1, 14357:
1, 14266: 1, 14243: 1, 13804: 1, 13702: 1, 13696: 1, 13695: 1, 13604: 1, 13549: 1, 13498:
1, 13363: 1, 13350: 1, 13292: 1, 13160: 1, 12933: 1, 12903: 1, 12867: 1, 12821: 1, 12778:
1, 12697: 1, 12692: 1, 12617: 1, 12590: 1, 12491: 1, 12464: 1, 12437: 1, 12402: 1, 12397:
1, 12356: 1, 12354: 1, 12318: 1, 12255: 1, 12226: 1, 12159: 1, 12144: 1, 12104: 1, 12103:
1, 12101: 1, 12080: 1, 11987: 1, 11945: 1, 11933: 1, 11925: 1, 11906: 1, 11861: 1, 11793:
1, 11771: 1, 11728: 1, 11711: 1, 11678: 1, 11603: 1, 11547: 1, 11506: 1, 11416: 1, 11365:
1, 11336: 1, 11335: 1, 11299: 1, 11168: 1, 11085: 1, 11040: 1, 10906: 1, 10903: 1, 10876:
1, 10854: 1, 10840: 1, 10700: 1, 10607: 1, 10597: 1, 10525: 1, 10351: 1, 10336: 1, 10265:
1, 10259: 1, 10241: 1, 10188: 1, 10161: 1, 10153: 1, 10130: 1, 10052: 1, 10023: 1, 9979:
1, 9920: 1, 9913: 1, 9899: 1, 9876: 1, 9874: 1, 9849: 1, 9812: 1, 9805: 1, 9736: 1, 9566:
1, 9548: 1, 9547: 1, 9534: 1, 9460: 1, 9455: 1, 9411: 1, 9407: 1, 9349: 1, 9348: 1, 9324:
1, 9247: 1, 9216: 1, 9214: 1, 9186: 1, 9179: 1, 9124: 1, 9116: 1, 9090: 1, 9088: 1, 9063:
1, 9048: 1, 9012: 1, 8974: 1, 8962: 1, 8960: 1, 8937: 1, 8928: 1, 8895: 1, 8842: 1, 8810:
1, 8774: 1, 8752: 1, 8736: 1, 8714: 1, 8696: 1, 8693: 1, 8671: 1, 8666: 1, 8617: 1, 8572:
1, 8564: 1, 8525: 1, 8485: 1, 8413: 1, 8403: 1, 8396: 1, 8379: 1, 8350: 1, 8298: 1, 8286:
1, 8268: 1, 8262: 1, 8222: 1, 8205: 1, 8165: 1, 8160: 1, 8152: 1, 8115: 1, 8107: 1, 8089:
1, 8084: 1, 7955: 1, 7930: 1, 7893: 1, 7844: 1, 7816: 1, 7804: 1, 7793: 1, 7786: 1, 7746:
1, 7694: 1, 7688: 1, 7651: 1, 7639: 1, 7630: 1, 7608: 1, 7585: 1, 7581: 1, 7575: 1, 7558:

1, 7553: 1, 7522: 1, 7511: 1, 7507: 1, 7502: 1, 7475: 1, 7471: 1, 7452: 1, 7421: 1, 7407:
1, 7347: 1, 7344: 1, 7282: 1, 7270: 1, 7244: 1, 7240: 1, 7206: 1, 7188: 1, 7183: 1, 7151:
1, 7148: 1, 7143: 1, 7061: 1, 7046: 1, 7043: 1, 7018: 1, 7008: 1, 6994: 1, 6987: 1, 6984:
1, 6957: 1, 6946: 1, 6945: 1, 6913: 1, 6895: 1, 6878: 1, 6864: 1, 6844: 1, 6816: 1, 6814:
1, 6778: 1, 6770: 1, 6760: 1, 6753: 1, 6728: 1, 6683: 1, 6667: 1, 6663: 1, 6644: 1, 6624:
1, 6604: 1, 6601: 1, 6595: 1, 6594: 1, 6586: 1, 6567: 1, 6549: 1, 6538: 1, 6532: 1, 6530:
1, 6526: 1, 6524: 1, 6517: 1, 6482: 1, 6471: 1, 6465: 1, 6451: 1, 6436: 1, 6425: 1, 6420:
1, 6356: 1, 6354: 1, 6353: 1, 6348: 1, 6327: 1, 6322: 1, 6282: 1, 6268: 1, 6267: 1, 6265:
1, 6255: 1, 6234: 1, 6233: 1, 6229: 1, 6228: 1, 6219: 1, 6212: 1, 6183: 1, 6179: 1, 6175:
1, 6087: 1, 6073: 1, 6064: 1, 6063: 1, 6062: 1, 6038: 1, 6026: 1, 6025: 1, 6019: 1, 6015:
1, 6010: 1, 5998: 1, 5984: 1, 5980: 1, 5955: 1, 5938: 1, 5933: 1, 5928: 1, 5895: 1, 5879:
1, 5875: 1, 5874: 1, 5867: 1, 5810: 1, 5806: 1, 5738: 1, 5719: 1, 5716: 1, 5703: 1, 5699:
1, 5696: 1, 5682: 1, 5678: 1, 5665: 1, 5664: 1, 5648: 1, 5639: 1, 5632: 1, 5630: 1, 5614:
1, 5610: 1, 5599: 1, 5581: 1, 5576: 1, 5573: 1, 5572: 1, 5529: 1, 5524: 1, 5522: 1, 5488:
1, 5485: 1, 5472: 1, 5436: 1, 5428: 1, 5416: 1, 5414: 1, 5376: 1, 5359: 1, 5351: 1, 5341:
1, 5306: 1, 5302: 1, 5288: 1, 5281: 1, 5280: 1, 5268: 1, 5267: 1, 5254: 1, 5232: 1, 5221:
1, 5220: 1, 5203: 1, 5192: 1, 5190: 1, 5176: 1, 5174: 1, 5152: 1, 5131: 1, 5129: 1, 5127:
1, 5126: 1, 5066: 1, 5063: 1, 5059: 1, 5044: 1, 5021: 1, 5010: 1, 5003: 1, 4989: 1, 4967:
1, 4965: 1, 4964: 1, 4961: 1, 4948: 1, 4933: 1, 4926: 1, 4910: 1, 4905: 1, 4901: 1, 4897:
1, 4871: 1, 4862: 1, 4854: 1, 4838: 1, 4835: 1, 4833: 1, 4815: 1, 4805: 1, 4797: 1, 4795:
1, 4793: 1, 4780: 1, 4764: 1, 4751: 1, 4749: 1, 4731: 1, 4723: 1, 4722: 1, 4716: 1, 4714:
1, 4703: 1, 4696: 1, 4695: 1, 4690: 1, 4686: 1, 4684: 1, 4679: 1, 4667: 1, 4658: 1, 4655:
1, 4654: 1, 4645: 1, 4636: 1, 4595: 1, 4575: 1, 4567: 1, 4563: 1, 4556: 1, 4547: 1, 4525:
1, 4514: 1, 4498: 1, 4497: 1, 4494: 1, 4488: 1, 4468: 1, 4448: 1, 4446: 1, 4441: 1, 4432:
1, 4408: 1, 4407: 1, 4399: 1, 4377: 1, 4373: 1, 4361: 1, 4353: 1, 4348: 1, 4345: 1, 4334:
1, 4322: 1, 4320: 1, 4316: 1, 4315: 1, 4313: 1, 4303: 1, 4298: 1, 4295: 1, 4285: 1, 4282:
1, 4280: 1, 4279: 1, 4263: 1, 4254: 1, 4237: 1, 4236: 1, 4233: 1, 4227: 1, 4225: 1, 4220:
1, 4216: 1, 4215: 1, 4210: 1, 4208: 1, 4190: 1, 4186: 1, 4185: 1, 4172: 1, 4170: 1, 4168:
1, 4161: 1, 4151: 1, 4150: 1, 4141: 1, 4130: 1, 4128: 1, 4123: 1, 4119: 1, 4116: 1, 4102:
1, 4095: 1, 4086: 1, 4084: 1, 4078: 1, 4064: 1, 4056: 1, 4053: 1, 4043: 1, 4038: 1, 4036:
1, 4024: 1, 4020: 1, 4012: 1, 4004: 1, 4002: 1, 3997: 1, 3992: 1, 3990: 1, 3989: 1, 3973:
1, 3972: 1, 3959: 1, 3954: 1, 3953: 1, 3952: 1, 3945: 1, 3939: 1, 3937: 1, 3934: 1, 3923:
1, 3911: 1, 3873: 1, 3865: 1, 3859: 1, 3858: 1, 3848: 1, 3831: 1, 3827: 1, 3826: 1, 3810:
1, 3806: 1, 3790: 1, 3789: 1, 3786: 1, 3783: 1, 3780: 1, 3775: 1, 3771: 1, 3762: 1, 3761:
1, 3760: 1, 3749: 1, 3747: 1, 3745: 1, 3744: 1, 3737: 1, 3734: 1, 3732: 1, 3724: 1, 3720:
1, 3719: 1, 3713: 1, 3702: 1, 3688: 1, 3670: 1, 3661: 1, 3659: 1, 3647: 1, 3642: 1, 3641:
1, 3635: 1, 3626: 1, 3621: 1, 3619: 1, 3600: 1, 3598: 1, 3578: 1, 3576: 1, 3569: 1, 3566:
1, 3565: 1, 3562: 1, 3560: 1, 3558: 1, 3556: 1, 3555: 1, 3544: 1, 3543: 1, 3538: 1, 3536:
1, 3532: 1, 3523: 1, 3520: 1, 3512: 1, 3511: 1, 3507: 1, 3505: 1, 3501: 1, 3498: 1, 3495:
1, 3492: 1, 3487: 1, 3473: 1, 3471: 1, 3469: 1, 3466: 1, 3463: 1, 3455: 1, 3447: 1, 3442:
1, 3441: 1, 3438: 1, 3432: 1, 3431: 1, 3425: 1, 3419: 1, 3417: 1, 3409: 1, 3406: 1, 3404:
1, 3403: 1, 3402: 1, 3401: 1, 3390: 1, 3386: 1, 3385: 1, 3377: 1, 3366: 1, 3363: 1, 3360:
1, 3359: 1, 3349: 1, 3344: 1, 3341: 1, 3332: 1, 3330: 1, 3326: 1, 3320: 1, 3307: 1, 3295:
1, 3293: 1, 3281: 1, 3279: 1, 3276: 1, 3271: 1, 3268: 1, 3267: 1, 3263: 1, 3255: 1, 3254:
1, 3250: 1, 3244: 1, 3241: 1, 3226: 1, 3223: 1, 3222: 1, 3213: 1, 3204: 1, 3203: 1, 3198:
1, 3194: 1, 3193: 1, 3191: 1, 3183: 1, 3181: 1, 3175: 1, 3172: 1, 3171: 1, 3169: 1, 3164:
1, 3162: 1, 3161: 1, 3156: 1, 3153: 1, 3145: 1, 3142: 1, 3134: 1, 3131: 1, 3124: 1, 3123:
1, 3116: 1, 3113: 1, 3112: 1, 3109: 1, 3102: 1, 3095: 1, 3086: 1, 3076: 1, 3074: 1, 3063:
1, 3060: 1, 3058: 1, 3052: 1, 3050: 1, 3020: 1, 3015: 1, 3000: 1, 2992: 1, 2986: 1, 2980:
1, 2974: 1, 2969: 1, 2961: 1, 2945: 1, 2944: 1, 2939: 1, 2938: 1, 2934: 1, 2931: 1, 2930:
1, 2922: 1, 2921: 1, 2903: 1, 2898: 1, 2895: 1, 2884: 1, 2881: 1, 2877: 1, 2875: 1, 2872:
1, 2870: 1, 2868: 1, 2865: 1, 2864: 1, 2862: 1, 2848: 1, 2847: 1, 2844: 1, 2841: 1, 2834:
1, 2826: 1, 2824: 1, 2823: 1, 2822: 1, 2816: 1, 2809: 1, 2807: 1, 2798: 1, 2797: 1, 2793:
1, 2773: 1, 2771: 1, 2767: 1, 2762: 1, 2761: 1, 2758: 1, 2753: 1, 2751: 1, 2750: 1, 2749:
1, 2745: 1, 2743: 1, 2737: 1, 2728: 1, 2727: 1, 2724: 1, 2721: 1, 2714: 1, 2710: 1, 2706:
1, 2703: 1, 2697: 1, 2694: 1, 2688: 1, 2684: 1, 2679: 1, 2675: 1, 2673: 1, 2670: 1, 2669:
1, 2661: 1, 2653: 1, 2647: 1, 2644: 1, 2640: 1, 2639: 1, 2638: 1, 2635: 1, 2633: 1, 2628:
1, 2624: 1, 2622: 1, 2620: 1, 2617: 1, 2615: 1, 2613: 1, 2612: 1, 2611: 1, 2609: 1, 2607:
1, 2606: 1, 2604: 1, 2596: 1, 2594: 1, 2591: 1, 2584: 1, 2581: 1, 2563: 1, 2561: 1, 2556:
1, 2555: 1, 2554: 1, 2547: 1, 2545: 1, 2544: 1, 2542: 1, 2540: 1, 2539: 1, 2528: 1, 2526:
1, 2523: 1, 2522: 1, 2520: 1, 2519: 1, 2518: 1, 2516: 1, 2514: 1, 2502: 1, 2501: 1, 2498:
1, 2497: 1, 2496: 1, 2494: 1, 2484: 1, 2481: 1, 2478: 1, 2475: 1, 2473: 1, 2470: 1, 2465:
1, 2464: 1, 2461: 1, 2455: 1, 2452: 1, 2451: 1, 2447: 1, 2445: 1, 2438: 1, 2436: 1, 2433:
1, 2432: 1, 2430: 1, 2426: 1, 2422: 1, 2420: 1, 2418: 1, 2416: 1, 2413: 1, 2412: 1, 2411:
1, 2404: 1, 2402: 1, 2401: 1, 2397: 1, 2393: 1, 2392: 1, 2390: 1, 2384: 1, 2381: 1, 2377:
1, 2375: 1, 2372: 1, 2370: 1, 2368: 1, 2367: 1, 2362: 1, 2349: 1, 2348: 1, 2346: 1, 2344:

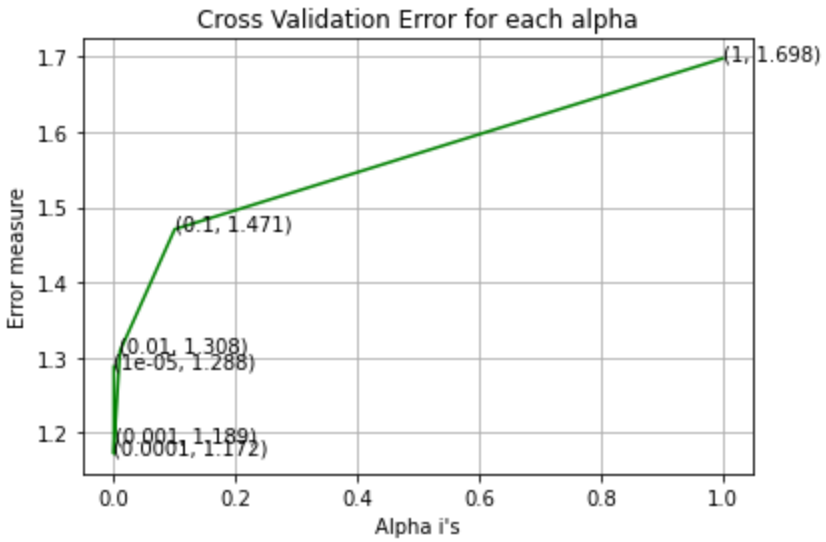
```
1, 2342: 1, 2339: 1, 2338: 1, 2333: 1, 2330: 1, 2329: 1, 2327: 1, 2325: 1, 2323: 1, 2318:
1, 2313: 1, 2310: 1, 2306: 1, 2301: 1, 2295: 1, 2293: 1, 2292: 1, 2286: 1, 2284: 1, 2278:
1, 2276: 1, 2273: 1, 2270: 1, 2266: 1, 2263: 1, 2260: 1, 2255: 1, 2253: 1, 2252: 1, 2251:
1, 2249: 1, 2246: 1, 2244: 1, 2243: 1, 2242: 1, 2234: 1, 2230: 1, 2227: 1, 2226: 1, 2225:
1, 2217: 1, 2204: 1, 2203: 1, 2201: 1, 2199: 1, 2198: 1, 2197: 1, 2189: 1, 2186: 1, 2185:
1, 2183: 1, 2180: 1, 2173: 1, 2172: 1, 2168: 1, 2167: 1, 2165: 1, 2163: 1, 2162: 1, 2160:
1, 2156: 1, 2155: 1, 2153: 1, 2152: 1, 2151: 1, 2147: 1, 2143: 1, 2138: 1, 2133: 1, 2130:
1, 2125: 1, 2124: 1, 2122: 1, 2120: 1, 2118: 1, 2115: 1, 2114: 1, 2113: 1, 2112: 1, 2108:
1, 2107: 1, 2104: 1, 2103: 1, 2100: 1, 2096: 1, 2095: 1, 2093: 1, 2088: 1, 2086: 1, 2085:
1, 2083: 1, 2077: 1, 2075: 1, 2073: 1, 2071: 1, 2060: 1, 2058: 1, 2057: 1, 2053: 1, 2052:
1, 2047: 1, 2046: 1, 2042: 1, 2039: 1, 2036: 1, 2028: 1, 2025: 1, 2024: 1, 2023: 1, 2016:
1, 2013: 1, 2012: 1, 2009: 1, 2007: 1, 2006: 1, 2004: 1, 2001: 1, 1999: 1, 1993: 1, 1991:
1, 1989: 1, 1987: 1, 1986: 1, 1984: 1, 1979: 1, 1976: 1, 1973: 1, 1972: 1, 1968: 1, 1967:
1, 1966: 1, 1964: 1, 1963: 1, 1962: 1, 1961: 1, 1960: 1, 1959: 1, 1957: 1, 1954: 1, 1953:
1, 1950: 1, 1947: 1, 1945: 1, 1941: 1, 1940: 1, 1937: 1, 1931: 1, 1930: 1, 1929: 1, 1928:
1, 1925: 1, 1918: 1, 1910: 1, 1903: 1, 1901: 1, 1900: 1, 1899: 1, 1898: 1, 1897: 1, 1895:
1, 1889: 1, 1886: 1, 1885: 1, 1884: 1, 1883: 1, 1879: 1, 1872: 1, 1870: 1, 1867: 1, 1862:
1, 1858: 1, 1854: 1, 1853: 1, 1849: 1, 1846: 1, 1844: 1, 1842: 1, 1841: 1, 1836: 1, 1831:
1, 1826: 1, 1825: 1, 1819: 1, 1817: 1, 1814: 1, 1806: 1, 1804: 1, 1803: 1, 1802: 1, 1801:
1, 1800: 1, 1799: 1, 1798: 1, 1796: 1, 1793: 1, 1792: 1, 1791: 1, 1790: 1, 1783: 1, 1780:
1, 1778: 1, 1773: 1, 1771: 1, 1769: 1, 1768: 1, 1767: 1, 1766: 1, 1764: 1, 1763: 1, 1754:
1, 1751: 1, 1749: 1, 1747: 1, 1742: 1, 1735: 1, 1734: 1, 1733: 1, 1732: 1, 1730: 1, 1727:
1, 1724: 1, 1722: 1, 1719: 1, 1715: 1, 1713: 1, 1712: 1, 1709: 1, 1708: 1, 1703: 1, 1698:
1, 1697: 1, 1696: 1, 1687: 1, 1675: 1, 1672: 1, 1671: 1, 1670: 1, 1669: 1, 1666: 1, 1664:
1, 1661: 1, 1660: 1, 1658: 1, 1655: 1, 1654: 1, 1650: 1, 1646: 1, 1645: 1, 1640: 1, 1631:
1, 1629: 1, 1628: 1, 1620: 1, 1617: 1, 1608: 1, 1606: 1, 1604: 1, 1603: 1, 1597: 1, 1595:
1, 1594: 1, 1593: 1, 1590: 1, 1586: 1, 1584: 1, 1581: 1, 1578: 1, 1577: 1, 1574: 1, 1568:
1, 1566: 1, 1565: 1, 1557: 1, 1556: 1, 1553: 1, 1548: 1, 1547: 1, 1545: 1, 1540: 1, 1532:
1, 1531: 1, 1522: 1, 1521: 1, 1520: 1, 1518: 1, 1515: 1, 1513: 1, 1512: 1, 1503: 1, 1502:
1, 1497: 1, 1496: 1, 1493: 1, 1490: 1, 1483: 1, 1475: 1, 1472: 1, 1471: 1, 1470: 1, 1467:
1, 1465: 1, 1464: 1, 1463: 1, 1461: 1, 1451: 1, 1449: 1, 1443: 1, 1442: 1, 1438: 1, 1437:
1, 1436: 1, 1435: 1, 1434: 1, 1431: 1, 1429: 1, 1420: 1, 1418: 1, 1415: 1, 1413: 1, 1412:
1, 1411: 1, 1404: 1, 1403: 1, 1400: 1, 1399: 1, 1394: 1, 1393: 1, 1383: 1, 1382: 1, 1380:
1, 1379: 1, 1373: 1, 1372: 1, 1371: 1, 1370: 1, 1367: 1, 1365: 1, 1361: 1, 1360: 1, 1355:
1, 1354: 1, 1343: 1, 1340: 1, 1339: 1, 1337: 1, 1335: 1, 1334: 1, 1333: 1, 1332: 1, 1329:
1, 1326: 1, 1324: 1, 1320: 1, 1319: 1, 1316: 1, 1315: 1, 1308: 1, 1307: 1, 1306: 1, 1305:
1, 1304: 1, 1303: 1, 1300: 1, 1297: 1, 1294: 1, 1293: 1, 1288: 1, 1286: 1, 1285: 1, 1282:
1, 1279: 1, 1273: 1, 1272: 1, 1270: 1, 1269: 1, 1264: 1, 1263: 1, 1261: 1, 1260: 1, 1256:
1, 1254: 1, 1253: 1, 1252: 1, 1250: 1, 1245: 1, 1242: 1, 1241: 1, 1240: 1, 1239: 1, 1238:
1, 1235: 1, 1234: 1, 1232: 1, 1230: 1, 1225: 1, 1224: 1, 1223: 1, 1215: 1, 1211: 1, 1208:
1, 1205: 1, 1197: 1, 1194: 1, 1193: 1, 1191: 1, 1189: 1, 1186: 1, 1183: 1, 1181: 1, 1175:
1, 1173: 1, 1163: 1, 1161: 1, 1160: 1, 1157: 1, 1144: 1, 1142: 1, 1139: 1, 1137: 1, 1134:
1, 1133: 1, 1128: 1, 1126: 1, 1124: 1, 1123: 1, 1122: 1, 1121: 1, 1111: 1, 1106: 1, 1105:
1, 1103: 1, 1101: 1, 1100: 1, 1096: 1, 1085: 1, 1082: 1, 1075: 1, 1073: 1, 1068: 1, 1065:
1, 1062: 1, 1059: 1, 1051: 1, 1049: 1, 1046: 1, 1043: 1, 1039: 1, 1037: 1, 1034: 1, 1030:
1, 1028: 1, 1020: 1, 1017: 1, 1016: 1, 1015: 1, 1011: 1, 1010: 1, 1008: 1, 1007: 1, 1005:
1, 1004: 1, 1001: 1, 996: 1, 995: 1, 994: 1, 992: 1, 988: 1, 986: 1, 981: 1, 979: 1, 977:
1, 970: 1, 968: 1, 960: 1, 959: 1, 957: 1, 956: 1, 955: 1, 953: 1, 945: 1, 936: 1, 933: 1,
927: 1, 924: 1, 918: 1, 913: 1, 910: 1, 909: 1, 907: 1, 898: 1, 888: 1, 886: 1, 885: 1, 88
3: 1, 880: 1, 878: 1, 876: 1, 875: 1, 874: 1, 861: 1, 860: 1, 852: 1, 851: 1, 848: 1, 847:
1, 843: 1, 840: 1, 837: 1, 835: 1, 833: 1, 832: 1, 830: 1, 825: 1, 818: 1, 816: 1, 811: 1,
808: 1, 807: 1, 806: 1, 803: 1, 801: 1, 795: 1, 794: 1, 793: 1, 788: 1, 779: 1, 775: 1, 77
0: 1, 768: 1, 765: 1, 753: 1, 745: 1, 727: 1, 726: 1, 721: 1, 720: 1, 714: 1, 712: 1, 710:
1, 706: 1, 704: 1, 697: 1, 696: 1, 688: 1, 686: 1, 685: 1, 677: 1, 676: 1, 663: 1, 645: 1,
644: 1, 635: 1, 598: 1, 593: 1, 592: 1, 587: 1, 548: 1, 541: 1, 522: 1, 478: 1, 465: 1, 46
4: 1, 424: 1})
```

In [57]:

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

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
```

```
# 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
# 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

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
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:
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
```

```
For values of alpha =  1e-05 The log loss is: 1.2877826969150827
For values of alpha =  0.0001 The log loss is: 1.17150042181323
For values of alpha =  0.001 The log loss is: 1.1892513964255067
For values of alpha =  0.01 The log loss is: 1.3081596381565888
For values of alpha =  0.1 The log loss is: 1.4706105445442743
For values of alpha =  1 The log loss is: 1.6979796144478045
```




```
For values of best alpha =  0.0001 The train log loss is: 0.6397601761381825
For values of best alpha =  0.0001 The cross validation log loss is: 1.17150042181323
For values of best alpha =  0.0001 The test log loss is: 1.1157796174637893
```

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

Ans. Yes, it seems like!

In [58]:
```
def get_intersec_text(df):
    df_text_vec = CountVectorizer(min_df=3)
    df_text_fea = df_text_vec.fit_transform(df['TEXT'])
    df_text_features = df_text_vec.get_feature_names()

    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 [59]:
```
len1,len2 = get_intersec_text(test_df)
print(np.round((len2/len1)*100, 3), "% of word of test data appeared in train data")
len1,len2 = get_intersec_text(cv_df)
print(np.round((len2/len1)*100, 3), "% of word of Cross Validation appeared in train data"
```

```
97.514 % of word of test data appeared in train data
97.014 % of word of Cross Validation appeared in train data
```

4. Machine Learning Models

In [60]:
```
#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
    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.s
    plot_confusion_matrix(test_y, pred_y)
```

In [61]:

```
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 [62]:

```
# 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 = CountVectorizer(min_df=3)

    gene_vec = gene_count_vec.fit(train_df['Gene'])
    var_vec  = var_count_vec.fit(train_df['Variation'])
    text_vec = text_count_vec.fit(train_df['TEXT'])

    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 = gene_vec.get_feature_names()[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,y
        elif (v < fea1_len+fea2_len):
            word = var_vec.get_feature_names()[v-(fea1_len)]
            yes_no = True if word == var else False
            if yes_no:
                word_present += 1
                print(i, "variation feature [{}] present in test data point [{}]".format(w
        else:
            word = text_vec.get_feature_names()[v-(fea1_len+fea2_len)]
            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,y

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

Stacking the three types of features

In [63]:

```
# 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_feat
test_gene_var_onehotCoding = hstack((test_gene_feature_onehotCoding,test_variation_feature
cv_gene_var_onehotCoding = hstack((cv_gene_feature_onehotCoding,cv_variation_feature_oneh

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

test_x_onehotCoding = hstack((test_gene_var_onehotCoding, test_text_feature_onehotCoding))
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']))
```

In [64]:

```
print("One hot encoding features :")
print("(number of data points * number of features) in train data = ", train_x_onehotCodin
print("(number of data points * number of features) in test data = ", test_x_onehotCoding.
print("(number of data points * number of features) in cross validation data =", cv_x_oneh
```

```
One hot encoding features :
(number of data points * number of features) in train data =  (2124, 55264)
(number of data points * number of features) in test data =  (665, 55264)
(number of data points * number of features) in cross validation data = (532, 55264)
```

In [65]:

```
train_gene_var_responseCoding = np.hstack((train_gene_feature_responseCoding,train_variati
test_gene_var_responseCoding = np.hstack((test_gene_feature_responseCoding,test_variation_
cv_gene_var_responseCoding = np.hstack((cv_gene_feature_responseCoding,cv_variation_featur

train_x_responseCoding = np.hstack((train_gene_var_responseCoding, train_text_feature_resp
test_x_responseCoding = np.hstack((test_gene_var_responseCoding, test_text_feature_respons
cv_x_responseCoding = np.hstack((cv_gene_var_responseCoding, cv_text_feature_responseCodin
```

In [66]:

```
print(" Response encoding features :")
print("(number of data points * number of features) in train data = ", train_x_responseCod
print("(number of data points * number of features) in test data = ", test_x_responseCodin
print("(number of data points * number of features) in cross validation data =", cv_x_resp
```

```
 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

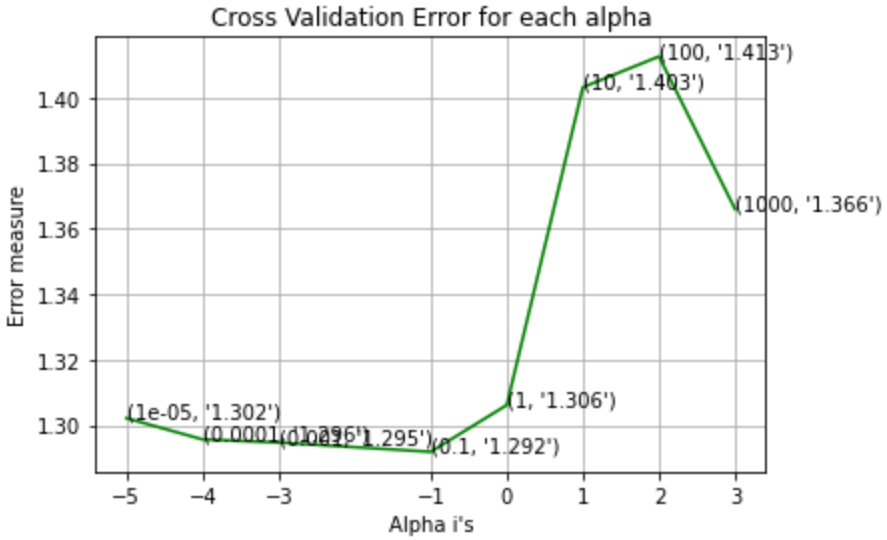
In [67]:

```
# find more about Multinomial Naive base function here http://scikit-learn.org/stable/modu
# -------------------------
# 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/naiv
# -----------------------
```

```
# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ----------------------------
# 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/naiv
# ----------------------


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 estima
    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
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:
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
```

```
for alpha = 1e-05
Log Loss : 1.30218234162866
for alpha = 0.0001
Log Loss : 1.2957016594135218
for alpha = 0.001
Log Loss : 1.2947623117564007
for alpha = 0.1
Log Loss : 1.2919062049210135
for alpha = 1
```

```
Log Loss : 1.3061770123980903
for alpha = 10
Log Loss : 1.4031782775116215
for alpha = 100
Log Loss : 1.4125795628424025
for alpha = 1000
Log Loss : 1.3660051398852722
```




```
For values of best alpha =  0.1 The train log loss is: 0.8439790645631208
For values of best alpha =  0.1 The cross validation log loss is: 1.2919062049210135
For values of best alpha =  0.1 The test log loss is: 1.2274039487856714
```

4.1.1.2. Testing the model with best hyper paramters

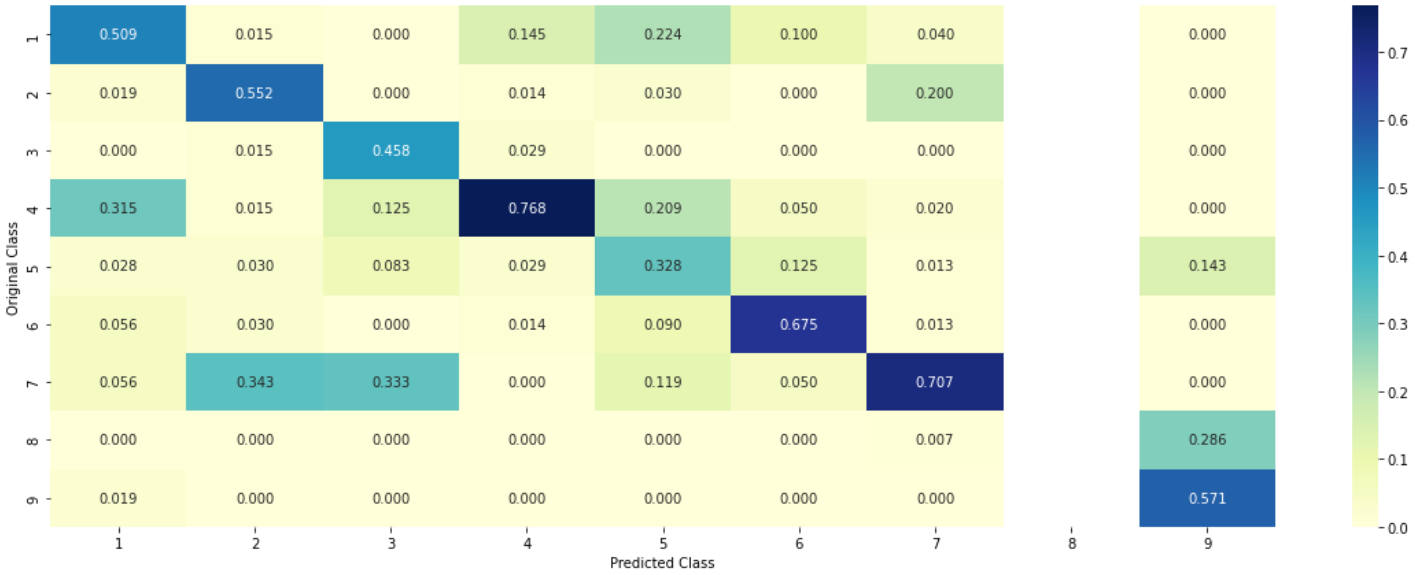
In [68]:

```
# find more about Multinomial Naive base function here http://scikit-learn.org/stable/modu
# -------------------------
# 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/naiv
# -------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ----------------------------
# 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))
```

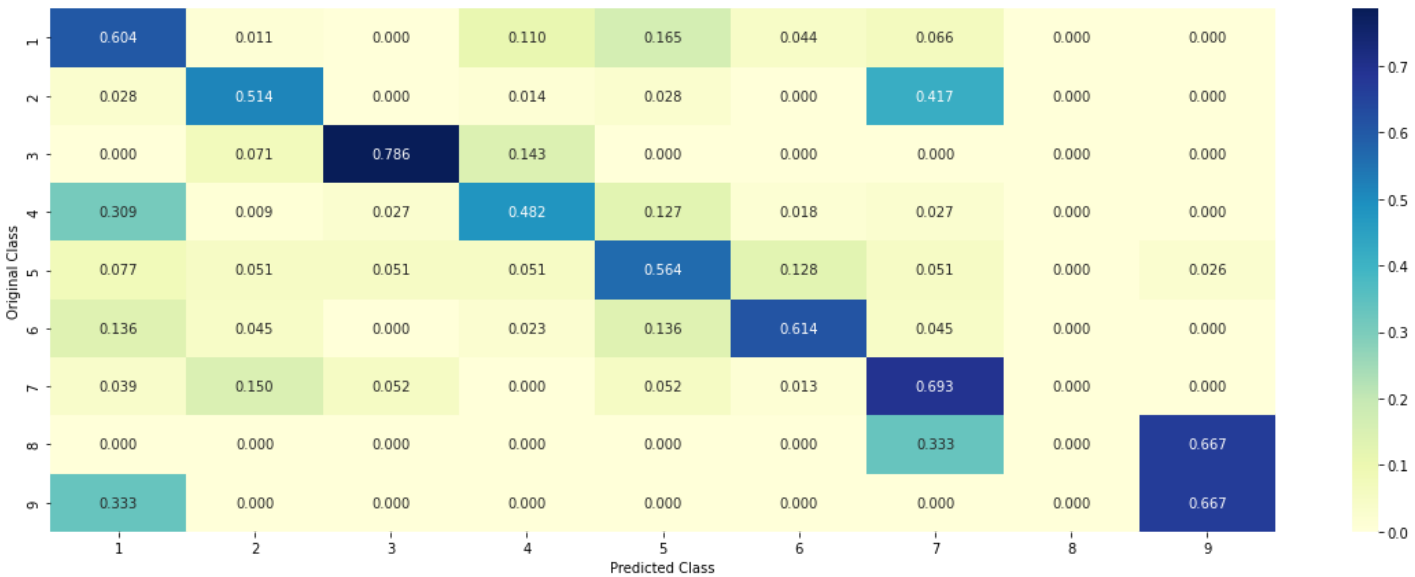
```
print("Number of missclassified point :", np.count_nonzero((sig_clf.predict(cv_x_onehotCod
plot_confusion_matrix(cv_y, sig_clf.predict(cv_x_onehotCoding.toarray()))
```

```
Log Loss : 1.2919062049210135
Number of missclassified point : 0.40789473684210525
-------------------- Confusion matrix --------------------
```


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




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



4.1.1.3. Feature Importance, Correctly classified point

In [69]:

```
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_onehotCodinc
print("Actual Class :", test_y[test_point_index])
indices=np.argsort(-1*clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.0712 0.067  0.0154 0.1018 0.0372 0.0361 0.6642 0.004
0.0031]]
Actual Class : 7
--------------------------------------------------
17 Text feature [kinase] present in test data point [True]
18 Text feature [presence] present in test data point [True]
19 Text feature [activating] present in test data point [True]
20 Text feature [downstream] present in test data point [True]
21 Text feature [independent] present in test data point [True]
22 Text feature [inhibitor] present in test data point [True]
23 Text feature [contrast] present in test data point [True]
24 Text feature [expressing] present in test data point [True]
25 Text feature [well] present in test data point [True]
26 Text feature [compared] present in test data point [True]
27 Text feature [recently] present in test data point [True]
28 Text feature [potential] present in test data point [True]
29 Text feature [growth] present in test data point [True]
30 Text feature [activation] present in test data point [True]
31 Text feature [showed] present in test data point [True]
32 Text feature [also] present in test data point [True]
33 Text feature [shown] present in test data point [True]
34 Text feature [cells] present in test data point [True]
35 Text feature [however] present in test data point [True]
36 Text feature [factor] present in test data point [True]
37 Text feature [cell] present in test data point [True]
38 Text feature [previously] present in test data point [True]
39 Text feature [10] present in test data point [True]
40 Text feature [treated] present in test data point [True]
41 Text feature [suggest] present in test data point [True]
42 Text feature [similar] present in test data point [True]
43 Text feature [obtained] present in test data point [True]
44 Text feature [higher] present in test data point [True]
45 Text feature [addition] present in test data point [True]
46 Text feature [inhibition] present in test data point [True]
47 Text feature [mutations] present in test data point [True]
48 Text feature [may] present in test data point [True]
49 Text feature [without] present in test data point [True]
50 Text feature [inhibitors] present in test data point [True]
51 Text feature [studies] present in test data point [True]
56 Text feature [found] present in test data point [True]
57 Text feature [described] present in test data point [True]
58 Text feature [respectively] present in test data point [True]
59 Text feature [interestingly] present in test data point [True]
60 Text feature [figure] present in test data point [True]
61 Text feature [reported] present in test data point [True]
62 Text feature [1a] present in test data point [True]
63 Text feature [12] present in test data point [True]
64 Text feature [sensitive] present in test data point [True]
65 Text feature [concentrations] present in test data point [True]
66 Text feature [total] present in test data point [True]
67 Text feature [observed] present in test data point [True]
68 Text feature [absence] present in test data point [True]
69 Text feature [activated] present in test data point [True]
```

```
72 Text feature [confirmed] present in test data point [True]
73 Text feature [including] present in test data point [True]
74 Text feature [using] present in test data point [True]
75 Text feature [new] present in test data point [True]
76 Text feature [approximately] present in test data point [True]
77 Text feature [various] present in test data point [True]
78 Text feature [constitutive] present in test data point [True]
79 Text feature [consistent] present in test data point [True]
80 Text feature [small] present in test data point [True]
81 Text feature [3a] present in test data point [True]
82 Text feature [although] present in test data point [True]
83 Text feature [constitutively] present in test data point [True]
84 Text feature [3b] present in test data point [True]
85 Text feature [increase] present in test data point [True]
86 Text feature [thus] present in test data point [True]
87 Text feature [either] present in test data point [True]
89 Text feature [mutation] present in test data point [True]
90 Text feature [show] present in test data point [True]
91 Text feature [three] present in test data point [True]
92 Text feature [phosphorylation] present in test data point [True]
93 Text feature [identified] present in test data point [True]
94 Text feature [therapeutic] present in test data point [True]
95 Text feature [increased] present in test data point [True]
96 Text feature [hours] present in test data point [True]
97 Text feature [due] present in test data point [True]
98 Text feature [approved] present in test data point [True]
Out of the top  100  features  75 are present in query point
```

In [70]:

```
test_df['TEXT'].iloc[test_point_index]
```

Out[70]:

```
'non small cell lung cancer leading cause death cancer united states chemotherapy slightly
prolongs survival among patients advanced disease cost clinically significant adverse effe
cts 1 success abl tyrosine kinase inhibitor imatinib treatment chronic myeloid leukemia cm
l demonstrated effectiveness targeting critical genetic lesion promotes proliferative sign
als cancer cells 2 gefitinib targets atp cleft within tyrosine kinase epidermal growth fac
tor receptor egfr 3 overexpressed 40 80 percent non small cell lung cancers many epithelia
l cancers 4 egfr signaling triggered binding growth factors epidermal growth factor egf re
sulting dimerization egfr molecules heterodimerization closely related receptors her2 neu
autophosphorylation transphosphorylation receptors tyrosine kinase domains leads recruitme
nt downstream effectors activation proliferative cell survival signals 5 despite ubiquitou
s expression inactivation egfr gene mouse causes minimal defects 6 7 suggesting pharmacolo
gic inhibition egfr gefitinib adverse effects gefitinib inhibits growth cancer derived cel
l lines tumor xenografts although effect well correlated level expression egfr related mem
bers erbb family receptors 3 initial clinical studies gefitinib minimal adverse effects 8
10 tumor responses observed 10 19 percent patients chemotherapy refractory advanced non sm
all cell lung cancer 11 12 addition gefitinib traditional chemotherapy provided benefit 13
14 even gliomas finding frequent amplification rearrangements egfr gene suggests egfr play
s important role gefitinib failed induce clinically significant responses 15 16 despite di
scouraging results remarkably rapid often profound response gefitinib subgroup patients no
n small cell lung cancer led approval single drug therapy refractory lung cancer 17 evalua
ted tumors patients dramatic responses determine underlying mechanisms methods nucleotide
sequence analysis tumor specimens tumor specimens obtained diagnostic surgical procedures
patients non small cell lung cancer subsequently treated gefitinib according protocol appr
oved institutional review board massachusetts general hospital boston frozen tumor specime
ns along matched normal tissue available four patients paraffin embedded material used pat
ients addition specimens 25 patients primary non small cell lung cancer exposed gefitinib
15 bronchoalveolar cancer 7 adenocarcinoma 3 large cell lung cancer matched normal tissues
obtained massachusetts general hospital tumor bank mutational analysis entire egfr coding
sequence dna extracted specimens 28 exons amplified uncloned polymerase chain reaction pcr
fragments sequenced analyzed sense antisense directions presence heterozygous mutations se
quence variants confirmed multiple independent pcr amplifications primer sequences amplifi
cation conditions explained supplementary appendix egfr mutations exons 19 21 also sought
primary tumors breast 15 specimens colon 20 specimens kidney 16 specimens pancreas 40 spec
imens brain 4 specimens along panel 108 cancer derived cell lines representing diverse his
tologic types listed supplementary appendix functional analysis mutant egfr constructs l85
```

8r dell747 p753inss mutations introduced full length egfr coding sequence use site directe
d mutagenesis inserted cytomegalovirus promoter driven expression construct puse upstate c
os 7 cells transfected lipofectamine 2000 invitrogen 1 g expression constructs replated 18
hours later concentration 5 104 cells per well 12 well plates costar dulbecco minimal esse
ntial medium without fetal calf serum 16 hours serum starvation cells stimulated 10 ng egf
per milliliter sigma determine whether mutant receptors inhibited gefitinib drug added cul
ture medium three hours addition 100 ng egf per milliliter cells exposed egf 30 minutes ce
ll lysates prepared 100 l laemmli lysis buffer followed resolution proteins 10 percent sod
ium dodecyl sulfate polyacrylamide gel electrophoresis transfer membranes western blot ana
lysis use enhanced chemiluminescence reagent amersham autophosphorylation egfr measured an
tibody phosphotyrosine position 1068 standardized total protein expression shown use antib
ody egfr working concentration 1 1000 cell signaling technology results clinical character
istics patients response gefitinib patients advanced chemotherapy refractory non small cel
l lung cancer treated gefitinib single agent since 2000 massachusetts general hospital tot
al 275 patients treated approval may 2003 food drug administration fda part compassionate
use expanded access program subsequently use commercial supply period 25 patients identifi
ed physicians clinically significant responses drug clinically significant response define
d partial response according response evaluation criteria solid tumors18 patients measurab
le disease patients whose tumor burden could quantified use criteria response assessed two
physicians table 1table 1 characteristics nine patients non small cell lung cancer respons
e gefitinib shows clinical characteristics nine patients tumor specimens obtained time dia
gnosis available tissue available patients response gefitinib commonly diagnostic specimen
s limited needle aspirates group nine patients derived substantial benefit gefitinib thera
py median duration survival start drug treatment exceeded 18 months median duration therap
y greater 16 months consistent previous reports found patients response gefitinib women ne
ver smoked bronchoalveolar tumors 11 12 patient 6 representative cohort patient 32 year ol
d man history smoking presented multiple brain lesions bronchoalveolar carcinoma right lun
g treated whole brain radiotherapy followed series chemotherapy regimens carboplatin gemci
tabine docetaxel vinorelbine tumor respond declining functional status progressive lung tu
mor burden started therapy 250 mg gefitinib per day dyspnea promptly improved computed tom
ography lung six weeks initiation treatment revealed dramatic improvement figure 1figure 1
example response gefitinib patient refractory non small cell lung cancer egfr mutations pa
tients response gefitinib hypothesized patients non small cell lung cancer striking respon
ses gefitinib somatic mutations egfr gene would indicate essential role egfr signaling pat
hway tumor search mutations first looked rearrangements within extracellular domain egfr c
haracteristic gliomas15 none detected therefore sequenced entire coding region gene using
pcr amplification individual exons heterozygous mutations observed eight nine patients clu
stered within tyrosine kinase domain egfr table 2table 2 somatic mutations tyrosine kinase
domain egfr patients non small cell lung cancer figure 2figure 2 mutations egfr gene gefit
inib responsive tumors four tumors frame deletions removing amino acids 746 750 dele746 a7
50 patient 1 747 751 dell747 t751inss patient 2 747 753 dell747 p753inss patients 3 4 seco
nd third deletions associated insertion serine residue resulting generation novel codon de
letion breakpoint remarkably deletions overlapped sharing deletion four amino acids leucin
e arginine glutamic acid alanine codons 747 750 within exon 19 another three tumors amino
acid substitutions within exon 21 leucine arginine codon 858 l858r patients 5 6 leucine gl
utamine codon 861 l861q patient 7 l861q mutation particular interest since amino acid chan
ge mouse egfr gene responsible dark skin dsk5 trait associated altered egfr signaling 19 f
ourth missense mutation tyrosine kinase domain resulted substitution cysteine glycine codo
n 719 within exon 18 g719c patient 8 matched normal tissue available patients 1 4 5 6 show
ed wild type sequence indicating mutations arisen somatically tumor formation comparison m
utations observed seven patients non small cell lung cancer response gefitinib p 0 001 two
sided fisher exact test prevalence specific egfr mutations non small cell lung cancer type
s cancer unlike gliomas rearrangements affecting egfr extracellular domain extensively stu
died 15 frequency egfr mutations non small cell lung cancer defined therefore sequenced en
tire coding region gene tumors 25 patients primary non small cell lung cancer involved gef
itinib study including 15 bronchoalveolar lung cancer associated responsiveness gefitinib
previous clinical trials 11 12 heterozygous mutations detected two patients bronchoalveola
r cancers frame deletions kinase domain identical found patients response gefitinib namely
dell747 p753inss dele746 a750 table 2 given apparent clustering egfr mutations sequenced e
xons 19 21 total 95 primary tumors 108 cancer derived cell lines representing diverse tumo
r types see supplementary appendix mutations detected suggesting subgroup cancers egfr sig
naling may play critical role tumorigenesis harbor egfr mutations increase egf induced act
ivation gefitinib induced inhibition mutant egfr proteins study functional properties enco
ded mutations expressed receptor l747 p753inss deletion receptor l858r missense mutation c
ultured cells transient transfection wild type mutant constructs cos 7 cells demonstrated
equivalent expression levels indicating mutations affect stability protein egfr activation

quantified measuring phosphorylation tyrosine1068 residue commonly used marker autophospho
rylation egfr 20 absence serum associated growth factors neither wild type mutant egfr dem
onstrated autophosphorylation figure 3afigure 3 enhanced egf dependent activation mutant e
gfr increased sensitivity mutant egfr gefitinib figure 3b however addition egf doubled tri
pled activation mutant egfrs compared activation wild type receptor moreover whereas activ
ation normal egfr regulated 15 minutes consistent internalization receptor two mutant rece
ptors demonstrated continued activation three hours figure 3a similar results obtained use
antibodies measure total phosphorylation egfr addition egf data shown since seven eight eg
fr tyrosine kinase mutations reside near atp cleft targeted gefitinib assessed whether mut
ant proteins altered sensitivity inhibitor egf induced autophosphorylation egfr measured c
ells pretreated various concentrations gefitinib remarkably mutant receptors sensitive wil
d type receptor inhibition gefitinib wild type egfr inhibited 50 percent gefitinib concent
ration 0 1 completely inhibited concentration 2 0 whereas respective values two mutant pro
teins 0 015 0 2 figure 3c figure 3d difference drug sensitivity may clinically relevant si
nce pharmacokinetic studies indicate daily oral administration 400 600 mg gefitinib result
s mean steady state trough plasma concentration 1 1 1 4 whereas currently recommended dail
y dose 250 mg leads mean trough concentration 0 4 21 discussion gefitinib first agent desi
gned known molecular target receive fda approval treatment lung cancer yet activity limite
d subgroup patients non small cell lung cancer identified specific activating mutations wi
thin tyrosine kinase domain egfr molecular correlate dramatic responses gefitinib subgroup
somatic mutations identified eight nine patients response gefitinib ninth patient may unde
tected mutation mutation heterodimerization partner egfr results together finding egfr mut
ations tumors 2 25 patients non small cell lung cancer received gefitinib 8 percent sugges
t mutations account majority responses gefitinib reported clinical studies 11 12 heterozyg
ous nature egfr mutations suggests exert dominant oncogenic effect evident despite presenc
e second wild type allele presence additive specific gain function supported observation i
dentical somatic mutations different tumors mutations clustered near atp cleft tyrosine ki
nase domain flank amino acids shown crystallographic studies mediate binding 4 anilinoquin
azoline compounds gefitinib22 figure 4figure 4 clustering mutations egfr gene critical sit
es within atp binding pocket postulate mutations result repositioning critical residues st
abilizing interaction atp competitive inhibitor gefitinib mechanism would explain increase
d receptor activation ligand binding enhanced inhibition induced gefitinib structural anal
ysis mutant receptors therefore provide important insight mechanisms regulate activation e
gfr design potent inhibitors targeting mutant receptors observations implications identifi
cation molecular targets cancer therapy using small molecule kinase inhibitors effectivene
ss imatinib cml based ability target abl tyrosine kinase activated bcr abl translocation e
philadelphia chromosome patients disease transform hematopoietic cells 2 23 similar eviden
ce designating protein optimal therapeutic target available epithelial cancers data sugges
t egfr tyrosine kinase mutations used identify subgroup patients non small cell lung cance
r growth factor receptor may essential tumor growth whereas overexpression egfr absence mu
tations may reflect less critical role played factor majority cases emphasis genetic alter
ations consistent observation amplification her2 neu gene reliable predictor protein overe
xpression responsiveness breast cancer targeting antibody trastuzumab c kit mutations used
determine response gastrointestinal stromal tumors imatinib 24 25 ongoing large scale sequ
encing efforts may reveal additional mutations kinases linking different cancers potential
therapeutic targets 26 27 gefitinib elicited clinical responses patients gliomas despite h
igh frequency amplification rearrangements egfr gene patients 15 16 however egfr tyrosine
kinase mutations patients non small cell lung cancer fundamentally different glioma associ
ated deletions within extracellular domain egfr truncated egfr proteins resemble avian ery
throblastosis viral oncogene v erbb mediating constitutive ligand independent activation r
eceptor alter atp cleft tyrosine kinase bound gefitinib enhanced sensitivity gefitinib ass
ociated tyrosine kinase mutations may therefore contribute substantially clinical response
s certain patients non small cell lung cancer plasma concentrations gefitinib achieved use
current dosage recommendations21 exceed drug concentration suppressed autophosphorylation
mutant egfr tyrosine kinase assays required suppress wild type receptor vitro analysis wil
d type egfr also suggested low concentrations gefitinib may sufficient suppress autophosph
orylation tyrosine residues abrogation downstream signaling requires higher dose 28 thus p
atients gliomas biologic dependence egfr signaling identified presence gene amplification
deletions within extracellular domain clinical response may require plasma concentrations
egfr tyrosine kinase inhibitor sufficient abrogate downstream signaling understanding mole
cular basis responsiveness gefitinib immediate clinical implications respect patients non
small cell lung cancer clustering mutations within specific regions egfr tyrosine kinase d
omain makes possible potential development rapid reliable diagnostic testing guide clinica
l use gefitinib patients whose tumors activating mutations egfr dramatic responses gefitin
ib patients whose disease refractory therapies suggest agent may effective used earlier co
urse treatment prospective validation egfr tyrosine kinase mutations predictors responsive

ness gefitinib warranted genotype directed clinical trials tyrosine kinase inhibitor initi
al treatment advanced non small cell lung cancer even adjuvant setting surgical resection
considered somatic mutations tyrosine kinase tk domain epidermal growth factor receptor eg
fr gene reportedly associated sensitivity lung cancers gefitinib iressa kinase inhibitor f
rame deletions occur exon 19 whereas point mutations occur frequently codon 858 exon 21 fo
und sequencing egfr tk domain 7 10 gefitinib sensitive tumors similar types alterations mu
tations found eight gefitinib refractory tumors p 0 004 five seven tumors sensitive erloti
nib tarceva related kinase inhibitor clinically relevant target undocumented analogous som
atic mutations opposed none 10 erlotinib refractory tumors p 0 003 mutation positive tumor
s adenocarcinomas patients smoked 100 cigarettes lifetime never smokers screened egfr exon
s 2 28 15 adenocarcinomas resected untreated never smokers seven tumors tk domain mutation
s contrast 4 81 non small cell lung cancers resected untreated former current smokers p 0
0001 immunoblotting lysates cells transiently transfected various egfr constructs demonstr
ated compared wild type protein exon 19 deletion mutant induced diminished levels phosphot
yrosine whereas phosphorylation tyrosine 1092 exon 21 point mutant inhibited 10 fold lower
concentrations drug collectively data show adenocarcinomas never smokers comprise distinct
subset lung cancers frequently containing mutations within tk domain egfr associated gefit
inib erlotinib sensitivity tyrosine kinases tks regulate signaling pathways control critic
al cellular activities 1 overexpressed activated mutations tks contribute development canc
ers tumor cells depend mutant tk survival illustrated certain mouse models cancer 2 3 muta
ted enzyme fortuitously serve achilles heel cancer therapy 4 human examples include bcr ab
l dependent chronic myelogenous acute lymphoblastic leukemias 5 kit pdgfra dependent gastr
ointestinal stromal tumors 6 pdgfra dependent hypereosinophilic syndrome 7 disease activat
ed oncogenes encode tks inhibition imatinib mesylate gleevec leads rapid durable clinical
responses egfr tk erbb family presumptive target tk inhibitor tki gefitinib drug anilinoqu
inazoline fig 5 published supporting information pnas web site reversibly competes atp cri
tical atp binding site lysine 745 k745 within epidermal growth factor egf receptor egfr pr
otein 8 9 vitro gefitinib selectively inhibits kinase activity egfr versus handful kinases
10 two phase ii trials radiographic regressions tumors observed 28 patients treated japan
10 studied europe u 11 12 dramatic responses occurred within first two weeks initiating th
erapy e g ref 13 similar seen murine human examples noted assuming drug affect kinase kind
s responses suggested least lung tumors depended specific genetic lesion tumor survival ho
wever gefitinib approved second third line treatment patients non small cell lung cancer n
sclc clinically relevant target drug human tumors unknown analyses preclinical xenograft m
odels 14 specimens gefitinib sensitive refractory tumors 15 reveal obvious relationship eg
fr expression levels tumor sensitivity retrospective epidemiologic analyses suggested gefi
tinib likely effective japanese patients 11 individuals adenocarcinomas bronchioloalveolar
carcinoma bac subtype never smokers 16 recently two groups shown mutations tk domain egfr
associated sensitivity nsclc gefitinib 17 18 total deletions amino acid substitutions exon
s 18 19 21 egfr found 13 14 tumors sensitive drug none 11 tumors response lynch colleagues
17 found mutations another 2 25 primary nsclcs paez et al 18 found egfr mutations 16 119 u
nselected tumors striking predominance mutations found 15 58 28 specimens japan compared 1
61 u 2 confirm extend data gefitinib sensitivity examined status tk domain egfr tumors sen
sitive refractory drug determine whether related distinct tki erlotinib fig 5 targets simi
lar subset nsclcs also profiled erlotinib sensitive refractory tumors clinically relevant
target erlotinib yet documented examine whether smoking history predictive likelihood egfr
mutations determined incidence egfr tk domain mutations 96 resected nsclcs never smokers w
ell former current smokers never received tki finally effort explain selective advantage c
ells mutant egfr drug sensitivity conferred upon mutant bearing tumors began characterize
biochemical properties egfr mutants vitro go methods tissue procurement tumor specimens ob
tained protocols approved institutional review board human tissue utilization committee me
morial sloan kettering cancer center paraffin blocks tumor material obtained patients syst
emic treatment lung cancer collected retrospectively patients gefitinib n 18 prospectively
patients erlotinib n 17 frozen tumor specimens untreated patients stage iiia nsclc n 96 pr
ospectively collected time surgical resection supporting text table 3 published supporting
information pnas web site mutational analyses egfr lung tumors genomic dna derived either
tumors embedded paraffin blocks fresh frozen tumors see supporting text tables 4 5 publish
ed supporting information pnas web site pcr primers sequencing reactions performed forward
reverse directions mutations confirmed pcr amplification independent dna isolate drug sens
itive tumors mutations sequences exons 19 21 also determined least two independently deriv
ed pcr products fisher exact test used calculate p values functional analyses mutant egfrs
egfr two numbering systems first denotes initiating methionine signal sequence amino acid
24 second used denotes methionine amino acid 1 commercial antibodies y1068 specific anti p
hospho egfr use first nomenclature consistent consider y1068 y1092 mutations introduced fu
ll length egfr using quikchange site directed mutagenesis kit stratagene see supporting te
xt mutant clones fully resequenced ensure additional mutations introduced cdnas cloned pcd

na3 1 expression vectors invitrogen transient transfections 293t human embryonic kidney ce lls transfected 2 105 cells per well six well plates using fugene roche applied science 0 8 g plasmid dna cells grown dmem high glucose 10 fcs 2 mm l glutamine 10 units ml penicill in 10 g ml streptomycin 37 c 5 co2 cells serum starved media containing 0 1 serum egf cell signaling technology beverly used 100 ng ml cells treated various concentrations gefitinib erlotinib provided astrazeneca genentech respectively least three independent experiments performed analyses immunoblotting see supporting text details cell lysis immunoblotting sp ecific proteins detected using enhanced chemiluminescence amersham pharmacia following ant ibodies horseradish peroxidase hrp conjugated anti phosphotyrosine rc20 1 3 333 anti total egfr 1 2 500 bd transduction laboratories anti actin 1 2 000 sigma anti phospho egfr tyr 1 068 y1092 1 1 000 cell signaling technology hrp conjugated anti rabbit ig 1 5 000 amersham pharmacia hrp conjugated anti mouse igg 1 2 000 roche applied science blots stripped resto re stripping buffer pierce 37 c 15 min stripping verified reexposure film reprobing densit ometry performed using imagequant v1 2 molecular dynamics least three independent experime nts performed analyses go results mutations egfr commonly found lung tumors sensitive gefi tinib ascertain whether mutations within tk domain egfr associated sensitivity tki gefitin ib 17 18 performed mutational profiling exons 18 24 egfr tumors 10 patients demonstrated p artial response marked clinical improvement defined supporting text treated drug single ag ent memorial sloan kettering cancer center patients selected retrospectively seven tumors 70 mutations six frame nucleotide deletions exon 19 occur adjacent k745 encoded nucleotide s 2233 2235 k745 shown critical binding atp 19 seventh patient nonsynonymous mutation nucl eotide 2573 g exon 21 resulting substitution arginine leucine position 858 l858r change oc curs adjacent highly conserved dfg motif amino acids 855 857 activation loop kinase 20 pre viously detected nsclcs 17 18 table 1 fig 6 published supporting information pnas web site mutations found dna peripheral blood available four patients g2 g3 g5 g6 implying lesions arose somatic cells among seven tumors mutations five arose never smokers six adenocarcino ma histology features bac none patients east asian origin four exon 19 deletion mutations previously described del e746 a750 17 18 two previously described del l747 s752 del e746 t 751insi del e746 t751insi mutation may resulted double deletion nucleotides 2229 2236 2245 2252 insertion single nucleotide fig 1 b remarkably mutant despite elimination codon k745 remaining nucleotide sequence reencoded lysine eliminated amino acids lrea exon 19 deletio ns lrea motif completely conserved among egfrs vertebrates fig 1c fig 1 fig 1 deletion mut ations exon 19 egfr nsclcs sensitive tkis gefitinib g erlotinib e lack four amino acids lr ea conserved among vertebrate species nucleotide alignments b amino acid alignments c amin o acid alignments somatic egfr mutations nsclcs reported heterozygous however analysis two seven tumors g1 g3 wild type sequence detected region encompassing deletion result implies mutations tumors patients g1 g3 hemi homozygous ii mutant gene selectively amplified tumor s iii mutations general may homozygous wild type sequence originating contaminating normal dna table 1 fig 6a see discussion table 1 table 1 characteristics patients sensitive gefit inib g erlotinib e none eight gefitinib refractory tumors contained mutations within exons 18 24 egfr p 0 004 thus egfr mutations appear associated sensitivity nsclcs gefitinib lung tumors sensitive erlotinib also harbor mutations egfr erlotinib like gefitinib atp competi tive inhibitor egfr tk fig 5 erlotinib also appears particularly effective lung cancers ba c histology patients never smokers drug clinically relevant target tumors yet documented s ee whether lung tumors sensitive erlotinib also contained mutations within tk domain egfr analyzed exons 18 24 seven tumors patients demonstrated partial response phase ii trial ag ent bac five tumors contained mutations two multinucleotide frame deletions within exon 19 near atp binding site k745 similar found gefitinib sensitive tumors deletions eliminated a mino acids lrea amino acids 747 750 fig 1 b one mutation del l747 s752insq previously repo rted three tumors contained l858r mutation nucleotide 2573 also found gefitinib responsive tumors one specimen also second mutation previously unreported c change nucleotide 2326 ex on 20 substituting cysteine arginine position 776 r776c carboxy terminal p loop fig 6e fou r five mutation positive tumors obtained never smokers none patients east asian origin mat ched normal tissue available four patients e1 e3 e5 showed wild type sequence indicating m utations arose somatic cells mutations observed exons 18 24 tumors 10 patients trial parti al response p 0 003 thus similar results gefitinib mutations egfr tk domain also associate d sensitivity nsclcs erlotinib egfr mutations commonly found lung adenocarcinomas never sm okers nine 12 75 mutation positive tumors study adenocarcinoma histology derived never smo kers table 1 moreover 10 cases lung cancer arise patients history tobacco use adenocarcino ma histology 21 22 determine prospective manner frequency egfr mutations lung adenocarcino mas never smokers sequenced exons 2 28 egfr 15 tumor specimens fit clinical criteria sampl es selected prospectively collected tissue bank surgically resected nsclcs derived patient s stage ii iiia disease time surgery tissue banking none patients primary tumor resected r eceived treatment lung cancer including tkis enriched population tumors never smokers 7 15 47 mutations tk domain egfr one tumor specimen 230 deletion exon 19 del e746 a750 five spe cimens 3 24 25 42 166 l858r amino acid substitution exon 21 table 2 seventh tumor specimen

77 contained previously unreported point mutation exon 21 nucleotide 2504 resulting substi
tution leucine histidine position 835 h835l fig 6f mutation predicted lie adjacent activat
ion loop egfr tk domain contrast l858r mutation loop mutations found adjacent normal appea
ring tissue seven patients table 2 table 2 somatic mutations tk domain egfr common surgica
lly resected nsclcs derived never smokers infrequent former current smokers also sequenced
exons 2 28 egfr additional 81 primary nsclcs randomly selected tumor bank tumors cohort de
rived former current smokers 24 tumors squamous cell histology four 81 5 mutations egfr tk
domain previously observed l858r amino acid substitution within exon 21 table 2 correspond
ing normal tissue available three four tumors specimens 5 65 134 wild type sequence intere
stingly among four tumors l858r mutation three specimens 65 98 134 arose patients limited
exposure cigarette smoking three smoked 1 pack per day 9 years 9 pack years quit least 30
years surgery specimen 5 resected individual 14 pack year history quit 1 month surgery tak
en together data demonstrate egfr mutations commonly found nsclcs never smokers opposed fo
rmer current smokers 7 15 vs 4 81 p 0 0001 tumors likely mutation positive identified usin
g specific clinical characteristics biochemical properties egfr mutants gain insight cells
containing mutant egfrs selected growth certain nsclcs mutations confer susceptibility tki
s begun characterize mutant proteins cultured cells wild type del l747 s752 l858r egfr pro
duced transient transfection expression vectors 293t cells low levels endogenous egfr fig
2 expression total egfr egfr assessed immunoblotting using anti egfr monoclonal antibody a
ctin served indicator relative levels protein per sample size mutant egfrs virtually indis
tinguishable wild type egfr assessed immunoblotting interestingly amount egfr relative act
in average 3 fold higher del l747 s752 protein wild type egfr n 5 differences egfr likely
caused varying transfection efficiencies equal numbers cells used separate transfection le
vels egfr another egfr mutant l858r comparable wild type fig 2c fig 2 fig 2 del l747 s752
mutant egfr appears reduced kinase activity 293t cells transiently transfected vector alon
e v vector containing wild type wt egfr del l747 s752 l858r thirty six hours later cells s
erum starved next used immunoblotting extracts cells expressing various egfrs assess vario
us aspects protein activity drug sensitivity surrogate gauge kinase activity measured leve
ls autophosphorylated tyr 1092 egfr y1092 site binding adaptor molecules grb2 leads activa
tion mitogen activated protein kinase extracellular signal related kinase cascade using y1
092 specific antibody e phospho egfr p egfr relation levels egfr protein also assessed pat
tern levels induced tyrosine phosphorylation cell proteins using anti phosphotyrosine anti
body rc 20 absence serum egf extracts cells transfected wild type egfr demonstrated averag
e 16 fold greater ability autophosphorylate y1092 del l747 s752 mutant even addition egf n
5 fig 2 c consistent results del l747 s752 mutant also induced markedly low levels tyrosin
e phosphorylated proteins compared wild type egfr fig 2 b c longer exposure demonstrated r
elative intensities proteins appeared qualitatively different wild type well fig 2b right
contrast deletion mutant results y1092 specific antibody l858r mutant similar observed wil
d type egfr figs figs 2c2c and3b 3b however pattern phosphotyrosine staining cell proteins
still distinct fig 2c fig 3 fig 3 compared wild type egfr del l747 s752 mutant similar sen
sitivity tkis whereas l858r mutant inhibited 10 fold lower concentrations drug dose depend
ent inhibition gefitinib del l747 s752 mutant egfr compared finally assessed sensitivity d
el l747 s752 l858r egfr mutants tkis measuring ratio p egfr egfr lysates transiently trans
fected cells serum starved pretreated gefitinib erlotinib wild type egfr del l747 s752 mut
ant appeared approximately sensitivity gefitinib fig 3a erlotinib data shown contrast l858
r mutant 10 fold greater sensitivity gefitinib data shown erlotinib fig 3b fig 7 published
supporting information pnas web site go discussion study confirm extend recent work associ
ating egfr mutations sensitivity tki gefitinib 17 18 furthermore establish tumors sensitiv
e related kinase inhibitor erlotinib contain similar types egfr mutations data two publish
ed reports study used 25 31 81 tumors individuals experiencing partial responses marked cl
inical improvement taking gefitinib erlotinib contain mutations egfr tk domain contrast no
ne 29 specimens patients refractory agents mutations p 10 10 findings demonstrate mutation
s tk domain egfr associated sensitivity two drugs whether gefitinib erlotinib target exact
ly overlapping sets nsclc patients whether distinct mutations confer greater sensitivity s
pecific egfr tk inhibitors yet determined number sensitive tumors analyzed still small als
o demonstrate 11 96 12 primary nsclcs resected untreated patients contain mutations egfr w
ithin tk domain none tumors derived patients east asian origin taken together published li
terature egfr mutations primary lung cancers patients u 14 182 tumors 8 positive egfr muta
tions data could account responses seen phase ii trials gefitinib 10 european american pat
ients experienced radiographic regressions 12 however mechanisms drug sensitivity may also
apply remarkably selecting tumors certain clinical characteristics predictive response tki
s e tumors never smokers adenocarcinoma histology enriched percentage patients mutations t
hus 7 15 tumors never smoking patients adenocarcinoma histology egfr mutations whereas 4 8
1 nsclcs former current smokers contained moreover three four patients latter cohort relat
ively short smoking histories data show lung tumors patients minimal direct exposure cigar
ettes adenocarcinoma histology usually features bac distinct molecular phenotype distingui

shes remainder nsclcs patients mutation positive nsclc various stages disease treated unkn
own clearly warrants prompt investigation time critical mutations kinases sought nsclcs wi
ld type egfr never former current smokers among nsclc associated egfr mutations reported d
ate 49 56 88 occur two hotspots fig 4 total 29 49 59 multinucleotide frame deletions elimi
nate four amino acids lrea exon 19 20 49 hotspot mutations 41 point mutations exon 21 resu
lt specific amino acid substitution position 858 l858r remaining 7 56 mutations 12 nucleot
ide substitutions found exons 18 21 outside common sites mutation including previously unr
eported r776c h835l mutations described nearly cases one mutation detected per tumor howev
er study find one tumor sample two mutations patient e3 r776c l858r significance r776c mut
ation whether two mutations chromosome different chromosomes cells different subclones tum
or specimen unknown fig 4 fig 4 summary mutations reported previously detected tk domain e
gfr nsclcs schematic view egfr key domains expanded view tk domain encoded exons 18 24 ami
no acids 718 964 yellow sensitive gefitinib nsclc associated egfr mutations frequently het
erozygous however paez et al 18 reported one mutation involving exon 19 appeared homozygou
s detected two cases interpretation mutational status solely dna sequencing problematic on
e hand contaminating normal cells wild type egfr could account apparent heterozygosity han
d amplification mutant egfr occurs lung cancer 23 could account detection mutant sequences
mouse models expressing mutant egfr proteins lung analysis mutant positive nsclcs fluoresc
ence situ hybridization array based comparative genomic hybridization may help address iss
ues interestingly one tumors g3 detected heterozygous intronic polymorphism downstream exo
n 19 data shown case probable gene conversion event occurred encompassing area deletion ex
on 19 five 17 reported patients experienced partial responses marked clinical improvement
therapy tkis wild type sequence exons 18 24 series reported lynch et al 17 one nine gefiti
nib sensitive patients mutations within entire coding region egfr explanations results inc
lude tumor fragments analyzed represent tumors assessed patients drug studies ii mutations
present detection rate sequencing assays iii mutations lay outside exons encoding tk domai
n iv mechanisms involving wild type egfr e g amplification may confer drug sensitivity reg
ards third point mutations outside tk domain unlikely mutations egfr found 240 lung tumor
specimens sequenced date 119 tumors paez et al 18 25 tumors lynch et al 17 96 resected nsc
lcs study thus important determine whether wild type egfr kinases play role tumor response
s gefitinib erlotinib mutant egfrs selected confer susceptibility tkis questions require i
nvestigation gain insight begun study various egfrs wild type exon 19 deletion del l747 s7
52 exon 21 aa substitution l858r cultured cells low levels endogenous egfr using transient
transfection assays parameters activity ability phosphorylate tyrosine 1092 egfr phosphory
late tyrosine residues cell proteins general compared wild type egfr relative intensities
tyrosine phosphorylated proteins induced cells transfection either egfr mutant qualitative
ly quantitatively different level phosphotyrosine especially diminished del l747 s752 muta
nt result also obtained two cell lines e cos1 hpl1d immortalized human peripheral lung epi
thelia ref 24 293t cells using another deletion mutant del e746 a750 data shown furthermor
e contrast published data 17 found two egfr mutants differential sensitivities tkis measur
ed effect drug phosphorylation egfr y1092 cells transiently transfected various egfrs wher
eas l858r mutant inhibited 10 fold lower concentrations tki del l747 s752 mutant appeared
similar sensitivities wild type egfr drug discrepancies published reports could due differ
ent experimental conditions however data support notion egfr mutants enhanced activity sug
gested 17 rather results suggest del l747 s752 l858r mutants could altered substrate speci
ficity compared wild type protein interestingly mutant egfrs critical k745 residue changed
methionine arginine also reduced kinase activity still activate mitogen activated protein
kinase cascade incomplete program cellular tyrosine phosphorylations signaling postulated
occur via heterodimerization erbb family members erbb2 neu 25 27 biochemical properties de
l l747 s752 mutant evaluated reminiscent certain b raf mutants found human cancers 28 alth
ough majority b raf mutants elevated kinase activity 3 22 reported b raf mutants found red
uced kinase activity toward mek vitro nevertheless three mutants found signal erk cells ac
tivating c raf possibly via allosteric transphosphorylation mechanism tumors sensitive eit
her gefitinib erlotinib eventually progress despite continued treatment tkis patients bcr
abl dependent chronic myelogenous leukemia mutations within amplification bcr abl lead cli
nical resistance 29 whether resistance egfr tkis caused similar mechanisms affecting egfr
mechanisms affecting alternative molecules remains determined twenty one cases primary lun
g carcinoma analyzed correlations presence somatic mutations epidermal growth factor recep
tor egfr gene phosphorylation status egfr analyzed immunohistochemistry antibodies recogni
zing phosphorylated form egfr somatic mutations detected 11 52 4 21 cases immunohistochemi
stry antibody recognizing egfr phosphorylated tyrosine pegfr tyr 992 antibody recognizing
egfr phosphorylated tyrosine 1173 pegfr tyr1173 revealed 12 57 1 21 100 21 cases positive
respectively interestingly mutation status egfr gene strongly correlated immunoreactivity
pegfr tyr992 p 0019 pegfr tyr992 immunoreactivity significantly correlated clinical respon
siveness gefitinib p 0011 findings suggest immunohistochemical evaluation anti pegfr tyr99
2 antibody useful prediction responsiveness gefitinib keywords epidermal growth factor rec

eptor gefitinib immunohistochemistry phosphorylation predictive marker 1 introduction lung cancer leading cause cancer related death accounting 1 18 million deaths annually worldwid e 1 overall 5 year survival rate diagnosis lung cancer 15 2 may largely due practical diff iculties diagnosing lung cancer early stage addition even early stage tumors already acqui red potential invasion systemic metastasis therefore advances multidisciplinary therapy es pecially chemotherapy essential order improve survival patients lung cancer gefitinib oral ly active molecular targeted drug inhibits epidermal growth factor receptor egfr dependent growth cancer cells patients non small cell lung carcinoma nsclc known competitive inhibit or adenosine triphosphate binding cleft within tyrosine kinase domain egfr 3 although gefi tinib initially considered effective less toxic conventional cytotoxic agents recent phase iii clinical trials 4 5 6 failed demonstrate effectiveness patients nsclc overall furtherm ore severe adverse events gefitinib interstitial lung disease reported 7 administration be come restricted however patients certain personal histologic characteristics asian female nonsmoker diagnosed adenocarcinoma reportedly responsive gefitinib 8 9 10 11 furthermore r ecent reports stressed patients somatic mutations egfr gene exhibited good responsiveness gefitinib 12 13 14 15 16 17 18 mitsudomi et al 15 showed gefitinib effective 83 patients e gfr mutations whereas effective 10 patients without egfr mutations p 0001 similarly taron et al 18 showed gefitinib effective 94 1 patients egfr mutations compared 12 6 patients wi thout egfr mutations p 0001 findings strongly suggest possible select patients would respo nsive gefitinib according whether egfr gene mutation study analyzed 20 cases primary pulmo nary adenocarcinoma 1 case adenosquamous carcinoma phosphorylation status egfr protein mut ation status egfr gene immunohistochemistry antibodies specifically recognizing phosphoryl ated form egfr revealed positive immunoreactivity egfr phosphorylated tyrosine pegfr tyr 9 92 significantly correlated presence mutation egfr gene suggesting immunohistochemistry an ti pegfr tyr992 antibody may convenient tool predicting presence egfr gene mutation 2 mate rials methods 2 1 patients tissues tissue samples obtained 21 japanese patients underwent surgical resection primary lung carcinoma oita university hospital oita prefectual hospita l specimens routinely fixed 10 formalin embedded paraffin serial sections cut paraffin emb edded blocks used routine histopathology mutation analysis egfr gene immunohistochemistry clinical pathologic data patients shown table 1 median age 21 patients 71 years ranging 45 78 years male female ratio 11 10 nine patients current former smokers pathologically 8 2 1 0 1 21 cases categorized stage ii iii iv respectively total 21 cases 20 histopathologicall y diagnosed adenocarcinoma remaining 1 case diagnosed adenosquamous carcinoma furthermore well known histopathologic subtypes frequently admixed one tumor histologic subtype determ ined microscopically observing largest cut surface tumor table 1 dominant subtypes 21 case s 3 cases solid adenocarcinoma 16 cases papillary adenocarcinoma 1 case bronchioloalveolar carcinoma 1 case acinar adenocarcinoma table 1 patient characteristics clinical histologic analysis age sex smoking stage histology histologic subtype ps recurrent site response gef itinib lung others 1 75 f pt2n0m0 ib adsq solid acinar 2 pr 2 76 f pt1n0m0 ia ad papillary acinar 1 pr 3 63 f pt2n1m0 iib ad papillary acinar bac 1 pr 4 71 f pt1n2m0 iiia ad papilla ry acinar 0 pr 5 64 pt4n2m0 iiib ad papillary acinar 0 pr 6 77 f pt2n0m0 ib ad papillary b ac acinar 0 pr 7 72 f pt1n0m0 ia ad papillary acinar bac 0 pr 8 64 pt2n0m0 ib ad papillary acinar 0 pr 9 59 pt4n2m0 iiib ad papillary bac acinar 2 pr 10 76 pt2n0m0 ib ad bac acinar 1 sd 11 45 f txn3m0 iiib ad papillary solid 1 sd 12 75 f pt3n2m0 iiia ad papillary acinar bac 2 sd 13 47 pt1n2m0 iiia ad acinar papillary 1 sd 14 77 pt1n0m0 ia ad papillary 2 sd 15 49 f pt4n2m0 iiib ad papillary 0 sd 16 77 pt1n0m0 ia ad papillary 3 pd 17 71 pt3n0m0 iib a d papillary acinar 2 pd 18 51 pt4n2m0 iiia ad solid acinar 1 pd 19 76 pt4n2m0 iiib ad soli d acinar 2 pd 20 78 pt2n2m1 vi ad papillary acinar 1 pd 21 51 f pt1n2m0 iiia ad papillary acinar 1 pd abbreviations f female male adsq adenosquamous carcinoma ad adenocarcinoma bac bronchioloalveolar carcinoma ps performance status table options primary surgical operatio n 21 patients suffered disease relapse treated 250 mg gefitinib daily recurrent tumor loca lized remnant lungs 7 patients remaining 14 cases tumor found disseminated organs brain li ver bone clinical response gefitinib determined basis response evaluation criteria solid t umors 19 numbers cases diagnosed showing partial response pr stable disease sd progressive disease pd 9 42 8 6 28 6 6 28 6 respectively present study pr cases categorized responders sd pd cases nonresponders present study approved ethics committees oita university hospita l approval number p 05 06 oita prefectural hospital approval number 18 4 2 2 extraction ge nomic dna paraffin embedded carcinoma tissues collect carcinoma cells tissue sections seri al sections 15 thick prepared one stained hematoxylin eosin observing carcinoma tissues li ght microscopy directly cut carcinoma cells serial sections using sterile needles collecte d carcinoma tissues deparaffined xylene rinsed 100 vol vol ethanol incubated 1 te buffer 1 25 g ml proteinase k wako osaka japan 0 5 sodium dodecyl sulfate wako 55 c overnight subse quently samples treated rnase wako 10 g ml 1 hour 37 c finally genomic dna obtained routin e phenol chloroform extraction followed ethanol precipitation 2 3 mutation analysis egfr g ene polymerase chain reaction pcr single strand conformation polymorphism analysis perform ed 4 exons exons 18 21 encoding tyrosine kinase domain egfr subsequently presence mutation

s confirmed sequencing analysis pcr product used primers used previous study 20 exon 18 fo
rward 5 ctctgtgttcttgtcccccc 3 reverse 5 gcctgtgccagggaccttac 3 exon 19 forward 5 cgtcttcc
ttctctctctgt 3 reverse 5 ccacacagcaaagcagaaac 3 exon 20 1 forward 5 atgcgaagccacactgacgt 3
reverse 5 aagggcatgagctgcgtgat 3 exon 20 2 forward 5 atcacgcagctcatgcccttt 3 reverse 5 cgta
tctcccttcccctgatt 3 exon 21 forward 5 ccaggaacgtactggtgaaa 3 reverse 5 tgacctaaagccacctcctt
3 pcr reaction performed using 50 ng genomic dna extracted carcinoma cells 0 625 u blend t
aq toyobo osaka japan 1 buffer blend taq toyobo 0 2 mmol l dntps 0 2 mol l forward reverse
primers total reaction volume 25 l pcr conditions follows 1 cycle 94 c 3 minutes 35 cycles
94 c 30 seconds 55 c 30 seconds 72 c 1 minute followed reaction 72 c 5 minutes pcr product
s loaded onto genegel excel 12 5 24 genephor system ge healthcare bioscience uppsala swede
n 15 c following manufacturer protocol electrophoresis gels stained dna silver staining ki
t ge healthcare bioscience shifted bands collected dna fragments recovered gels subcloned
sequenced using abi prism 310 genetic analyzer applied biosystems foster city ca bigdye te
rminator cycle sequence ready reaction kit applied biosystems 2 4 antibodies immunohistoch
emistry primary rabbit antibodies egfr pegfr tyr845 992 1045 1068 1173 purchased cell sign
aling technology danvers immunohistochemistry performed according method described previou
sly 21 briefly deparaffinized rehydrated sections 3 thick autoclaved 0 01 mol l citrate bu
ffer ph 6 0 10 minutes 120 c antigen retrieval blocking 3 hydrogen peroxide 15 minutes roo
m temperature rt inactivate endogenous peroxidase activity sections incubated 10 normal go
at serum 30 minutes rt block nonspecific binding secondary goat antirabbit antibody sectio
ns incubated primary antibodies diluted 1 100 diluting solution dako north america carpint
eria ca overnight 4 c next sections washed 1 phosphate buffered saline pbs incubated bioti
nylated goat antirabbit immunoglobulin g nichirei tokyo japan 30 minutes rt washed 1 pbs s
ections incubated solution avidin conjugated horseradish peroxidase using vectastain elite
abc kit vector laboratories burlingame ca 20 minutes rt accordance manufacturer recommenda
tions finally washing 1 pbs signals visualized incubation h2o2 diaminobenzidine substrate
solution sections counterstained hematoxylin dehydration mounting according criteria descr
ibed previously du et al 22 evaluation immunohistochemical data performed 2 independent pa
thologists blinded egfr gene mutation status sensitivity gefitinib staining intensity judg
ed 0 stain 1 weak 2 moderate less bronchiolar cells positive internal control 3 strong int
ense positive internal control cells populations positively stained cells categorized 4 gr
oups basis percentage positive tumor cells follows 0 0 positive cells 1 1 9 positive cells
2 10 50 positive cells 3 50 positive cells scoring performed 3 times per case 3 distinct f
ields 3 scores averaged averaged scores intensity population summed summed scores 3 catego
rized positive study 2 5 statistical analysis statistical analyses study carried statview
statistical software package abacus concepts berkley ca fisher exact probability test used
evaluating correlations immunohistochemical clinical data differences p 05 considered stat
istically significant 3 results 3 1 correlations clinicopathologic status sensitivity gefi
tinib first examined correlations clinicopathologic factors patients responsiveness gefiti
nib shown table 2 smoking history exhibited statistically significant correlation responsi
veness gefitinib p 0011 however correlations observed factors including age sex pathologic
al stage dominant histological subtype performance status recurrence site table 2 comparis
on clinicopathologic factors responsiveness gefitinib responder pr n 9 nonresponder sd pd
n 12 p age 1 0000 71 5 7 71 4 5 sex 1984 male 3 8 female 6 4 smoking history 0011 current
former 0 9 never 9 3 stage 1984 ia iib 6 4 iiia iv 3 8 dominant histologic subtype 3383 pa
pillary 8 8 nonpapillary 1 4 ps 6424 2 2 5 2 7 7 recurrent site 3972 lung 4 3 others 5 9 t
able options 3 2 significant correlation egfr gene mutations sensitivity gefitinib next ex
amined correlation egfr gene mutation status patients sensitivity gefitinib mutations pulm
onary adenocarcinomas known confined region exons 18 21 egfr gene 12 13 14 23 24 25 26 27
28 performed pcr single strand conformation polymorphism analysis followed sequencing regi
on described materials methods shown table 3 somatic mutations egfr gene detected 11 52 4
21 patients 6 11 patients frame deletions exon 19 detected another 4 11 patients point mut
ation nucleotide 2573 g l858r exon 21 detected remaining patient 2 point mutations nucleot
ide 2126 c e709a nucleotide 2155 g g719c exon 18 detected mutations exon 20 detected patie
nts thus consistent previous reports 12 13 14 23 24 25 26 27 28 10 90 9 11 patients analyz
ed mutations restricted exon 19 21 respect clinical responsiveness gefitinib 9 6 6 21 pati
ents treated gefitinib diagnosed clinically showing pr sd pd respectively according respon
se evaluation criteria solid tumors criteria shown table 4 9 11 patients egfr mutations di
agnosed responders 2 nonresponders conversely 9 patients diagnosed responders found mutati
ons furthermore 10 patients showed mutations diagnosed nonresponders statistical analysis
results revealed presence egfr mutations significantly correlated responsiveness gefitinib
p 0002 consistent results recent reports 12 13 14 15 16 17 18 table 3 patient characterist
ics egfr mutation immunohistochemical analysis response gefitinib egfr mutation immunohist
ochemistry egfr pegfr tyr992 pegfr tyr1173 1 pr ex19 del e746 a750 2 pr ex21 l858r 3 pr ex
19 del a750 e758 ins p 4 pr ex19 del e746 a750 5 pr ex19 del e746 a750 6 pr ex19 del e746
t751 ins l 7 pr ex18 e709a g719c 8 pr ex21 l858r 9 pr ex21 l858r 10 sd 11 sd 12 sd ex21 l8

58r 13 sd ex19 del e746 a750 14 sd 15 sd 16 pd 17 pd 18 pd 19 pd 20 pd 21 pd abbreviations ex exon del deletion ins insertion mutations negative positive table options table 4 compa rison responsiveness gefitinib egfr genotype egfr expression egfr phosphorylation status t yrosine 1173 992 responder pr n 9 nonresponder sd pd n 12 p egfr genotype 0002 wild 0 10 m utated 9 2 egfr 1 0000 negative 0 0 positive 9 12 pegfr tyr1173 1 0000 negative 0 0 positi ve 9 12 pegfr tyr992 0 0011 negative 0 9 positive 9 3 table options 3 3 lack correlation e gfr expression responsiveness gefitinib analyzed expression egfr 21 patients immunohistoch emistry shown table 3 egfr expression detectable cases expression levels uniform fig 1 sho wing egfr constitutively expressed pulmonary adenocarcinomas therefore suggested overexpre ssion egfr may sufficient favorable responsiveness gefitinib immunohistochemistry pulmonar y adenocarcinoma anti egfr antibody fig 1 immunohistochemistry pulmonary adenocarcinoma an ti egfr antibody anti pegfr tyr1173 antibody anti pegfr tyr992 antibody immunohistochemist ry representative case pr case 7 indicated table 1 table 3 g j representative case sd case 10 indicated table 1 table 3 b e h k representative case pd case 17 indicated table 1 tabl e 3 c f l hematoxylin eosin staining b c immunohistochemistry anti egfr antibody e f anti pegfr tyr1173 antibody g h anti pegfr tyr992 antibody j k l performed although carcinoma c ells pr case positively immunostained anti pegfr tyr992 antibody j sd pd cases immunostain ed k l figure options 3 4 correlation egfr phosphorylation status tyrosine 992 responsiven ess gefitinib reported phosphorylation status tyrosine residues cytoplasmic domain egfr al tered cultured cells harboring mutations egfr gene 12 29 30 therefore next analyzed phosph orylation status egfr immunohistochemistry using specific antibodies phosphorylated egfr f irst investigated phosphorylation status egfr tyrosines 845 992 1045 1068 1173 immunohisto chemistry using commercially available antibodies see materials methods paraffin embedded tissue sections however unable detect phosphorylated egfr tyrosine 845 1045 1068 data show n however western blot analysis antiphosphorylated egfr antibodies already shown residues phosphorylated cell lines established pulmonary adenocarcinomas harboring mutations egfr g ene 12 29 30 therefore judged antibodies phosphorylated tyrosine residues 845 1045 1068 us ed present study suitable immunohistochemistry using paraffin embedded tissue sections han d anti pegfr tyr992 antibody anti pegfr tyr1173 antibody found immunoreactive using paraff in embedded tissue sections therefore performed following experiments using 2 antibodies f irst immunohistochemically analyzed 21 patients whose egfr gene already sequenced shown ta ble 3 fig 1 pegfr tyr1173 detectable 21 patients suggesting results immunohistochemistry p egfr tyr1173 may correlated clinical responsiveness gefitinib positively immunostained cel ls tended distributed diffusely within tumor tissues hand pegfr tyr992 detected 12 57 1 21 patients undetectable 9 42 9 table 3 positive immunoreactivity detected membranes cytoplas m cells positively immunostained anti pegfr tyr992 antibody tended distributed focally wit hin tumor tissues among 12 patients whose tumors positively immunostained anti pegfr tyr99 2 antibody 9 diagnosed responders gefitinib therapy whereas 3 diagnosed nonresponders tabl e 4 furthermore among 9 patients whose tumors immunostained anti pegfr tyr992 antibody non e diagnosed responders therefore findings suggest positive immunoreactivity anti pegfr tyr 992 antibody significantly correlated clinical responsiveness gefitinib therapy p 0011 tab le 4 next analyzed correlations immunoreactivity anti pegfr tyr992 antibody mutation statu s egfr gene shown table 5 pegfr tyr992 detected 10 90 9 11 patients somatic mutations egfr gene 2 20 0 10 patients without mutations thus suggested positive immunoreactivity anti pe gfr tyr992 antibody significantly correlated presence egfr mutations p 0019 table 5 correl ation egfr genotype egfr phosphorylation status tyrosine 992 pegfr tyr992 p negative n 9 p ositive n 12 egfr genotype 0019 wild n 10 8 2 mutated n 11 1 10 table options 4 discussion gefitinib tyrosine kinase inhibitor egfr become widely used therapeutic agent patients adv anced nsclc however also noticed responsiveness nsclc gefitinib differs according race sex histopathologic features whether individual smoker 4 5 6 8 9 present study responsiveness gefitinib correlated sex significantly correlated smoking history reported k ras mutations pulmonary adenocarcinomas found frequently smokers nonsmokers 31 patients adenocarcinomas harboring k ras mutations tend exhibit resistance gefitinib 32 regard histopathologic subt ypes kim et al 33 reported pulmonary adenocarcinomas possessing predominantly papillary el ements tend exhibit good responsiveness gefitinib present study 8 16 cases papillary eleme nts histologically predominant showed responsiveness gefitinib suggesting papillary elemen ts may correlated responsiveness gefitinib however 4 5 cases diagnosed nonpapillary adenoc arcinoma showed resistance gefitinib study needed determine whether responsiveness gefitin ib may affected histopathologic subtype recently groups reported amplification egfr gene 1 6 34 35 overexpression egfr protein 34 36 37 nsclc associated clinical outcome gefitinib t herapy however present study unable find correlation expression status clinical outcome su ggesting egfr overexpression may correlated responsiveness pulmonary adenocarcinoma gefiti nib hand groups stressed 60 94 patients mutations exons encoding kinase domain egfr respon sive gefitinib 12 13 14 15 16 17 18 present study found phosphorylation status egfr tyrosi ne 992 significantly correlated mutation status egfr gene sensitivity gefitinib suggesting monitoring phosphorylation status egfr tyrosine 992 immunohistochemistry performed present

study may useful predicting responsiveness gefitinib therapy well presence egfr gene mutat
ions reported 0 3 2 patients develop severe interstitial lung disease adverse effect gefit
inib treatment 7 therefore careful judgment balancing potential benefits toxicity required
useful method predicting responsiveness gefitinib chemotherapy awaited immunohistochemical
diagnosis egfr phosphorylation status tyrosine residue 992 appears number advantages mutat
ion analysis egfr gene first immunohistochemistry anti pegfr tyr992 antibody easier perfor
m mutation analysis egfr gene 38 39 40 41 42 43 44 second immunohistochemistry facilitates
direct identification tumor cells egfr phosphorylated whereas mutation analysis contaminat
ion genomic dna normal cells cannot ruled third method less invasive patients paraffin emb
edded tissue samples taken patients histopathologic diagnosis reused immunohistochemical a
nalysis obviating need another biopsy sample mutation analysis various cancers dysregulati
on downstream signaling pathways egfr including ras raf mitogen activated protein kinase p
athway phosphoinositide 3 kinase akt pathway janus kinase signal transducers activators tr
anscription pathway phospholipase c plc diacylglycerol protein kinase c pathway reported i
nvolved tumorigenesis 45 among plc known target phosphorylated tyrosine 992 egfr 45 hand p
hosphorylated tyrosine 1173 egfr serves major binding site src homology 2 domain containin
g phosphatase 1 leads egfr dephosphorylation 45 thus present immunohistochemical data sugg
est plc may play role anticancer effect mediated gefitinib studies required address possib
ility receptor tyrosine kinase genes sequenced non small cell lung cancer nsclc matched no
rmal tissue somatic mutations epidermal growth factor receptor gene egfr found 15of 58 uns
elected tumors japan 1 61 united states treatment egfr kinase inhibitor gefitinib iressa c
auses tumor regression patients nsclc frequently japan egfr mutations found additional lun
g cancer samples u patients responded gefitinib therapy lung adenocarcinoma cell line hype
rsensitive growth inhibition gefitinib gefitinib insensitive tumors cell lines results sug
gest egfr mutations may predict sensitivity gefitinib protein kinase activation somatic mu
tation chromosomal alteration common mechanism tumorigenesis 1 inhibition activated protei
n kinases use targeted small molecule drugs antibody based strategies emerged effective ap
proach cancer therapy 2 4 recently systematic analysis kinase genes identified mutations p
rotein serine threonine kinase gene braf melanoma human cancers 5 multiple tyrosine kinase
genes phosphatidylinositol 3 kinase p110 catalytic subunit gene pik3ca human colorectal ca
rcinoma 6 7 lung carcinoma leading cause cancer deaths united states worldwide men women 8
chemotherapy non small cell lung carcinoma nsclc accounts approximately 85 lung cancer cas
es remains marginally effective 9 recently epidermal growth factor receptor egfr tyrosine
kinase inhibitor gefitinib iressa approved japan united states treatment nsclc original ra
tionale use observation egfr abundantly expressed lung carcinoma tissue adjacent normal lu
ng 10 however egfr expression detected immunohistochemistry effective predictor response g
efitinib 11 clinical trials revealed significant variability response gefitinib higher res
ponses seen japanese patients predominantly european derived population 27 5 versus 10 4 m
ulti institutional phase ii trial 12 united states partial clinical responses gefitinib ob
served frequently women nonsmokers patients adenocarcinomas 13 15 determine whether mutati
on receptor tyrosine kinases plays causal role nsclc searched somatic genetic alterations
set 119 primary nsclc tumors consisting 58 samples nagoya city university hospital japan 6
1 brigham women hospital boston massachusetts tumors included 70 lung adenocarcinomas 49 n
sclc tumors 74 male 45 female patients none documented treatment gefitinib initial screen
amplified sequenced exons encoding activation loops 47 58 human receptor tyrosine kinase g
enes 16 table s1 genomic dna subset 58 nsclc samples included 41 lung adenocarcinomas thre
e tumors lung adenocarcinomas showed heterozygous missense mutations egfr present dna norm
al lung tissue patients table s2 s0361 s0388 s0389 mutations detected amplicons receptor t
yrosine kinase genes three tumors egfr mutation predicted change leucine 858 arginine fig
1a ctg cgg l858r download high res image open new tab download powerpoint fig 1 sequence a
lignment selected regions within egfr b raf kinase domains depiction type egfr mutation hu
man nsclc egfr gb x00588 mutations nsclc tumors highlighted yellow b raf gb m95712 mutatio
ns multiple tumor types 5 highlighted blue asterisks denote residues conserved egfr b raf
l858r mutations activation loop b g719s mutant p loop c deletion mutants egfr exon 19 next
examined exons 2 25 egfr complete collection 119 nsclc tumors exon sequencing genomic dna
revealed missense deletion mutations egfr total 16 tumors within exons 18 21 kinase domain
sequence alterations group heterozygous tumor dna case paired normal lung tissue patient s
howed wild type sequence confirming mutations somatic origin distribution nucleotide prote
in sequence alterations patient characteristics associated abnormalities summarized table
s2 substitution mutations g719s l858r detected two three tumors respectively mutations loc
ated gxgxxg motif nucleotide triphosphate binding domain p loop adjacent highly conserved
dfg motif activation loop 17 respectively mutated residues nearly invariant protein kinase
s analogous residues g463 l596 b raf protein serine threonine kinase somatically mutated c
olorectal ovarian lung carcinomas 5 18 fig 1 b also detected multiple deletion mutations c
lustered region spanning codons 746 759 within kinase domain egfr ten tumors carried one t
wo overlapping 15 nucleotide deletions eliminating egfr codons 746 750 starting nucleotide

2235 2236 del 1 fig 1c table s2 egfr dna another tumor displayed heterozygous 24 nucleotid
e gap leading deletion codons 752 759 del 2 fig 1c representative chromatograms shown fig
s1 positions substitution mutations del 1 deletion three dimensional structure active form
egfr kinase domain 19 shown fig 2 note sequence alterations cluster around active site kin
ase substitution mutations lie activation loop glycine rich p loop structural elements kno
wn important autoregulation many protein kinases 17 download high res image open new tab d
ownload powerpoint fig 2 positions missense mutations g719s l858r del 1 deletion three dim
ensional structure egfr kinase domain activation loop shown yellow p loop blue c lobe n lo
be indicated residues targeted mutation deletion highlighted red del 1 mutation targets re
sidues elrea codons 746 750 egfr mutations show striking correlation patient characteristi
cs mutations frequent adenocarcinomas 15 70 21 nsclcs 1 49 2 frequent women 9 45 20 men 7
74 9 frequent patients japan 15 58 26 14 41 adenocarcinomas 32 united states 1 61 2 1 29 a
denocarcinomas 3 highest fraction egfr mutations observed japanese women adenocarcinoma 8
14 57 notably patient characteristics correlate presence egfr mutations correlate clinical
response gefitinib treatment investigate whether egfr mutations might determinant gefitini
b sensitivity pretreatment nsclc samples obtained 5 patients responded 4 patients progress
ed treatment gefitinib 125 patients treated dana farber cancer institute either expanded a
ccess program regulatory approval gefitinib 13 four patients partial radiographic response
s 50 tumor regression computed tomography scan 2 months treatment whereas fifth patient ex
perienced dramatic symptomatic improvement less 2 months patients united states caucasian
sequencing kinase domain exons 18 24 revealed mutations tumors four patients progressed ge
fitinib five tumors gefitinib responsive patients harbored egfr kinase domain mutations ch
i square test revealed difference egfr mutation frequency gefitinib responders 5 5 nonresp
onders 0 4 statistically significant p 0 0027 whereas difference gefitinib responders unse
lected u nsclc patients 5 5 versus 1 61 also significant p 10 12 20 egfr l858r mutation pr
eviously observed unselected tumors identified one gefitinib sensitive lung adenocarcinoma
fig 1a table s3 ir3t three gefitinib sensitive tumors contained heterozygous frame deletio
ns fig 1c table s3 del 3 two cases del 4 one one contained homozygous inframe deletion fig
1c table s3 del 5 deletions found within codons 746 753 egfr deletions also found unselect
ed tumors three deletions also associated amino acid substitution table s3 four samples ma
tched normal tissue available mutations confirmed somatic determine whether mutations egfr
confer gefitinib sensitivity vitro mutation status response gefitinib determined four lung
adenocarcinoma bronchioloalveolar carcinoma cell lines h3255 cell line originally derived
malignant pleural effusion caucasian female nonsmoker lung adenocarcinoma 21 cell line 50
times sensitive gefitinib lines ic50 40 nm cell survival 72 hour assay fig 3a download hi
gh res image open new tab download powerpoint fig 3 lung adenocarcinoma cell line egfr rec
eptor mutation sensitive growth signaling inhibition gefitinib cells treated gefitinib ind
icated concentrations viable cells measured 72 hours treatment percentage cell growth show
n relative untreated controls h3255 cells egfr l858r mutation whereas three remaining cell
lines wild type egfr wt b inhibition egfr phosphorylation downstream phosphorylation akt
erk1 2 cell lines treated gefitinib 24 hours cell extracts immunoblotted detect indicated
protein species akt v akt murine thymoma viral oncogene homolog erk extracellular signal
responsive kinase treatment 100 nm gefitinib completely inhibited egfr autophosphorylatio
n h3255 fig 3b treatment also inhibited phosphorylation known stream targets egfr extracel
lular signal regulated kinase 1 2 erk1 2 v akt murine thymoma viral oncogene homolog akt k
inase fig 3b correlation noted others 22 contrast three cell lines showed comparable level
s inhibition target protein phosphorylation gefitinib present concentrations roughly 100 t
imes high fig 3b sequence analysis egfr cdna four cell lines showed l858r mutations h3255
table s3 whereas three cell lines contain egfr mutations also confirmed presence l858r mu
tation primary tumor h3255 derived table s3 irg although matched normal tissue available r
esults suggest l858r mutant egfr particularly sensitive inhibition gefitinib compared wild
type enzyme likely accounts extraordinary drug sensitivity h3255 cell line identification
egfr mutations subset human lung carcinomas association egfr mutation gefitinib sensitivi
ty extend emerging paradigm whereby genetic alterations specific kinases simply kinase exp
ression render tumors sensitive selective inhibitors case imatinib treatment c kit mutant
gastrointestinal stromal tumors 23 thus although randomized trials cytotoxic therapy with
out gefitinib revealed survival benefit gefitinib treated nsclc patients 24 25 current dat
a suggest gefitinib may particularly effective treating lung cancers somatic egfr mutation
s prospective clinical trials egfr inhibition patients egfr mutations might reveal increas
ed patient survival identification egfr mutations malignancies perhaps including glioblast
omas egfr alterations already known 26 may identify patients could similarly benefit treat
ment egfr inhibitors important questions remain answered including whether alterations res
ult activated transforming alleles egfr whether receptors harboring mutations show differe
ntial sensitivity multiple egfr small molecule inhibitors whether egfr receptors harboring
mutations inhibited antibodies directed extracellular domain furthermore interest determi
ne whether resistance egfr inhibition emerges secondary mutation case imatinib treated chr

```
onic myelogenous leukemia 27 results stimulate vitro studies regarding questions finally s
triking differences frequency egfr mutation response gefitinib japanese u patients raise g
eneral questions regarding variations molecular pathogenesis cancer different ethnic cultu
ral geographic groups argue benefit population diversity cancer clinical trials '
```

In [71]:
```
no_feature
```

Out[71]:
```
100
```

In [72]:
```
test_df['Gene'].iloc[test_point_index]
```

Out[72]:
```
'EGFR'
```

In [73]:
```
test_df['Variation'].iloc[test_point_index]
```

Out[73]:
```
'A750_E758delinsP'
```

In [74]:
```
clf.coef_.shape
```

Out[74]:
```
(9, 55264)
```

In [75]:
```
indices=np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
indices[0]
```

Out[75]:
```
array([15050, 22422, 47160, 11178, 11179, 22421, 47157, 47155, 22417,
       22416, 36693, 22411, 11190, 36697, 11194, 47150, 47149, 11198,
       11199, 47146, 11202, 47145, 47142, 36698, 22406, 22405, 36699,
       36702, 36688, 47132, 36687, 36685, 47194, 11138, 36676, 47192,
       47186, 11142, 47185, 36678, 22435, 11147, 11148, 47183, 11151,
       11152, 11153, 11155, 11157, 36679, 47179, 47176, 11163, 22430,
       47172, 36682, 11168, 47170, 47168, 47164, 47131, 47130, 47128,
       11263, 47095, 36748, 36749, 47092, 36750, 36751, 11270, 47090,
       22354, 22353, 11274, 11276, 22351, 47082, 22349, 22348, 47081,
       22347, 11283, 11284, 36752, 22343, 36754, 22341, 22339, 11292,
       22360, 22362, 36740, 11259, 47127, 36703, 36704, 22396, 47122,
       36707], dtype=int64)
```

In [76]:
```
# 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 = CountVectorizer(min_df=3)

    gene_vec = gene_count_vec.fit(train_df['Gene'])
    var_vec  = var_count_vec.fit(train_df['Variation'])
    text_vec = text_count_vec.fit(train_df['TEXT'])

    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 = gene_vec.get_feature_names()[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,y
        elif (v < fea1_len+fea2_len):
            word = var_vec.get_feature_names()[v-(fea1_len)]
            yes_no = True if word == var else False
            if yes_no:
                word_present += 1
                print(i, "variation feature [{}] present in test data point [{}]".format(w
        else:
            word = text_vec.get_feature_names()[v-(fea1_len+fea2_len)]
            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,y

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

In [77]:

```
for i in range(10):
  test_point_index = i
  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_onehotCodi
  print("Actual Class :", test_y[test_point_index])
  indices=np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
  print("-"*50)
  get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].
```

```
Predicted Class : 1
Predicted Class Probabilities: [[0.4094 0.0786 0.0172 0.2598 0.0416 0.0405 0.1449 0.0045
0.0034]]
Actual Class : 4
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 7
Predicted Class Probabilities: [[0.0712 0.067  0.0154 0.1018 0.0372 0.0361 0.6642 0.004
0.0031]]
Actual Class : 7
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 1
Predicted Class Probabilities: [[0.4224 0.0736 0.0169 0.1438 0.041  0.0396 0.2549 0.0045
0.0033]]
Actual Class : 1
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 7
Predicted Class Probabilities: [[0.0746 0.1581 0.0161 0.1063 0.0388 0.038  0.5605 0.0042
0.0032]]
Actual Class : 7
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 2
Predicted Class Probabilities: [[0.0785 0.4554 0.017  0.1123 0.041  0.0399 0.2478 0.0045
0.0034]]
Actual Class : 2
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 1
Predicted Class Probabilities: [[0.5816 0.0711 0.0164 0.1083 0.0396 0.0384 0.137  0.0043
0.0033]]
Actual Class : 4
--------------------------------------------------
Out of the top  100  features  0 are present in query point
```

```
Predicted Class : 5
Predicted Class Probabilities: [[0.0963 0.0906 0.0209 0.1378 0.3925 0.0489 0.2034 0.0055
0.0041]]
Actual Class : 2
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 2
Predicted Class Probabilities: [[0.0775 0.4599 0.0168 0.1104 0.0407 0.0394 0.2475 0.0044
0.0033]]
Actual Class : 7
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 5
Predicted Class Probabilities: [[0.0917 0.0855 0.0197 0.1315 0.3408 0.0463 0.2754 0.0052
0.0038]]
Actual Class : 5
--------------------------------------------------
Out of the top  100  features  0 are present in query point
Predicted Class : 1
Predicted Class Probabilities: [[0.5816 0.0711 0.0164 0.1083 0.0396 0.0384 0.137  0.0043
0.0033]]
Actual Class : 1
--------------------------------------------------
Out of the top  100  features  0 are present in query point
```

4.1.1.4. Feature Importance, Incorrectly classified point

In [78]:

```
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
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 9
Predicted Class Probabilities: [[0.0778 0.0735 0.0169 0.1116 0.0408 0.0395 0.1421 0.0044
0.4932]]
Actual Class : 9
--------------------------------------------------
Out of the top  100  features  0 are present in query point
```

4.2. K Nearest Neighbour Classification

4.2.1. Hyper parameter tuning

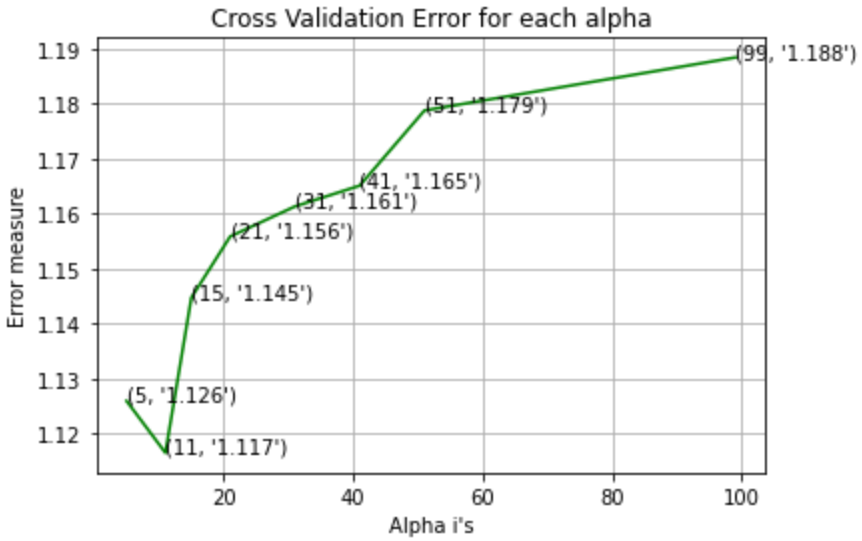
In [79]:

```
# find more about KNeighborsClassifier() here http://scikit-learn.org/stable/modules/genei
# ------------------------
# default parameter
# KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p
# 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-ne
#-------------------------------------
```

```
# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ----------------------------
# 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 estima
    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)

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
predict_y = sig_clf.predict_proba(cv_x_responseCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:
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
```

```
for alpha = 5
Log Loss : 1.1259935219443484
for alpha = 11
Log Loss : 1.116565337412182
for alpha = 15
Log Loss : 1.1447420897818323
for alpha = 21
Log Loss : 1.155863109454507
for alpha = 31
Log Loss : 1.1613186783905407
for alpha = 41
Log Loss : 1.1650970640168066
for alpha = 51
```

```
Log Loss : 1.1787486681524186
for alpha = 99
Log Loss : 1.188377932829178
```



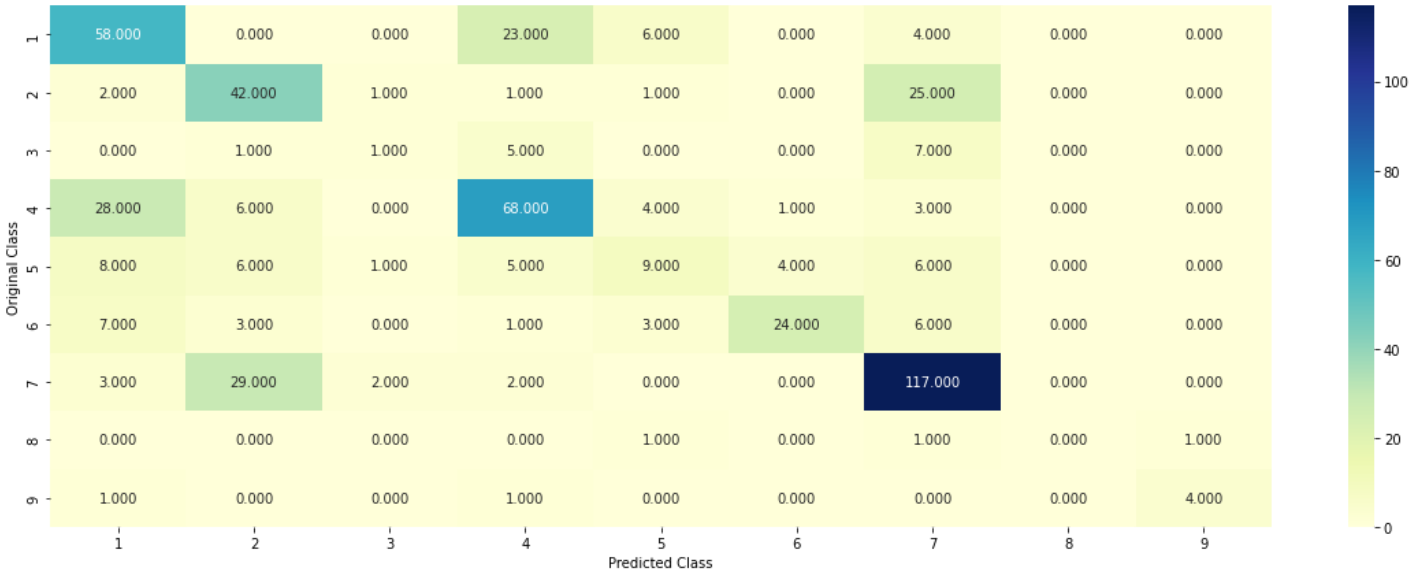

```
For values of best alpha =  11 The train log loss is: 0.5766237578158686
For values of best alpha =  11 The cross validation log loss is: 1.116565337412182
For values of best alpha =  11 The test log loss is: 0.9838927497680462
```

4.2.2. Testing the model with best hyper paramters

In [80]:

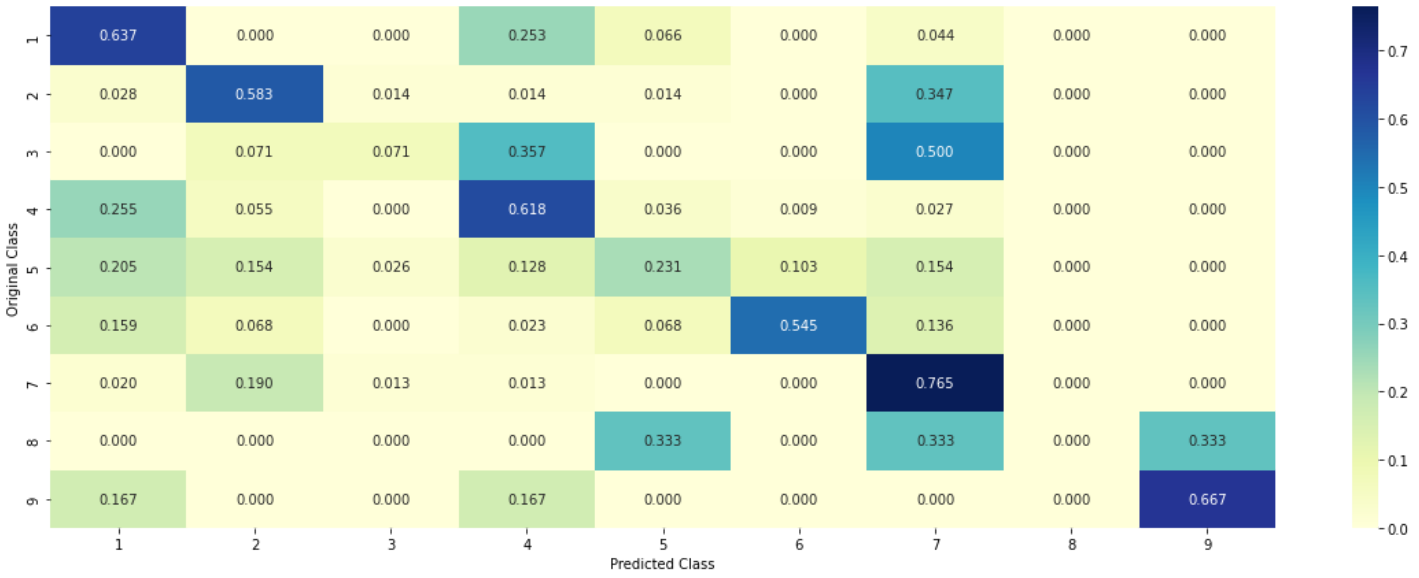
```
# find more about KNeighborsClassifier() here http://scikit-learn.org/stable/modules/gene
# ------------------------
# default parameter
# KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p
# metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

# 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-ne
#-------------------------------------
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
predict_and_plot_confusion_matrix(train_x_responseCoding, train_y, cv_x_responseCoding, cv
```

```
Log loss : 1.116565337412182
Number of mis-classified points : 0.39285714285714285
-------------------- Confusion matrix --------------------
```



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


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



4.2.3.Sample Query point -1

In [81]:

```
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[k
print("The ",alpha[best_alpha]," nearest neighbours of the test points belongs to classes"
print("Fequency of nearest points :",Counter(train_y[neighbors[1][0]]))
```

```
Predicted Class : 4
Actual Class : 7
The  11  nearest neighbours of the test points belongs to classes [7 7 7 7 7 7 7 7 7 2 7]
Fequency of nearest points : Counter({7: 10, 2: 1})
```

4.2.4. Sample Query Point-2

In [82]:

```
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[k
print("the k value for knn is",alpha[best_alpha],"and the nearest neighbours of the test p
print("Fequency of nearest points :",Counter(train_y[neighbors[1][0]]))
```

```
Predicted Class : 9
Actual Class : 9
the k value for knn is 11 and the nearest neighbours of the test points belongs to classes
[9 9 9 9 9 9 7 7 9 9 8]
Fequency of nearest points : Counter({9: 8, 7: 2, 8: 1})
```

4.3. Logistic Regression

4.3.1. With Class balancing

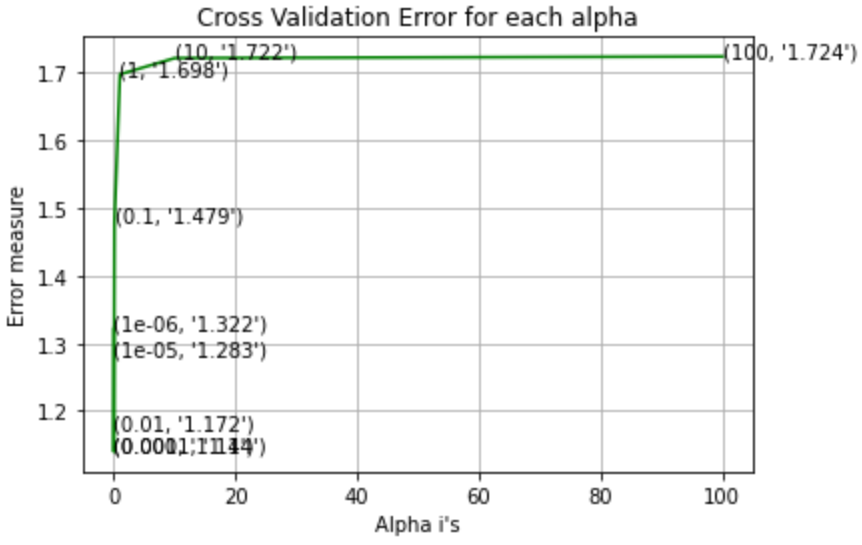
4.3.1.1. Hyper paramter tuning

In [83]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
# 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
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geor
#-------------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ------------------------------
# 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', randor
    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 estima
    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='
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
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:
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
```

```
for alpha = 1e-06
Log Loss : 1.3215095593762107
for alpha = 1e-05
Log Loss : 1.2827093746771931
for alpha = 0.0001
Log Loss : 1.1397019033364666
for alpha = 0.001
Log Loss : 1.1395607377152976
for alpha = 0.01
Log Loss : 1.1722722187976164
for alpha = 0.1
Log Loss : 1.479403962268447
for alpha = 1
Log Loss : 1.6979444364556724
for alpha = 10
Log Loss : 1.721979867481799
for alpha = 100
Log Loss : 1.7243799965769038
```




```
For values of best alpha =  0.001 The train log loss is: 0.5020426517775554
```

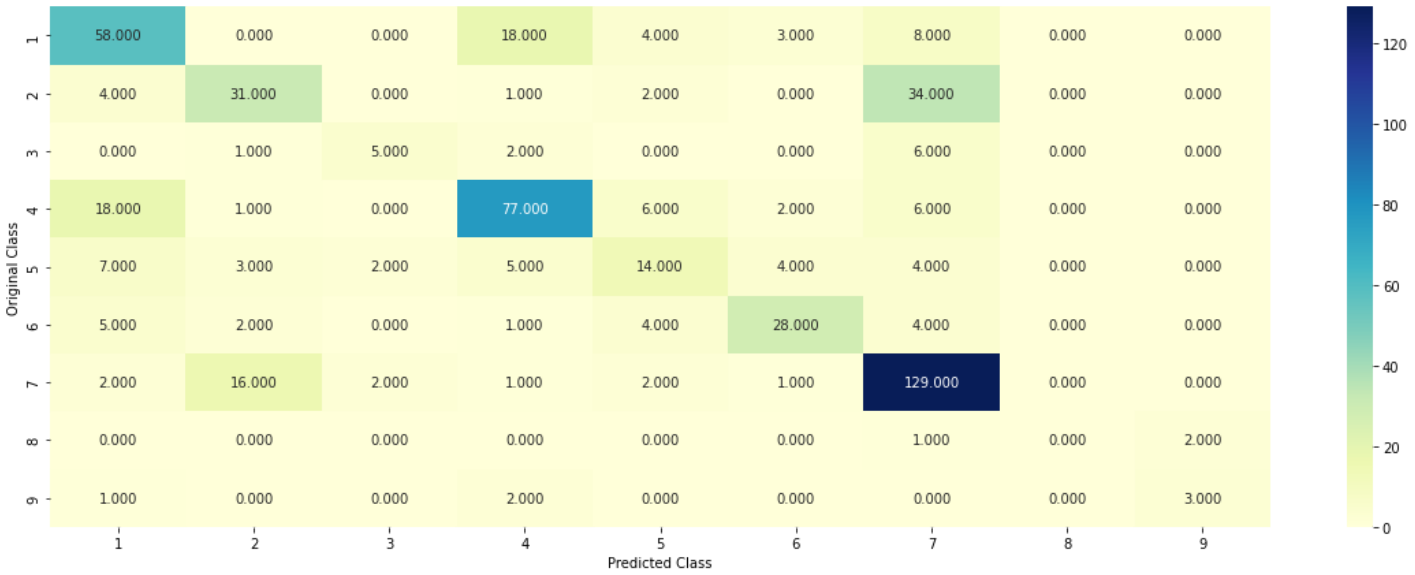
```
For values of best alpha =  0.001 The cross validation log loss is: 1.1395607377152976
For values of best alpha =  0.001 The test log loss is: 1.0577850592081122
```

4.3.1.2. Testing the model with best hyper paramters

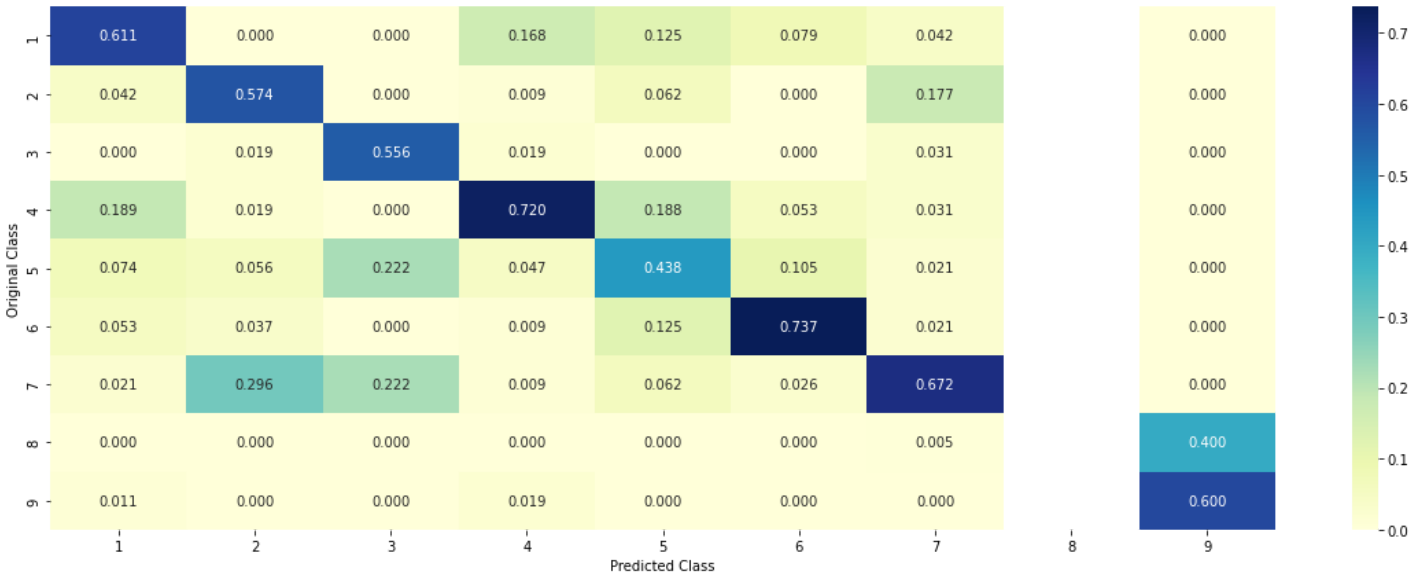
In [84]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
# 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
# predict(X)     Predict class labels for samples in X.

#-------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geor
#-------------------------------
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y, cv_x_onehotCoding, cv_y,
```

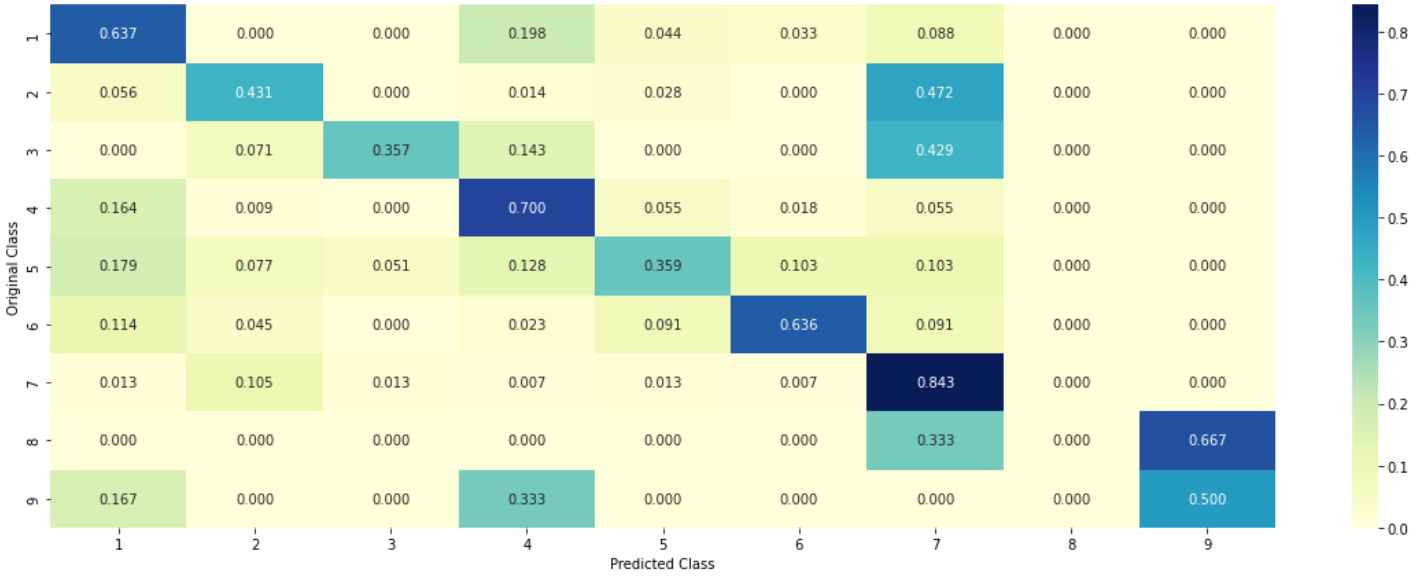
```
Log loss : 1.1395607377152976
Number of mis-classified points : 0.35150375939849626
-------------------- Confusion matrix --------------------
```




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




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



4.3.1.3. Feature Importance

In [85]:

```
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 [86]:

```
# from tabulate import tabulate
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='
clf.fit(train_x_onehotCoding,train_y)
test_point_index = 1
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
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 7
Predicted Class Probabilities: [[3.300e-03 1.303e-01 1.000e-03 4.100e-03 3.200e-03 2.000e-
03 8.529e-01
  2.400e-03 8.000e-04]]
Actual Class : 7
--------------------------------------------------
```

```
77 Text feature [y1068] present in test data point [True]
98 Text feature [constitutive] present in test data point [True]
117 Text feature [missense] present in test data point [True]
119 Text feature [blend] present in test data point [True]
124 Text feature [0011] present in test data point [True]
127 Text feature [egfrs] present in test data point [True]
129 Text feature [constitutively] present in test data point [True]
167 Text feature [0019] present in test data point [True]
168 Text feature [ligand] present in test data point [True]
172 Text feature [tyr1173] present in test data point [True]
192 Text feature [downstream] present in test data point [True]
196 Text feature [transforming] present in test data point [True]
203 Text feature [oncogene] present in test data point [True]
236 Text feature [lrea] present in test data point [True]
245 Text feature [reused] present in test data point [True]
247 Text feature [activated] present in test data point [True]
250 Text feature [tarceva] present in test data point [True]
284 Text feature [egf] present in test data point [True]
298 Text feature [tk] present in test data point [True]
304 Text feature [rc20] present in test data point [True]
335 Text feature [function] present in test data point [True]
336 Text feature [stability] present in test data point [True]
344 Text feature [activation] present in test data point [True]
351 Text feature [phospho] present in test data point [True]
354 Text feature [genephor] present in test data point [True]
357 Text feature [elicited] present in test data point [True]
384 Text feature [hpl1d] present in test data point [True]
394 Text feature [upstate] present in test data point [True]
397 Text feature [receptors] present in test data point [True]
398 Text feature [predicted] present in test data point [True]
400 Text feature [homozygous] present in test data point [True]
427 Text feature [activating] present in test data point [True]
430 Text feature [thymoma] present in test data point [True]
432 Text feature [2126] present in test data point [True]
436 Text feature [affected] present in test data point [True]
457 Text feature [strand] present in test data point [True]
461 Text feature [s752] present in test data point [True]
484 Text feature [multinucleotide] present in test data point [True]
485 Text feature [835] present in test data point [True]
Out of the top  500  features  39 are present in query point
```

4.3.1.3.2. Incorrectly Classified point

In [87]:

```
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_onehotCodinc
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 9
Predicted Class Probabilities: [[3.000e-04 0.000e+00 0.000e+00 3.000e-04 0.000e+00 0.000e+
00 9.000e-04
  9.000e-04 9.976e-01]]
Actual Class : 9
--------------------------------------------------
13 Text feature [h3k27me0] present in test data point [True]
40 Text feature [ehmt1] present in test data point [True]
46 Text feature [y641] present in test data point [True]
47 Text feature [sciex] present in test data point [True]
65 Text feature [firestein] present in test data point [True]
79 Text feature [y1067] present in test data point [True]
```

```
83 Text feature [h3k27me2] present in test data point [True]
84 Text feature [me1] present in test data point [True]
89 Text feature [psf91] present in test data point [True]
93 Text feature [amine] present in test data point [True]
99 Text feature [y641f] present in test data point [True]
103 Text feature [h3k9me2] present in test data point [True]
135 Text feature [me2] present in test data point [True]
142 Text feature [preferences] present in test data point [True]
163 Text feature [dimethylated] present in test data point [True]
181 Text feature [set7] present in test data point [True]
182 Text feature [trimethylate] present in test data point [True]
184 Text feature [dimethyltransferase] present in test data point [True]
194 Text feature [e86] present in test data point [True]
198 Text feature [y245] present in test data point [True]
202 Text feature [monomethylated] present in test data point [True]
212 Text feature [y641s] present in test data point [True]
214 Text feature [electrospray] present in test data point [True]
216 Text feature [h3k27me3] present in test data point [True]
223 Text feature [identified] present in test data point [True]
225 Text feature [methyltransferases] present in test data point [True]
233 Text feature [showed] present in test data point [True]
237 Text feature [flpe] present in test data point [True]
242 Text feature [ag] present in test data point [True]
244 Text feature [641] present in test data point [True]
388 Text feature [previously] present in test data point [True]
394 Text feature [well] present in test data point [True]
400 Text feature [h3k27me1] present in test data point [True]
427 Text feature [additional] present in test data point [True]
433 Text feature [domain] present in test data point [True]
436 Text feature [rbap48] present in test data point [True]
437 Text feature [aebp2] present in test data point [True]
440 Text feature [shown] present in test data point [True]
450 Text feature [members] present in test data point [True]
458 Text feature [two] present in test data point [True]
464 Text feature [three] present in test data point [True]
466 Text feature [mutation] present in test data point [True]
470 Text feature [addition] present in test data point [True]
472 Text feature [pfeiffer] present in test data point [True]
474 Text feature [located] present in test data point [True]
480 Text feature [discussion] present in test data point [True]
483 Text feature [y641n] present in test data point [True]
484 Text feature [found] present in test data point [True]
491 Text feature [mrm] present in test data point [True]
493 Text feature [bps] present in test data point [True]
496 Text feature [results] present in test data point [True]
498 Text feature [yoshida] present in test data point [True]
Out of the top  500  features  52 are present in query point
```

4.3.2. Without Class balancing

4.3.2.1. Hyper paramter tuning

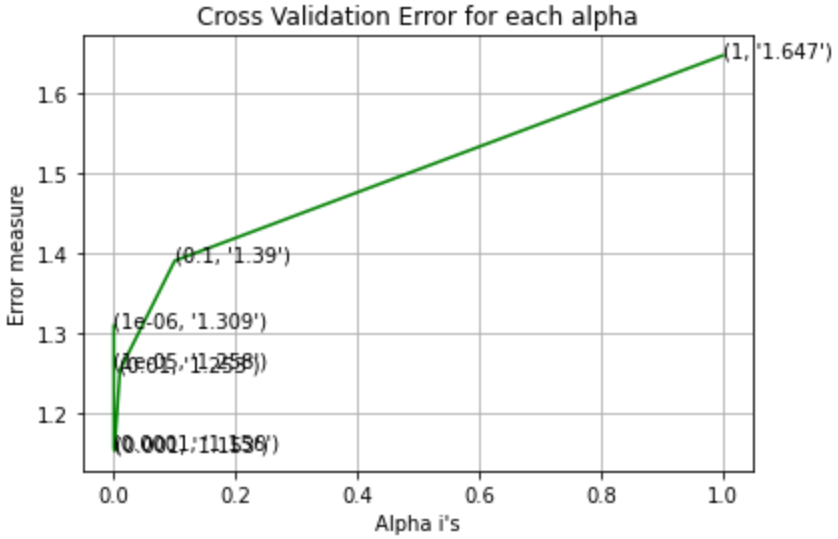
In [88]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
# 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
# predict(X)    Predict class labels for samples in X.

#------------------------------
```

```
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geor
#------------------------------



# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ----------------------------
# 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
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:
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
```

```
for alpha = 1e-06
Log Loss : 1.3091496313940838
for alpha = 1e-05
Log Loss : 1.2583414466449296
for alpha = 0.0001
Log Loss : 1.1559678604319865
for alpha = 0.001
Log Loss : 1.1526114052108944
for alpha = 0.01
Log Loss : 1.253397036858154
```

```
for alpha = 0.1
Log Loss : 1.390148563129698
for alpha = 1
Log Loss : 1.6467851174455206
```




```
For values of best alpha =  0.001 The train log loss is: 0.4988169978045054
For values of best alpha =  0.001 The cross validation log loss is: 1.1526114052108944
For values of best alpha =  0.001 The test log loss is: 1.0618564932149448
```

4.3.2.2. Testing model with best hyper parameters

In [89]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
# 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
# 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,
```

```
Log loss : 1.1526114052108944
Number of mis-classified points : 0.3533834586466165
-------------------- Confusion matrix --------------------
```

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

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

4.3.2.3. Feature Importance, Correctly Classified point

In [90]:

```
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 = 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
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 7
Predicted Class Probabilities: [[4.100e-03 1.599e-01 7.000e-04 5.400e-03 2.900e-03 1.900e-
03 8.227e-01
  2.400e-03 1.000e-04]]
Actual Class : 7
--------------------------------------------------
150 Text feature [y1068] present in test data point [True]
153 Text feature [0011] present in test data point [True]
157 Text feature [blend] present in test data point [True]
174 Text feature [constitutive] present in test data point [True]
177 Text feature [tyr1173] present in test data point [True]
212 Text feature [0019] present in test data point [True]
234 Text feature [constitutively] present in test data point [True]
243 Text feature [transforming] present in test data point [True]
275 Text feature [egfrs] present in test data point [True]
290 Text feature [reused] present in test data point [True]
310 Text feature [downstream] present in test data point [True]
315 Text feature [ligand] present in test data point [True]
319 Text feature [nonresponder] present in test data point [True]
320 Text feature [oncogene] present in test data point [True]
368 Text feature [genephor] present in test data point [True]
375 Text feature [992] present in test data point [True]
378 Text feature [summed] present in test data point [True]
384 Text feature [missense] present in test data point [True]
408 Text feature [tarceva] present in test data point [True]
411 Text feature [activated] present in test data point [True]
417 Text feature [egf] present in test data point [True]
434 Text feature [thymoma] present in test data point [True]
446 Text feature [lrea] present in test data point [True]
470 Text feature [activating] present in test data point [True]
474 Text feature [tk] present in test data point [True]
476 Text feature [2126] present in test data point [True]
Out of the top  500  features  26 are present in query point
```

4.3.2.4. Feature Importance, Inorrectly Classified point

In [91]:

```
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
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 9
Predicted Class Probabilities: [[3.940e-02 3.100e-03 0.000e+00 4.460e-02 3.000e-04 1.000e-
04 1.469e-01
  9.300e-03 7.561e-01]]
Actual Class : 9
--------------------------------------------------
40 Text feature [h3k27me0] present in test data point [True]
62 Text feature [y641] present in test data point [True]
65 Text feature [ehmt1] present in test data point [True]
76 Text feature [identified] present in test data point [True]
79 Text feature [y1067] present in test data point [True]
80 Text feature [h3k9me2] present in test data point [True]
```

81 Text feature [h3k27me2] present in test data point [True]
84 Text feature [firestein] present in test data point [True]
85 Text feature [showed] present in test data point [True]
86 Text feature [previously] present in test data point [True]
88 Text feature [y641f] present in test data point [True]
90 Text feature [located] present in test data point [True]
92 Text feature [domain] present in test data point [True]
95 Text feature [three] present in test data point [True]
96 Text feature [additional] present in test data point [True]
100 Text feature [two] present in test data point [True]
101 Text feature [mutation] present in test data point [True]
102 Text feature [shown] present in test data point [True]
105 Text feature [well] present in test data point [True]
108 Text feature [members] present in test data point [True]
110 Text feature [independent] present in test data point [True]
111 Text feature [mutations] present in test data point [True]
112 Text feature [described] present in test data point [True]
113 Text feature [addition] present in test data point [True]
115 Text feature [results] present in test data point [True]
117 Text feature [discussion] present in test data point [True]
118 Text feature [found] present in test data point [True]
119 Text feature [thus] present in test data point [True]
121 Text feature [indicating] present in test data point [True]
124 Text feature [although] present in test data point [True]
125 Text feature [protein] present in test data point [True]
126 Text feature [amino] present in test data point [True]
130 Text feature [presence] present in test data point [True]
131 Text feature [introduction] present in test data point [True]
134 Text feature [preferences] present in test data point [True]
135 Text feature [1b] present in test data point [True]
136 Text feature [activity] present in test data point [True]
138 Text feature [sciex] present in test data point [True]
139 Text feature [detected] present in test data point [True]
141 Text feature [may] present in test data point [True]
142 Text feature [fig] present in test data point [True]
143 Text feature [also] present in test data point [True]
144 Text feature [four] present in test data point [True]
145 Text feature [acid] present in test data point [True]
146 Text feature [me1] present in test data point [True]
150 Text feature [novel] present in test data point [True]
151 Text feature [affect] present in test data point [True]
152 Text feature [used] present in test data point [True]
153 Text feature [keywords] present in test data point [True]
154 Text feature [indicate] present in test data point [True]
155 Text feature [one] present in test data point [True]
157 Text feature [mutants] present in test data point [True]
158 Text feature [2c] present in test data point [True]
159 Text feature [show] present in test data point [True]
160 Text feature [therefore] present in test data point [True]
161 Text feature [revealed] present in test data point [True]
162 Text feature [type] present in test data point [True]
164 Text feature [effect] present in test data point [True]
165 Text feature [previous] present in test data point [True]
168 Text feature [encoding] present in test data point [True]
169 Text feature [shows] present in test data point [True]
170 Text feature [sequencing] present in test data point [True]
173 Text feature [1a] present in test data point [True]
174 Text feature [analyzed] present in test data point [True]
175 Text feature [reported] present in test data point [True]
177 Text feature [determine] present in test data point [True]
178 Text feature [indicated] present in test data point [True]
181 Text feature [respectively] present in test data point [True]
183 Text feature [whether] present in test data point [True]
184 Text feature [promega] present in test data point [True]
185 Text feature [furthermore] present in test data point [True]
186 Text feature [wild] present in test data point [True]

187 Text feature [suggesting] present in test data point [True]
188 Text feature [present] present in test data point [True]
190 Text feature [expected] present in test data point [True]
191 Text feature [analysis] present in test data point [True]
193 Text feature [finding] present in test data point [True]
194 Text feature [somatic] present in test data point [True]
195 Text feature [compared] present in test data point [True]
196 Text feature [vector] present in test data point [True]
197 Text feature [15] present in test data point [True]
198 Text feature [2a] present in test data point [True]
199 Text feature [majority] present in test data point [True]
200 Text feature [entire] present in test data point [True]
203 Text feature [performed] present in test data point [True]
204 Text feature [ag] present in test data point [True]
205 Text feature [10] present in test data point [True]
206 Text feature [table] present in test data point [True]
207 Text feature [2b] present in test data point [True]
209 Text feature [however] present in test data point [True]
210 Text feature [suggest] present in test data point [True]
211 Text feature [highly] present in test data point [True]
212 Text feature [similar] present in test data point [True]
213 Text feature [absence] present in test data point [True]
214 Text feature [tested] present in test data point [True]
215 Text feature [dimethylated] present in test data point [True]
218 Text feature [confirmed] present in test data point [True]
219 Text feature [represent] present in test data point [True]
220 Text feature [region] present in test data point [True]
221 Text feature [five] present in test data point [True]
222 Text feature [according] present in test data point [True]
224 Text feature [investigate] present in test data point [True]
225 Text feature [kinase] present in test data point [True]
226 Text feature [amine] present in test data point [True]
227 Text feature [dimethyltransferase] present in test data point [True]
228 Text feature [vitro] present in test data point [True]
229 Text feature [six] present in test data point [True]
231 Text feature [reduced] present in test data point [True]
232 Text feature [predicted] present in test data point [True]
233 Text feature [sequenced] present in test data point [True]
234 Text feature [including] present in test data point [True]
236 Text feature [within] present in test data point [True]
237 Text feature [whereas] present in test data point [True]
239 Text feature [using] present in test data point [True]
240 Text feature [studies] present in test data point [True]
241 Text feature [likely] present in test data point [True]
242 Text feature [missense] present in test data point [True]
243 Text feature [expressing] present in test data point [True]
245 Text feature [12] present in test data point [True]
247 Text feature [relevance] present in test data point [True]
249 Text feature [resulting] present in test data point [True]
250 Text feature [activating] present in test data point [True]
251 Text feature [selected] present in test data point [True]
254 Text feature [single] present in test data point [True]
255 Text feature [report] present in test data point [True]
257 Text feature [based] present in test data point [True]
258 Text feature [potential] present in test data point [True]
259 Text feature [contrast] present in test data point [True]
261 Text feature [transforming] present in test data point [True]
262 Text feature [due] present in test data point [True]
264 Text feature [investigated] present in test data point [True]
265 Text feature [several] present in test data point [True]
266 Text feature [individuals] present in test data point [True]
267 Text feature [known] present in test data point [True]
268 Text feature [consisting] present in test data point [True]
269 Text feature [recently] present in test data point [True]
270 Text feature [characterized] present in test data point [True]
271 Text feature [function] present in test data point [True]

272 Text feature [derived] present in test data point [True]
274 Text feature [suggests] present in test data point [True]
277 Text feature [monomethylated] present in test data point [True]
278 Text feature [new] present in test data point [True]
279 Text feature [mutagenesis] present in test data point [True]
280 Text feature [provide] present in test data point [True]
281 Text feature [identify] present in test data point [True]
282 Text feature [streptomycin] present in test data point [True]
283 Text feature [possible] present in test data point [True]
284 Text feature [seven] present in test data point [True]
285 Text feature [24] present in test data point [True]
287 Text feature [point] present in test data point [True]
288 Text feature [without] present in test data point [True]
290 Text feature [cell] present in test data point [True]
291 Text feature [y641s] present in test data point [True]
292 Text feature [encodes] present in test data point [True]
293 Text feature [y245] present in test data point [True]
294 Text feature [50] present in test data point [True]
295 Text feature [either] present in test data point [True]
297 Text feature [critical] present in test data point [True]
299 Text feature [would] present in test data point [True]
300 Text feature [1c] present in test data point [True]
301 Text feature [set7] present in test data point [True]
302 Text feature [cancer] present in test data point [True]
303 Text feature [determined] present in test data point [True]
304 Text feature [growth] present in test data point [True]
305 Text feature [except] present in test data point [True]
306 Text feature [penicillin] present in test data point [True]
307 Text feature [western] present in test data point [True]
309 Text feature [figure] present in test data point [True]
310 Text feature [result] present in test data point [True]
311 Text feature [genetic] present in test data point [True]
312 Text feature [importance] present in test data point [True]
313 Text feature [suggested] present in test data point [True]
314 Text feature [substitution] present in test data point [True]
316 Text feature [16] present in test data point [True]
317 Text feature [phosphorylation] present in test data point [True]
318 Text feature [green] present in test data point [True]
319 Text feature [data] present in test data point [True]
322 Text feature [higher] present in test data point [True]
323 Text feature [together] present in test data point [True]
324 Text feature [corresponding] present in test data point [True]
326 Text feature [studied] present in test data point [True]
328 Text feature [containing] present in test data point [True]
330 Text feature [important] present in test data point [True]
331 Text feature [40] present in test data point [True]
332 Text feature [available] present in test data point [True]
333 Text feature [since] present in test data point [True]
334 Text feature [27] present in test data point [True]
336 Text feature [activities] present in test data point [True]
337 Text feature [interestingly] present in test data point [True]
338 Text feature [considered] present in test data point [True]
340 Text feature [given] present in test data point [True]
341 Text feature [carried] present in test data point [True]
342 Text feature [me2] present in test data point [True]
343 Text feature [proteins] present in test data point [True]
344 Text feature [experiments] present in test data point [True]
345 Text feature [pcr] present in test data point [True]
346 Text feature [directed] present in test data point [True]
347 Text feature [42] present in test data point [True]
349 Text feature [25] present in test data point [True]
350 Text feature [able] present in test data point [True]
351 Text feature [vectors] present in test data point [True]
352 Text feature [sequence] present in test data point [True]
354 Text feature [indicates] present in test data point [True]
357 Text feature [human] present in test data point [True]

Text feature [assess] present in test data point [True]
Text feature [yoshida] present in test data point [True]
Text feature [caused] present in test data point [True]
Text feature [domains] present in test data point [True]
Text feature [h3k27me3] present in test data point [True]
Text feature [functional] present in test data point [True]
Text feature [inhibitor] present in test data point [True]
Text feature [product] present in test data point [True]
Text feature [ml] present in test data point [True]
Text feature [positive] present in test data point [True]
Text feature [lower] present in test data point [True]
Text feature [coding] present in test data point [True]
Text feature [cause] present in test data point [True]
Text feature [methods] present in test data point [True]
Text feature [molecular] present in test data point [True]
Text feature [use] present in test data point [True]
Text feature [support] present in test data point [True]
Text feature [failed] present in test data point [True]
Text feature [3a] present in test data point [True]
Text feature [member] present in test data point [True]
Text feature [29] present in test data point [True]
Text feature [trimethylate] present in test data point [True]
Text feature [manufacturer] present in test data point [True]
Text feature [negative] present in test data point [True]
Text feature [significantly] present in test data point [True]
Text feature [hours] present in test data point [True]
Text feature [study] present in test data point [True]
Text feature [cells] present in test data point [True]
Text feature [carrying] present in test data point [True]
Text feature [led] present in test data point [True]
Text feature [consistent] present in test data point [True]
Text feature [acids] present in test data point [True]
Text feature [summary] present in test data point [True]
Text feature [33] present in test data point [True]
Text feature [28] present in test data point [True]
Text feature [observed] present in test data point [True]
Text feature [37] present in test data point [True]
Text feature [another] present in test data point [True]
Text feature [might] present in test data point [True]
Text feature [examined] present in test data point [True]
Text feature [gene] present in test data point [True]
Text feature [directly] present in test data point [True]
Text feature [23] present in test data point [True]
Text feature [total] present in test data point [True]
Text feature [obtained] present in test data point [True]
Text feature [supplementary] present in test data point [True]
Text feature [effects] present in test data point [True]
Text feature [31] present in test data point [True]
Text feature [responsible] present in test data point [True]
Text feature [functionally] present in test data point [True]
Text feature [conserved] present in test data point [True]
Text feature [could] present in test data point [True]
Text feature [possibility] present in test data point [True]
Text feature [anti] present in test data point [True]
Text feature [occurred] present in test data point [True]
Text feature [binding] present in test data point [True]
Text feature [various] present in test data point [True]
Text feature [case] present in test data point [True]
Text feature [informed] present in test data point [True]
Text feature [evaluate] present in test data point [True]
Text feature [activation] present in test data point [True]
Text feature [transfected] present in test data point [True]
Text feature [reagent] present in test data point [True]
Text feature [30] present in test data point [True]
Text feature [germline] present in test data point [True]
Text feature [part] present in test data point [True]

```
451 Text feature [separate] present in test data point [True]
452 Text feature [frequently] present in test data point [True]
453 Text feature [resulted] present in test data point [True]
455 Text feature [included] present in test data point [True]
456 Text feature [different] present in test data point [True]
457 Text feature [complete] present in test data point [True]
458 Text feature [panel] present in test data point [True]
459 Text feature [added] present in test data point [True]
460 Text feature [remains] present in test data point [True]
461 Text feature [lead] present in test data point [True]
463 Text feature [full] present in test data point [True]
464 Text feature [transfection] present in test data point [True]
465 Text feature [antibodies] present in test data point [True]
466 Text feature [measured] present in test data point [True]
468 Text feature [key] present in test data point [True]
469 Text feature [similarly] present in test data point [True]
470 Text feature [essential] present in test data point [True]
471 Text feature [significant] present in test data point [True]
472 Text feature [verified] present in test data point [True]
474 Text feature [comparison] present in test data point [True]
475 Text feature [aebp2] present in test data point [True]
476 Text feature [rbap48] present in test data point [True]
477 Text feature [strongly] present in test data point [True]
478 Text feature [1d] present in test data point [True]
479 Text feature [moreover] present in test data point [True]
480 Text feature [generated] present in test data point [True]
481 Text feature [approximately] present in test data point [True]
483 Text feature [h3k27me1] present in test data point [True]
485 Text feature [assay] present in test data point [True]
486 Text feature [individual] present in test data point [True]
487 Text feature [expressed] present in test data point [True]
488 Text feature [constructs] present in test data point [True]
489 Text feature [times] present in test data point [True]
490 Text feature [next] present in test data point [True]
491 Text feature [enhanced] present in test data point [True]
492 Text feature [family] present in test data point [True]
494 Text feature [26] present in test data point [True]
495 Text feature [phenotype] present in test data point [True]
496 Text feature [system] present in test data point [True]
497 Text feature [form] present in test data point [True]
499 Text feature [leading] present in test data point [True]
Out of the top  500  features  311 are present in query point
```

4.4. Linear Support Vector Machines

4.4.1. Hyper paramter tuning

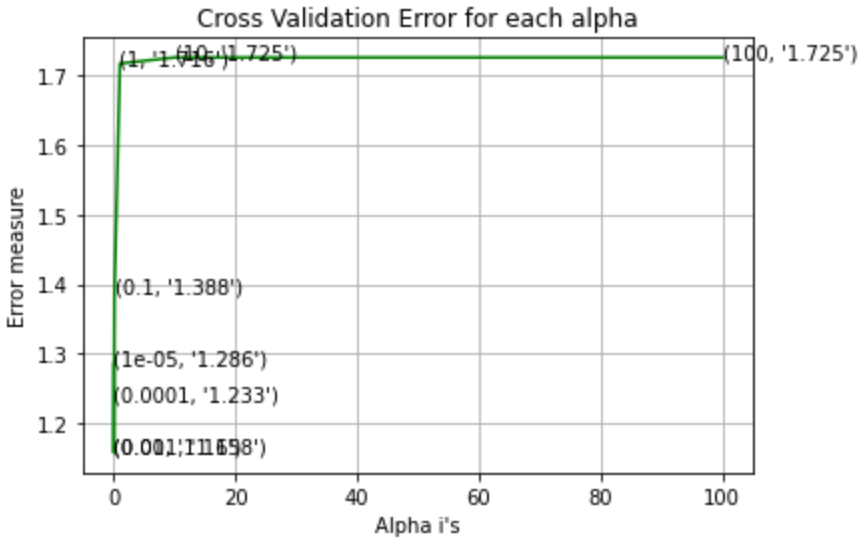
In [92]:

```
# read more about support vector machines with linear kernals here http://scikit-learn.org

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

# 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/math
# ---------------------------------
```

```
# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ----------------------------
# 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', ran
    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='
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
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log loss is:
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
```

```
for C = 1e-05
Log Loss : 1.286248506485416
for C = 0.0001
Log Loss : 1.2328190663940635
for C = 0.001
Log Loss : 1.1584112874620767
for C = 0.01
Log Loss : 1.1603787621235608
for C = 0.1
Log Loss : 1.3880543455847025
for C = 1
Log Loss : 1.716087544365558
for C = 10
```

```
Log Loss : 1.7247375123577955
for C = 100
Log Loss : 1.7247375284464248
```




```
For values of best alpha =  0.001 The train log loss is: 0.5015952116276196
For values of best alpha =  0.001 The cross validation log loss is: 1.1584112874620767
For values of best alpha =  0.001 The test log loss is: 1.1104511493656004
```

4.4.2. Testing model with best hyper parameters

In [93]:

```
# read more about support vector machines with linear kernals here http://scikit-learn.org

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

# 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/math
# ---------------------------------


# 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,c
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y,cv_x_onehotCoding,cv_y, cl
```

```
Log loss : 1.1584112874620767
Number of mis-classified points : 0.36278195488721804
-------------------- Confusion matrix --------------------
```

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

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

4.3.3. Feature Importance

4.3.3.1. For Correctly classified point

In [94]:

```
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
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.066  0.3774 0.0043 0.0413 0.0101 0.0082 0.4869 0.0022
0.0035]]
Actual Class : 7
--------------------------------------------------
194 Text feature [0011] present in test data point [True]
195 Text feature [tyr1173] present in test data point [True]
236 Text feature [blend] present in test data point [True]
240 Text feature [nonresponder] present in test data point [True]
257 Text feature [992] present in test data point [True]
258 Text feature [summed] present in test data point [True]
281 Text feature [0019] present in test data point [True]
331 Text feature [nonpapillary] present in test data point [True]
342 Text feature [reused] present in test data point [True]
354 Text feature [1173] present in test data point [True]
359 Text feature [immunostained] present in test data point [True]
377 Text feature [genephor] present in test data point [True]
417 Text feature [y1068] present in test data point [True]
420 Text feature [2126] present in test data point [True]
435 Text feature [obviating] present in test data point [True]
488 Text feature [judgment] present in test data point [True]
Out of the top  500  features  16 are present in query point
```

4.3.3.2. For Incorrectly classified point

In [95]:

```
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
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-1*abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].il
```

```
Predicted Class : 9
Predicted Class Probabilities: [[3.900e-03 1.660e-02 6.000e-04 7.290e-02 1.100e-03 2.000e-
04 8.210e-02
  1.500e-03 8.211e-01]]
Actual Class : 9
--------------------------------------------------
6 Text feature [h3k27me0] present in test data point [True]
29 Text feature [ehmt1] present in test data point [True]
33 Text feature [y641] present in test data point [True]
42 Text feature [y1067] present in test data point [True]
59 Text feature [me1] present in test data point [True]
61 Text feature [firestein] present in test data point [True]
74 Text feature [amine] present in test data point [True]
75 Text feature [y641f] present in test data point [True]
76 Text feature [h3k27me2] present in test data point [True]
77 Text feature [h3k9me2] present in test data point [True]
81 Text feature [sciex] present in test data point [True]
99 Text feature [preferences] present in test data point [True]
109 Text feature [dimethyltransferase] present in test data point [True]
```

```
111 Text feature [set7] present in test data point [True]
114 Text feature [y245] present in test data point [True]
125 Text feature [dimethylated] present in test data point [True]
128 Text feature [monomethylated] present in test data point [True]
136 Text feature [trimethylate] present in test data point [True]
143 Text feature [methyltransferases] present in test data point [True]
147 Text feature [641] present in test data point [True]
150 Text feature [y641s] present in test data point [True]
155 Text feature [me2] present in test data point [True]
189 Text feature [aebp2] present in test data point [True]
190 Text feature [rbap48] present in test data point [True]
191 Text feature [psf91] present in test data point [True]
196 Text feature [yoshida] present in test data point [True]
203 Text feature [pdbid] present in test data point [True]
213 Text feature [catalyzes] present in test data point [True]
485 Text feature [resuspension] present in test data point [True]
Out of the top  500  features  29 are present in query point
```

4.5 Random Forest Classifier

4.5.1. Hyper paramter tuning (With One hot Encoding)

In [96]:

```
# ------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=Nor
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=Nc
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, v
# 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/rand
# ------------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ----------------------------
# 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, randor
        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=1
        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_
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_
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:
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best estimator = ', alpha[int(best_alpha/2)], "The cross validation l
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:'
```

```
for n_estimators = 100 and max depth =  5
Log Loss : 1.2668142046618969
for n_estimators = 100 and max depth =  10
Log Loss : 1.1925838709073378
for n_estimators = 200 and max depth =  5
Log Loss : 1.2507053362553289
for n_estimators = 200 and max depth =  10
Log Loss : 1.1816843395830663
for n_estimators = 500 and max depth =  5
Log Loss : 1.243815537761731
for n_estimators = 500 and max depth =  10
Log Loss : 1.1762881614157652
for n_estimators = 1000 and max depth =  5
Log Loss : 1.2430196230647421
for n_estimators = 1000 and max depth =  10
Log Loss : 1.1729490178619308
for n_estimators = 2000 and max depth =  5
Log Loss : 1.2442899618978844
for n_estimators = 2000 and max depth =  10
Log Loss : 1.173706167936959
For values of best estimator =  1000 The train log loss is: 0.6702278344397153
For values of best estimator =  1000 The cross validation log loss is: 1.1729490178619308
For values of best estimator =  1000 The test log loss is: 1.1218840644342052
```

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

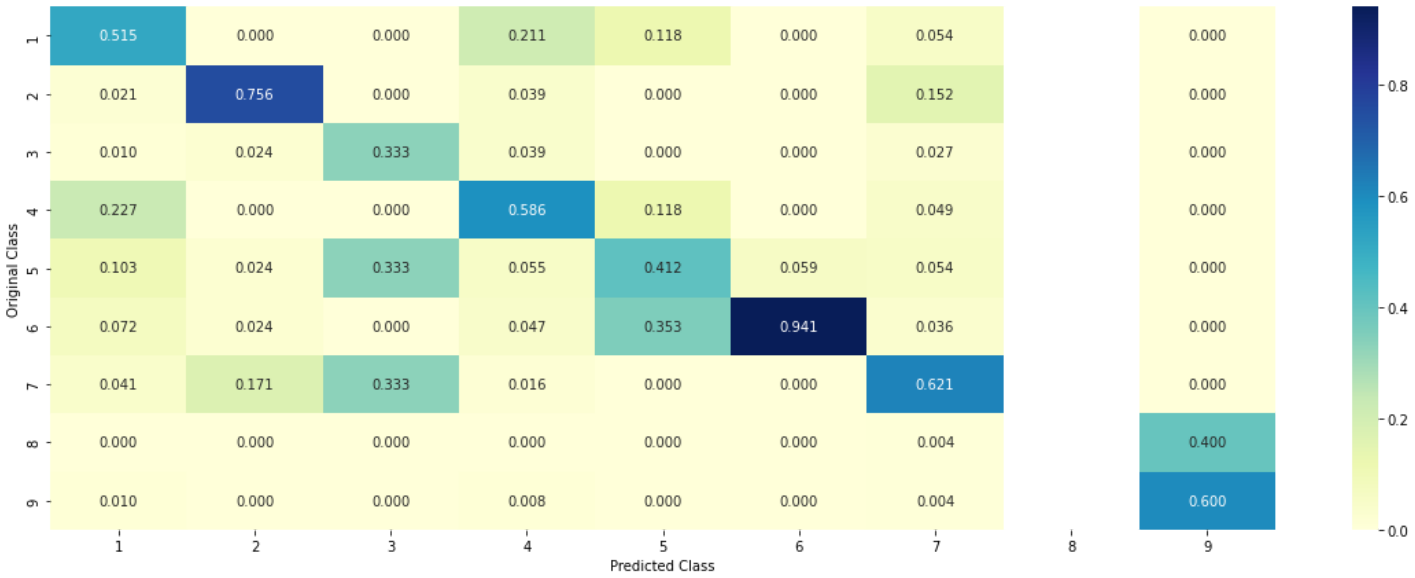
In [97]:

```
# ------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=Nor
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=No
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, v
```

```
# 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/rand
# --------------------------------

clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y,cv_x_onehotCoding,cv_y, cl
```

```
Log loss : 1.1729490178619308
Number of mis-classified points : 0.39473684210526316
-------------------- 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 [98]:

```
# test_point_index = 10
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_
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_onehotCodinc
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
```

```
Predicted Class : 7
Predicted Class Probabilities: [[0.0243 0.0814 0.0145 0.0213 0.0307 0.0275 0.7935 0.0038
0.003 ]]
Actual Class : 7
--------------------------------------------------
0 Text feature [kinase] present in test data point [True]
1 Text feature [tyrosine] present in test data point [True]
2 Text feature [activating] present in test data point [True]
3 Text feature [phosphorylation] present in test data point [True]
4 Text feature [activated] present in test data point [True]
5 Text feature [signaling] present in test data point [True]
6 Text feature [constitutive] present in test data point [True]
7 Text feature [activation] present in test data point [True]
9 Text feature [inhibitors] present in test data point [True]
10 Text feature [missense] present in test data point [True]
11 Text feature [inhibitor] present in test data point [True]
12 Text feature [erk] present in test data point [True]
15 Text feature [akt] present in test data point [True]
16 Text feature [function] present in test data point [True]
17 Text feature [oncogenic] present in test data point [True]
18 Text feature [treatment] present in test data point [True]
20 Text feature [growth] present in test data point [True]
21 Text feature [constitutively] present in test data point [True]
22 Text feature [downstream] present in test data point [True]
25 Text feature [kinases] present in test data point [True]
```

```
28 Text feature [activate] present in test data point [True]
29 Text feature [tki] present in test data point [True]
30 Text feature [cells] present in test data point [True]
31 Text feature [receptor] present in test data point [True]
32 Text feature [protein] present in test data point [True]
33 Text feature [inhibition] present in test data point [True]
37 Text feature [variants] present in test data point [True]
38 Text feature [trials] present in test data point [True]
39 Text feature [drug] present in test data point [True]
40 Text feature [functional] present in test data point [True]
44 Text feature [therapy] present in test data point [True]
45 Text feature [autophosphorylation] present in test data point [True]
46 Text feature [egfr] present in test data point [True]
47 Text feature [erk1] present in test data point [True]
50 Text feature [mitogen] present in test data point [True]
51 Text feature [patients] present in test data point [True]
53 Text feature [imatinib] present in test data point [True]
54 Text feature [stability] present in test data point [True]
55 Text feature [months] present in test data point [True]
59 Text feature [treated] present in test data point [True]
60 Text feature [ic50] present in test data point [True]
62 Text feature [respond] present in test data point [True]
63 Text feature [lines] present in test data point [True]
64 Text feature [inhibited] present in test data point [True]
65 Text feature [clinical] present in test data point [True]
68 Text feature [cell] present in test data point [True]
71 Text feature [therapeutic] present in test data point [True]
72 Text feature [transforming] present in test data point [True]
74 Text feature [nsclc] present in test data point [True]
75 Text feature [extracellular] present in test data point [True]
76 Text feature [dna] present in test data point [True]
77 Text feature [inactivation] present in test data point [True]
78 Text feature [phospho] present in test data point [True]
80 Text feature [amplification] present in test data point [True]
81 Text feature [proteins] present in test data point [True]
83 Text feature [likelihood] present in test data point [True]
85 Text feature [mek] present in test data point [True]
87 Text feature [daily] present in test data point [True]
88 Text feature [factor] present in test data point [True]
89 Text feature [gene] present in test data point [True]
90 Text feature [effective] present in test data point [True]
91 Text feature [sensitive] present in test data point [True]
93 Text feature [resistance] present in test data point [True]
94 Text feature [expressing] present in test data point [True]
97 Text feature [oncogene] present in test data point [True]
98 Text feature [type] present in test data point [True]
Out of the top  100  features  66 are present in query point
```

4.5.3.2. Inorrectly Classified point

In [99]:

```
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_onehotCodinc
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
```

```
Predicted Class : 9
Predicted Class Probabilities: [[0.0754 0.0649 0.0123 0.1001 0.027  0.0272 0.1803 0.0059
0.5069]]
Actuall Class : 9
--------------------------------------------------
```

```
0 Text feature [kinase] present in test data point [True]
1 Text feature [tyrosine] present in test data point [True]
2 Text feature [activating] present in test data point [True]
3 Text feature [phosphorylation] present in test data point [True]
4 Text feature [activated] present in test data point [True]
5 Text feature [signaling] present in test data point [True]
7 Text feature [activation] present in test data point [True]
8 Text feature [suppressor] present in test data point [True]
9 Text feature [inhibitors] present in test data point [True]
10 Text feature [missense] present in test data point [True]
11 Text feature [inhibitor] present in test data point [True]
12 Text feature [erk] present in test data point [True]
15 Text feature [akt] present in test data point [True]
16 Text feature [function] present in test data point [True]
17 Text feature [oncogenic] present in test data point [True]
18 Text feature [treatment] present in test data point [True]
20 Text feature [growth] present in test data point [True]
21 Text feature [constitutively] present in test data point [True]
22 Text feature [downstream] present in test data point [True]
23 Text feature [brca1] present in test data point [True]
25 Text feature [kinases] present in test data point [True]
27 Text feature [loss] present in test data point [True]
30 Text feature [cells] present in test data point [True]
32 Text feature [protein] present in test data point [True]
33 Text feature [inhibition] present in test data point [True]
36 Text feature [yeast] present in test data point [True]
37 Text feature [variants] present in test data point [True]
38 Text feature [trials] present in test data point [True]
39 Text feature [drug] present in test data point [True]
40 Text feature [functional] present in test data point [True]
42 Text feature [defective] present in test data point [True]
43 Text feature [variant] present in test data point [True]
44 Text feature [therapy] present in test data point [True]
45 Text feature [autophosphorylation] present in test data point [True]
50 Text feature [mitogen] present in test data point [True]
51 Text feature [patients] present in test data point [True]
54 Text feature [stability] present in test data point [True]
55 Text feature [months] present in test data point [True]
57 Text feature [efficacy] present in test data point [True]
58 Text feature [proliferation] present in test data point [True]
59 Text feature [treated] present in test data point [True]
63 Text feature [lines] present in test data point [True]
64 Text feature [inhibited] present in test data point [True]
65 Text feature [clinical] present in test data point [True]
68 Text feature [cell] present in test data point [True]
69 Text feature [neutral] present in test data point [True]
71 Text feature [therapeutic] present in test data point [True]
72 Text feature [transforming] present in test data point [True]
76 Text feature [dna] present in test data point [True]
77 Text feature [inactivation] present in test data point [True]
79 Text feature [abolish] present in test data point [True]
80 Text feature [amplification] present in test data point [True]
81 Text feature [proteins] present in test data point [True]
82 Text feature [potency] present in test data point [True]
84 Text feature [pten] present in test data point [True]
85 Text feature [mek] present in test data point [True]
87 Text feature [daily] present in test data point [True]
88 Text feature [factor] present in test data point [True]
89 Text feature [gene] present in test data point [True]
90 Text feature [effective] present in test data point [True]
93 Text feature [resistance] present in test data point [True]
94 Text feature [expressing] present in test data point [True]
97 Text feature [oncogene] present in test data point [True]
98 Text feature [type] present in test data point [True]
Out of the top  100  features  64 are present in query point
```

4.5.3. Hyper paramter tuning (With Response Coding)

In [100…

```
# -------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=Nor
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=Nc
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, v
# 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/rand
# -------------------------------


# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/ge
# ----------------------------
# 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, randor
        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=1
        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_
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_
```

```
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:",lo
predict_y = sig_clf.predict_proba(cv_x_responseCoding)
print('For values of best alpha = ', alpha[int(best_alpha/4)], "The cross validation log l
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
```

```
for n_estimators = 10 and max depth =  2
Log Loss : 2.1552595784106283
for n_estimators = 10 and max depth =  3
Log Loss : 1.7966069582625808
for n_estimators = 10 and max depth =  5
Log Loss : 1.5644478753846434
for n_estimators = 10 and max depth =  10
Log Loss : 1.7206150729479543
for n_estimators = 50 and max depth =  2
Log Loss : 1.7911031495315912
for n_estimators = 50 and max depth =  3
Log Loss : 1.5382964604433969
for n_estimators = 50 and max depth =  5
Log Loss : 1.469392488889737
for n_estimators = 50 and max depth =  10
Log Loss : 1.6823406029007182
for n_estimators = 100 and max depth =  2
Log Loss : 1.6427050275675261
for n_estimators = 100 and max depth =  3
Log Loss : 1.549971175282965
for n_estimators = 100 and max depth =  5
Log Loss : 1.3848766801392456
for n_estimators = 100 and max depth =  10
Log Loss : 1.7234078500299568
for n_estimators = 200 and max depth =  2
Log Loss : 1.6856539223857818
for n_estimators = 200 and max depth =  3
Log Loss : 1.5408189514675088
for n_estimators = 200 and max depth =  5
Log Loss : 1.4423928227567018
for n_estimators = 200 and max depth =  10
Log Loss : 1.768022546205898
for n_estimators = 500 and max depth =  2
Log Loss : 1.7276220213680489
for n_estimators = 500 and max depth =  3
Log Loss : 1.5767652696742693
for n_estimators = 500 and max depth =  5
Log Loss : 1.4547028322015
for n_estimators = 500 and max depth =  10
Log Loss : 1.7549095378192645
for n_estimators = 1000 and max depth =  2
Log Loss : 1.7255018706875715
for n_estimators = 1000 and max depth =  3
Log Loss : 1.5909688381826237
for n_estimators = 1000 and max depth =  5
Log Loss : 1.457108533790439
for n_estimators = 1000 and max depth =  10
Log Loss : 1.7374649370053212
For values of best alpha =  100 The train log loss is: 0.06995036734599012
For values of best alpha =  100 The cross validation log loss is: 1.3848766801392456
For values of best alpha =  100 The test log loss is: 1.2969170346443397
```

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

In [101…

```
# -------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=Nor
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=Nc
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, v
# 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/rand
# -------------------------------

clf = RandomForestClassifier(max_depth=max_depth[int(best_alpha%4)], n_estimators=alpha[ir
predict_and_plot_confusion_matrix(train_x_responseCoding, train_y,cv_x_responseCoding,cv_y
```

```
Log loss : 1.3848766801392456
Number of mis-classified points : 0.4943609022556391
-------------------- 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 [102…

```
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/4)], criterion='gini', max_
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_responseCodi
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 : 2
Predicted Class Probabilities: [[0.014  0.4421 0.1214 0.0272 0.0333 0.0452 0.2814 0.0239
0.0115]]
Actual Class : 7
--------------------------------------------------
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
Text is important feature
Gene 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
Variation is important feature
Text is important feature
Text is important feature
Text is important feature
Gene is important feature
Gene is important feature
Gene is important feature
```

4.5.5.2. Incorrectly Classified point

In [103…

```
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("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_responseCodi
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 : 9
Predicted Class Probabilities: [[0.0383 0.1474 0.0923 0.0425 0.0614 0.0577 0.0969 0.1541
0.3094]]
Actual Class : 9
--------------------------------------------------
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
Text is important feature
Gene 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
Variation is important feature
Text is important feature
Text is important feature
Text 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 [104…

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/skle
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=Tru
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optima
# 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
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geor
#-------------------------------


# read more about support vector machines with linear kernals here http://scikit-learn.org
# ---------------------------------
# default parameters
# SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=
# cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='

# 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/math
# ---------------------------------


# read more about support vector machines with linear kernals here http://scikit-learn.org
# ---------------------------------
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=Non
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=No
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, v
# 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/rand
# ---------------------------------


clf1 = SGDClassifier(alpha=0.001, penalty='l2', loss='log', class_weight='balanced', rando
clf1.fit(train_x_onehotCoding, train_y)
sig_clf1 = CalibratedClassifierCV(clf1, method="sigmoid")

clf2 = SGDClassifier(alpha=1, penalty='l2', loss='hinge', class_weight='balanced', random_
clf2.fit(train_x_onehotCoding, train_y)
sig_clf2 = CalibratedClassifierCV(clf2, method="sigmoid")
```

```
clf3 = MultinomialNB(alpha=0.001)
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
sig_clf2.fit(train_x_onehotCoding, train_y)
print("Support vector machines : Log Loss: %0.2f" % (log_loss(cv_y, sig_clf2.predict_proba
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_onehot
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=
    sclf.fit(train_x_onehotCoding, train_y)
    print("Stacking Classifer : for the value of alpha: %f Log Loss: %0.3f" % (i, log_loss
    log_error =log_loss(cv_y, sclf.predict_proba(cv_x_onehotCoding))
    if best_alpha > log_error:
        best_alpha = log_error
```

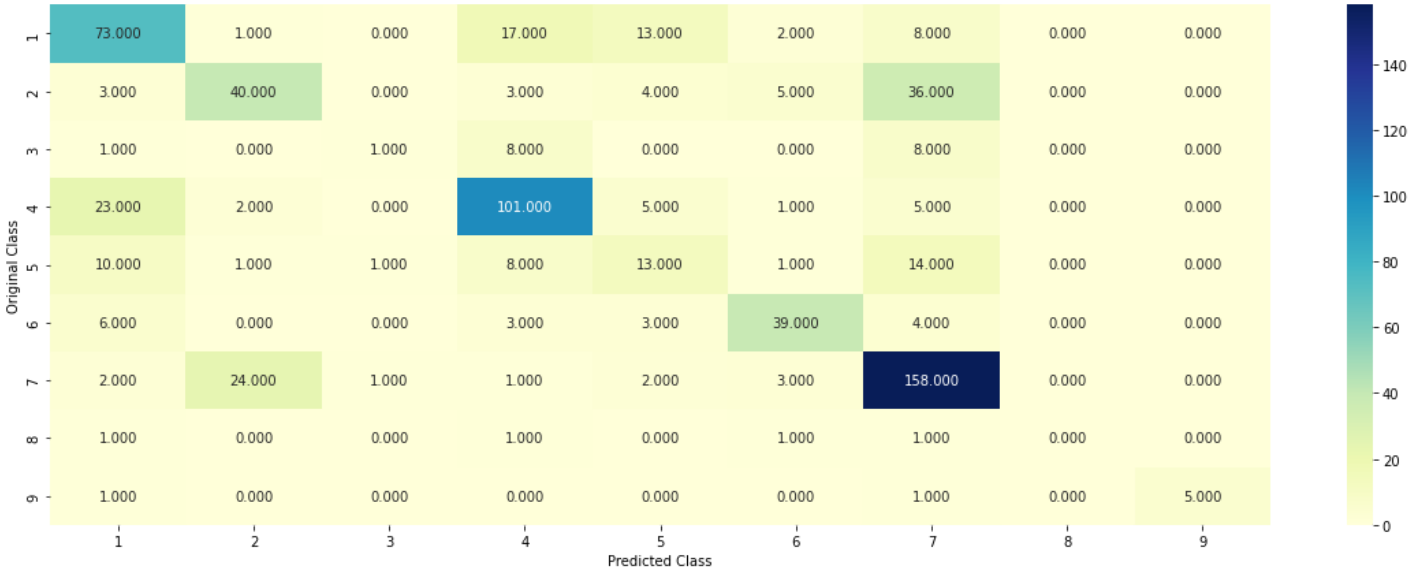
```
Logistic Regression :  Log Loss: 1.14
Support vector machines : Log Loss: 1.72
Naive Bayes : Log Loss: 1.29
--------------------------------------------------
Stacking Classifer : for the value of alpha: 0.000100 Log Loss: 1.819
Stacking Classifer : for the value of alpha: 0.001000 Log Loss: 1.726
Stacking Classifer : for the value of alpha: 0.010000 Log Loss: 1.352
Stacking Classifer : for the value of alpha: 0.100000 Log Loss: 1.219
Stacking Classifer : for the value of alpha: 1.000000 Log Loss: 1.500
Stacking Classifer : for the value of alpha: 10.000000 Log Loss: 1.880
```

4.7.2 testing the model with the best hyper parameters

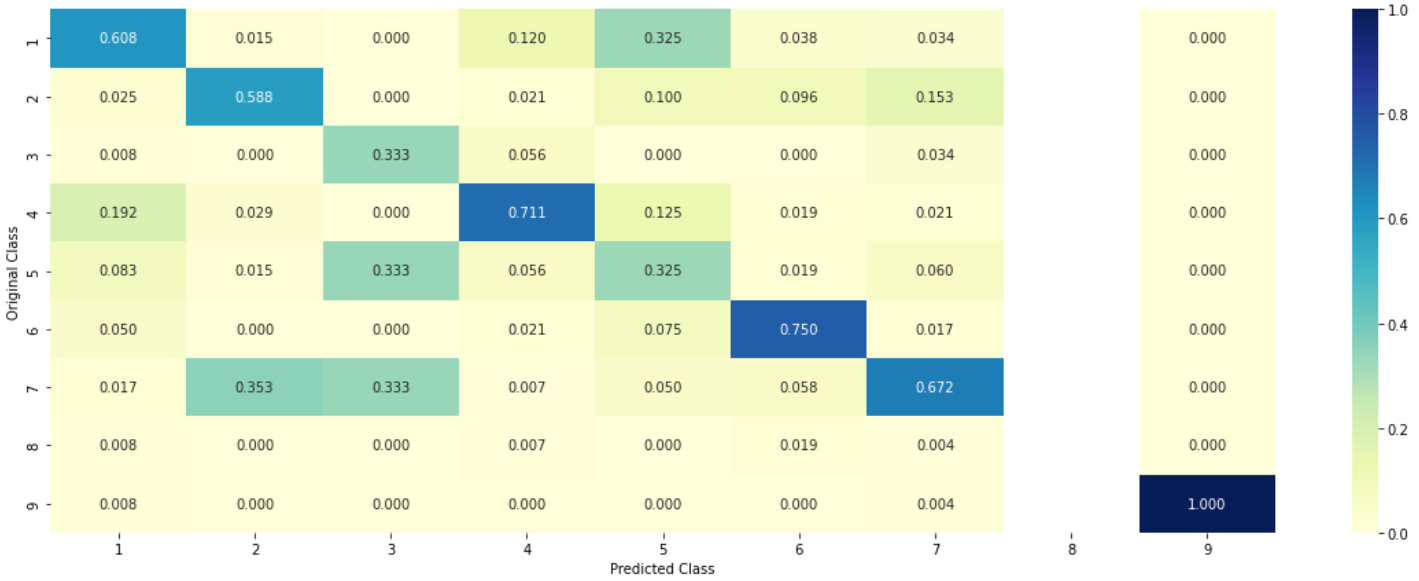
In [105…

```
lr = LogisticRegression(C=0.1)
sclf = StackingClassifier(classifiers=[sig_clf1, sig_clf2, sig_clf3], meta_classifier=lr,
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_onehotCodi
plot_confusion_matrix(test_y=test_y, predict_y=sclf.predict(test_x_onehotCoding))
```

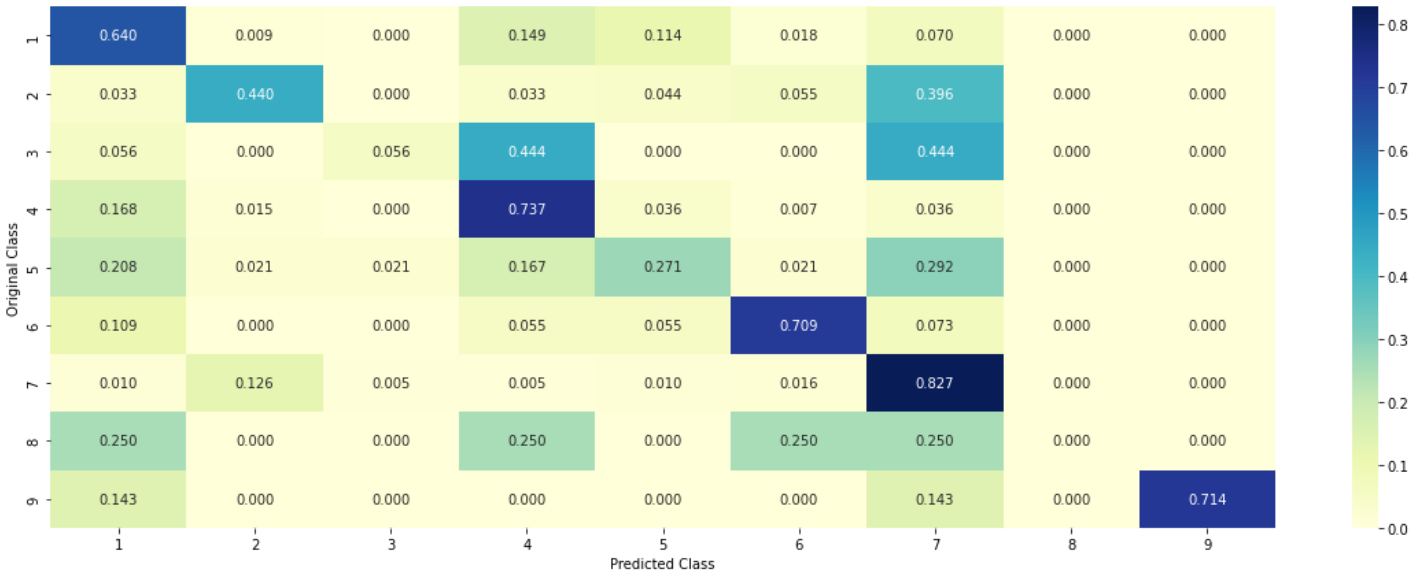
```
Log loss (train) on the stacking classifier : 0.4755047563847262
Log loss (CV) on the stacking classifier : 1.2187709296105815
Log loss (test) on the stacking classifier : 1.142759962875632
Number of missclassified point : 0.3533834586466165
-------------------- Confusion matrix --------------------
```



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



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



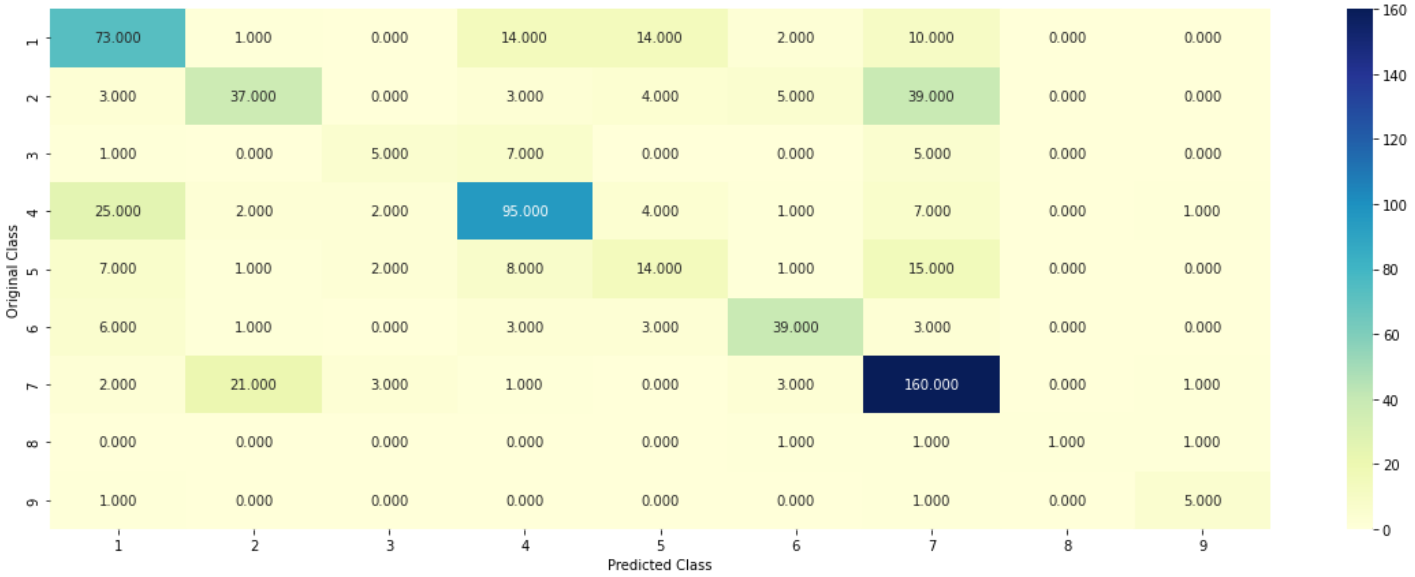
4.7.3 Maximum Voting classifier

In [106…

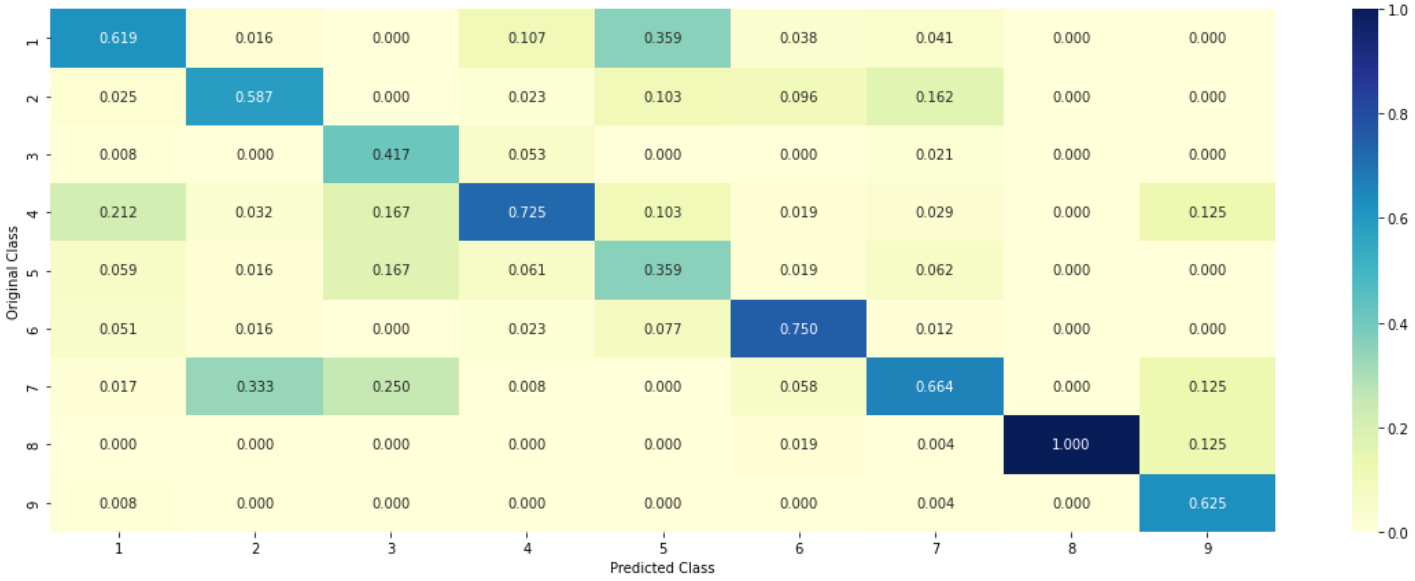
```
#Refer:http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.
from sklearn.ensemble import VotingClassifier
vclf = VotingClassifier(estimators=[('lr', sig_clf1), ('svc', sig_clf2), ('rf', sig_clf3)]
vclf.fit(train_x_onehotCoding, train_y)
```

```
print("Log loss (train) on the VotingClassifier :", log_loss(train_y, vclf.predict_proba(t
print("Log loss (CV) on the VotingClassifier :", log_loss(cv_y, vclf.predict_proba(cv_x_or
print("Log loss (test) on the VotingClassifier :", log_loss(test_y, vclf.predict_proba(tes
print("Number of missclassified point :", np.count_nonzero((vclf.predict(test_x_onehotCodi
plot_confusion_matrix(test_y=test_y, predict_y=vclf.predict(test_x_onehotCoding))
```

```
Log loss (train) on the VotingClassifier : 0.8395097007111234
Log loss (CV) on the VotingClassifier : 1.2044408311533519
Log loss (test) on the VotingClassifier : 1.151450222449073
Number of missclassified point : 0.3548872180451128
-------------------- Confusion matrix --------------------
```




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




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


Original Class
Predicted Class
1 0.640 0.009 0.000 0.123 0.123 0.018 0.088 0.000 0.000
2 0.033 0.407 0.000 0.033 0.044 0.055 0.429 0.000 0.000
3 0.056 0.000 0.278 0.389 0.000 0.000 0.278 0.000 0.000
4 0.182 0.015 0.015 0.693 0.029 0.007 0.051 0.000 0.007
5 0.146 0.021 0.042 0.167 0.292 0.021 0.312 0.000 0.000
6 0.109 0.018 0.000 0.055 0.055 0.709 0.055 0.000 0.000
7 0.010 0.110 0.016 0.005 0.000 0.016 0.838 0.000 0.005
8 0.000 0.000 0.000 0.000 0.000 0.250 0.250 0.250 0.250
9 0.143 0.000 0.000 0.000 0.000 0.000 0.143 0.000 0.714
1 2 3 4 5 6 7 8 9
0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.0