

Personalized cancer diagnosis

1. Business Problem

1.1. Description

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

Data: Memorial Sloan Kettering Cancer Center (MSKCC)

Download training_variants.zip and training_text.zip from Kaggle.

Context:

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

Problem statement :

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

1.2. Source/Useful Links

Some articles and reference blogs about the problem statement

1. <https://www.forbes.com/sites/matthewherper/2017/06/03/a-new-cancer-drug-helped-almost-everyone-who-took-it-almost-heres-what-it-teaches-us/#2a44ee2f6b25>

2. <https://www.youtube.com/watch?v=UwbuW7oK8rk>
3. <https://www.youtube.com/watch?v=qxXRKVompl8>

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 contains 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 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 Learning 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 probabilities => 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 [1]: !pip install nltk
!pip install mlxtend
!pip install seaborn
!pip install imblearn
```

Collecting nltk

Downloading <https://files.pythonhosted.org/packages/50/09/3b1755d528ad9156ee7243d52aa5cd2b809ef053a0f31b53d92853dd653a/nltk-3.3.0.zip> (1.4MB)

100% |#####| 1.4MB 7.3MB/s ta 0:00:011
Requirement already satisfied: six in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from nltk) (1.11.0)
Building wheels for collected packages: nltk
Running setup.py bdist_wheel for nltk ... done
Stored in directory: /home/jovyan/.cache/pip/wheels/d1/ab/40/3bceea46922767e42986aef7606a600538ca80de6062dc266c
Successfully built nltk

jupyter 1.0.0 requires qtconsole, which is not installed.
ipywidgets 7.0.3 has requirement widgetsnbextension~=3.0.0, but you'll have widgetsnbextension 3.2.1 which is incompatible.

```

Installing collected packages: nltk
Successfully installed nltk-3.3
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Collecting mlxtend
  Downloading https://files.pythonhosted.org/packages/d0/f9/798cb32550dcbc9e0e3c143dc7144d2631df171423ed143cdb8b38ee2e5e/mlxtend-0.13.0-py2.py3-none-any.whl (1.3MB)
    100% |#####| 1.3MB 11.7MB/s ta 0:00:01
Requirement already satisfied: numpy>=1.10.4 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from mlxtend) (1.12.1)
Requirement already satisfied: scikit-learn>=0.18 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from mlxtend) (0.19.0)
Requirement already satisfied: setuptools in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from mlxtend) (36.4.0)
Requirement already satisfied: scipy>=0.17 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from mlxtend) (0.19.1)
Requirement already satisfied: pandas>=0.17.1 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from mlxtend) (0.20.3)
Requirement already satisfied: matplotlib>=1.5.1 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from mlxtend) (2.1.2)
Requirement already satisfied: python-dateutil>=2 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from pandas>=0.17.1->mlxtend) (2.7.3)
Requirement already satisfied: pytz>=2011k in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from pandas>=0.17.1->mlxtend) (2018.4)
Requirement already satisfied: six>=1.10 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from matplotlib>=1.5.1->mlxtend) (1.11.0)
Requirement already satisfied: cyclor>=0.10 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from matplotlib>=1.5.1->mlxtend) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from matplotlib>=1.5.1->mlxtend) (2.2.0)
jupyter 1.0.0 requires qtconsole, which is not installed.
ipywidgets 7.0.3 has requirement widgetsnbextension~=3.0.0, but you'll have widgetsnbextension 3.2.1 which is incompatible.
Installing collected packages: mlxtend
Successfully installed mlxtend-0.13.0
You are using pip version 10.0.1, however version 18.0 is available.

```

```

You should consider upgrading via the 'pip install --upgrade pip' command.
Collecting seaborn
  Downloading https://files.pythonhosted.org/packages/a8/76/220ba4420459d9c4c9c9587c6ce607bf56c25b3d3d2de62056efe482dadcd/seaborn-0.9.0-py3-none-any.whl (208kB)
    100% |#####| 215kB 9.6MB/s eta 0:00:01
Requirement already satisfied: pandas>=0.15.2 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from seaborn) (0.20.3)
Requirement already satisfied: scipy>=0.14.0 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from seaborn) (0.19.1)
Requirement already satisfied: matplotlib>=1.4.3 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from seaborn) (2.1.2)
Requirement already satisfied: numpy>=1.9.3 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from seaborn) (1.12.1)
Requirement already satisfied: python-dateutil>=2 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from pandas>=0.15.2->seaborn) (2.7.3)
Requirement already satisfied: pytz>=2011k in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from pandas>=0.15.2->seaborn) (2018.4)
Requirement already satisfied: six>=1.10 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from matplotlib>=1.4.3->seaborn) (1.11.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from matplotlib>=1.4.3->seaborn) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from matplotlib>=1.4.3->seaborn) (2.2.0)
jupyter 1.0.0 requires qtconsole, which is not installed.
ipywidgets 7.0.3 has requirement widgetsnbextension~=3.0.0, but you'll have widgetsnbextension 3.2.1 which is incompatible.
Installing collected packages: seaborn
Successfully installed seaborn-0.9.0
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Collecting imblearn
  Downloading https://files.pythonhosted.org/packages/81/a7/4179e6ebfd654bd0eac0b9c06125b8b4c96a9d0a8ff9e9507eb2a26d2d7e/imblearn-0.0-py2.py3-none-any.whl
Collecting imbalanced-learn (from imblearn)

```

Downloading https://files.pythonhosted.org/packages/80/a4/900463a3c0af082aed9c5a43f4ec317a9469710c5ef80496c9abc26ed0ca/imbalanced_learn-0.3.3-py3-none-any.whl (144kB)

100% |#####| 153kB 11.0MB/s ta 0:00:01
Requirement already satisfied: scikit-learn in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from imbalanced-learn->imblearn) (0.19.0)
Requirement already satisfied: numpy in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from imbalanced-learn->imblearn) (1.12.1)
Requirement already satisfied: scipy in /opt/conda/envs/py3.6/lib/python3.6/site-packages (from imbalanced-learn->imblearn) (0.19.1)
jupyter 1.0.0 requires qtconsole, which is not installed.
ipywidgets 7.0.3 has requirement widgetsnbextension~=3.0.0, but you'll have widgetsnbextension 3.2.1 which is incompatible.
Installing collected packages: imbalanced-learn, imblearn
Successfully installed imbalanced-learn-0.3.3 imblearn-0.0
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.

```
In [2]: import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from imblearn.over_sampling import SMOTE
```



```

from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.cross_validation import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier
warnings.filterwarnings("ignore")

from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression

```

```

[nltk_data] Downloading package stopwords to /home/jovyan/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.

```

```

/opt/conda/envs/py3.6/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)

```

3.1. Reading Data

3.1.1. Reading Gene and Variation Data

```

In [3]: 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[3]:

	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 [4]: # note the separator in this file
data_text = pd.read_csv("training_text", sep="\\|\\|", engine="python", names
=["ID", "TEXT"], skiprows=1)
```

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

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

Out[4]:

	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 [5]: # 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 [6]: #text processing stage.
start_time = time.clock()
for index, row in data_text.iterrows():
    nlp_preprocessing(row['TEXT'], index, 'TEXT')
print('Time took for preprocessing the text :', time.clock() - start_time, "seconds")

```

Time took for preprocessing the text : 127.89899199999999 seconds

```

In [7]: #merging both gene_variations and text data based on ID
result = pd.merge(data, data_text, on='ID', how='left')
result.head()

```

Out[7]:

	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...

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 [8]: 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 o
f output variable 'y_true' [stratify=y_true]
X_train, test_df, y_train, y_test = train_test_split(result, y_true, st
ratify=y_true, test_size=0.2)
# split the train data into train and cross validation by maintaining s
ame distribution of output variable 'y_train' [stratify=y_train]
train_df, cv_df, y_train, y_cv = train_test_split(X_train, y_train, str
atify=y_train, test_size=0.2)
```

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

```
In [9]: 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 [10]: # it returns a dict, keys as class labels and values as the number of d
ata points in that class
train_class_distribution = train_df['Class'].value_counts().sortlevel()
test_class_distribution = test_df['Class'].value_counts().sortlevel()
cv_class_distribution = cv_df['Class'].value_counts().sortlevel()

my_colors = 'rgbkymc'
```

```

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

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

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

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

print('-'*80)
my_colors = 'rgbkymc'

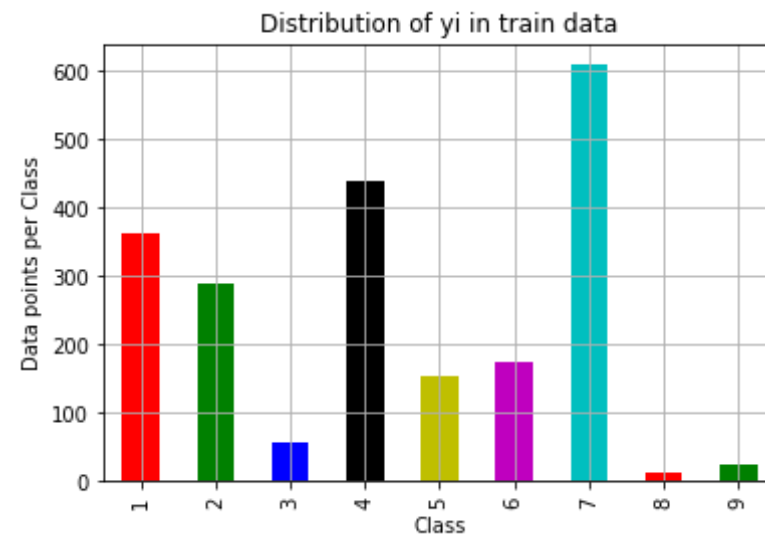
```

```

cv_class_distribution.plot(kind='bar', color=my_colors)
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in cross validation data')
plt.grid()
plt.show()

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

```

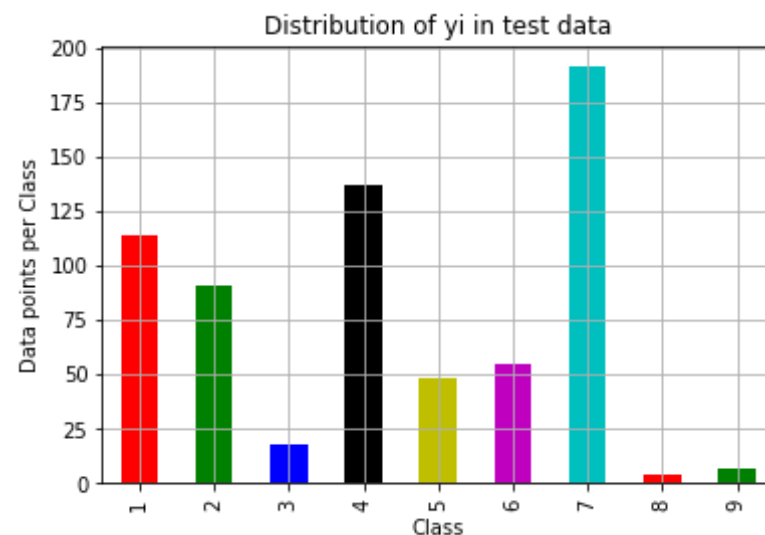


```

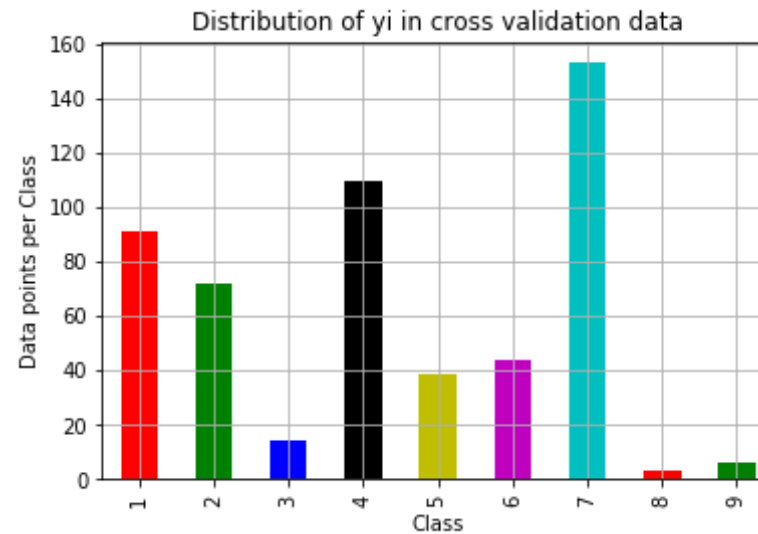
Number of data points in class 7 : 609 ( 28.672 %)
Number of data points in class 4 : 439 ( 20.669 %)
Number of data points in class 1 : 363 ( 17.09 %)
Number of data points in class 2 : 289 ( 13.606 %)
Number of data points in class 6 : 176 ( 8.286 %)
Number of data points in class 5 : 155 ( 7.298 %)

```

Number of data points in class 3 : 57 (2.684 %)
Number of data points in class 9 : 24 (1.13 %)
Number of data points in class 8 : 12 (0.565 %)



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



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

3.2 Prediction using a 'Random' Model

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

```
In [11]: # This function plots the confusion matrices given y_i, y_i_hat.
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    # C = 9,9 matrix, each cell (i,j) represents number of points of cl
```

```

ass i are predicted class j

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

# C = [[1, 2],
#      [3, 4]]
# C.T = [[1, 3],
#        [2, 4]]
# C.sum(axis = 1) axis=0 corresponsds to columns and axis=1 correspo
nds to rows in two dimensional array
# C.sum(axix =1) = [[3, 7]]
# ((C.T)/(C.sum(axis=1))) = [[1/3, 3/7]
#                             [2/3, 4/7]]

# ((C.T)/(C.sum(axis=1))).T = [[1/3, 2/3]
#                               [3/7, 4/7]]
# sum of row elements = 1

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

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

```

```

plt.show()

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

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

```

```

In [12]: # we need to generate 9 numbers and the sum of numbers should be 1
# one solution is to generate 9 numbers and divide each of the numbers
by their sum
# ref: https://stackoverflow.com/a/18662466/4084039
test_data_len = test_df.shape[0]
cv_data_len = cv_df.shape[0]

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

# Test-Set error.
#we create a output array that has exactly same as the test data
test_predicted_y = np.zeros((test_data_len,9))
for i in range(test_data_len):

```

```

rand_probs = np.random.rand(1,9)
test_predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Test Data using Random Model",log_loss(y_test,test_p
redicted_y, eps=1e-15))

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

```

Log loss on Cross Validation Data using Random Model 2.45255804431

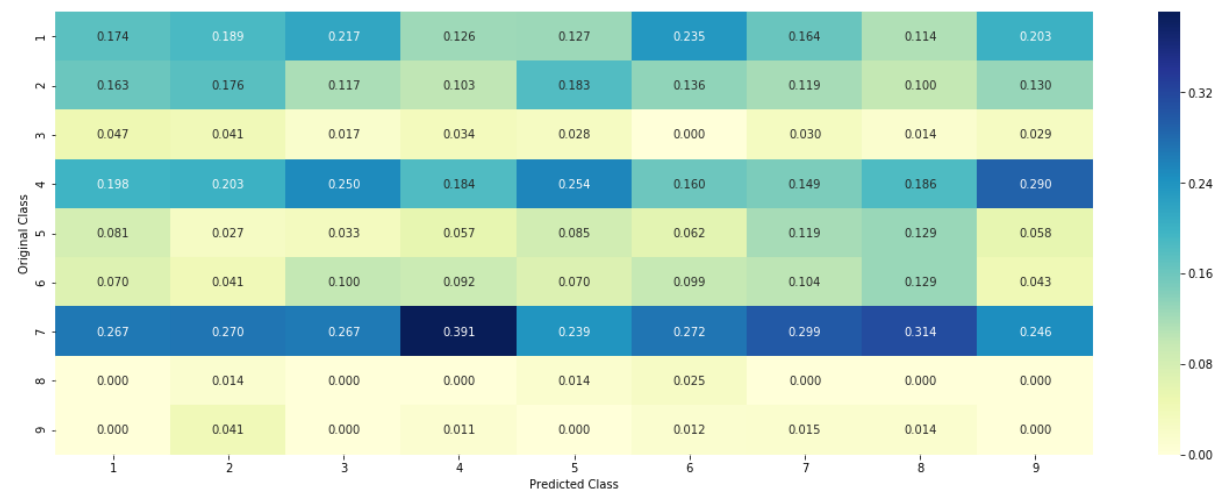
Log loss on Test Data using Random Model 2.48373944254

----- Confusion matrix -----

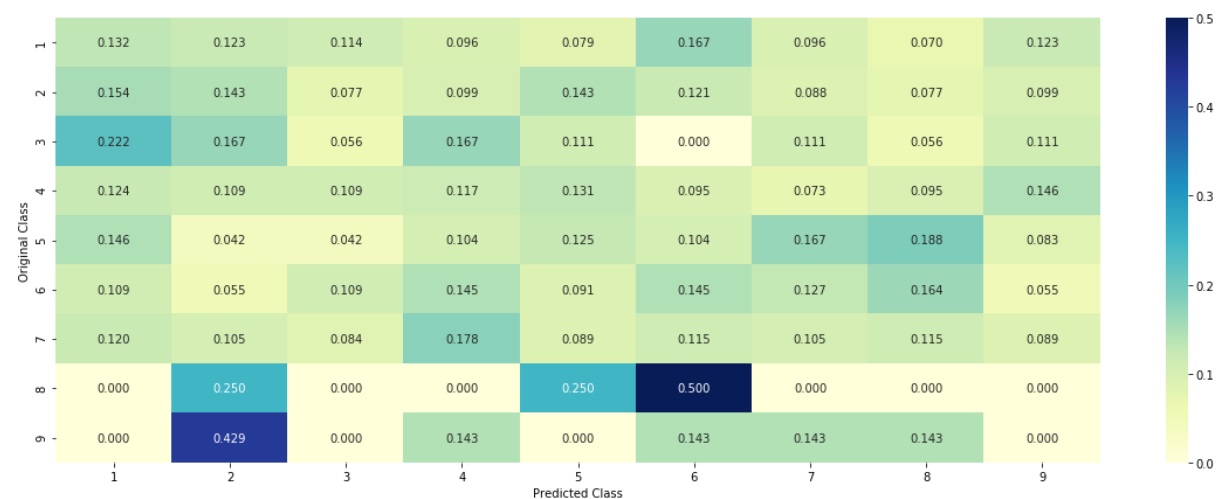


----- Precision matrix (Column Sum=1) -----

--



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



3.3 Univariate Analysis

```
In [13]: # 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 occurrences of given feature in train data dataframe
# build a vector (1*9) , the first element = (number of times it occurred in class1 + 10*alpha / number of time it occurred in total data+90*alpha)
# gv_dict is like a look up table, for every gene it store a (1*9) representation of it
# for a value of feature in df:
# if it is in train data:
# we add the vector that was stored in 'gv_dict' look up table to 'gv_fea'
# if it is not there is train:
# we add [1/9, 1/9, 1/9, 1/9,1/9, 1/9, 1/9, 1/9, 1/9] to 'gv_fea'
# return 'gv_fea'
# -----

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

```

```

# Amplification 43
# Fusions 22
# Overexpression 3
# E17K 3
# Q61L 3
# S222D 2
# P130S 2
# ...
# }
value_count = train_df[feature].value_counts()

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

# denominator will contain the number of time that particular feature occurred in whole data
for i, denominator in value_count.items():
    # vec will contain (p(yi==1/Gi) probability of gene/variation belongs to particular class
    # vec is 9 dimensional 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 ti

```

```

me that particular feature occurred in whole data
    vec.append((cls_cnt.shape[0] + alpha*10)/ (denominator + 90
*alpha))

    # we are adding the gene/variation to the dict as key and vec a
s 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, 0.068181
8181818177, 0.13636363636363635, 0.25, 0.19318181818181818, 0.0378787
8787878788, 0.03787878787878788, 0.03787878787878788],
    # 'TP53': [0.32142857142857145, 0.061224489795918366, 0.061224
489795918366, 0.27040816326530615, 0.061224489795918366, 0.066326530612
244902, 0.051020408163265307, 0.051020408163265307, 0.05612244897959183
7],
    # 'EGFR': [0.056818181818181816, 0.21590909090909091, 0.0625,
0.068181818181818177, 0.068181818181818177, 0.0625, 0.3465909090909091
2, 0.0625, 0.056818181818181816],
    # 'BRCA2': [0.13333333333333333, 0.060606060606060608, 0.06060
60606060608, 0.078787878787878782, 0.1393939393939394, 0.345454545454
54546, 0.060606060606060608, 0.060606060606060608, 0.060606060606060
8],
    # 'PTEN': [0.069182389937106917, 0.062893081761006289, 0.06918
2389937106917, 0.46540880503144655, 0.075471698113207544, 0.06289308176
1006289, 0.069182389937106917, 0.062893081761006289, 0.0628930817610062
89],
    # 'KIT': [0.066225165562913912, 0.25165562913907286, 0.0728476
82119205295, 0.072847682119205295, 0.066225165562913912, 0.066225165562
913912, 0.27152317880794702, 0.066225165562913912, 0.06622516556291391
2],
    # 'BRAF': [0.066666666666666666, 0.17999999999999999, 0.073333
33333333334, 0.07333333333333334, 0.09333333333333338, 0.08000000000
0000002, 0.29999999999999999, 0.066666666666666666, 0.066666666666666
6],
    # ...

```



```

#     }
gv_dict = get_gv_fea_dict(alpha, feature, df)
# value_count is similar in get_gv_fea_dict
value_count = train_df[feature].value_counts()

# gv_fea: Gene_variation feature, it will contain the feature for each feature value in the data
gv_fea = []
# for every feature values in the given data frame we will check if it is there in the train data then we will add the feature to gv_fea
# if not we will add [1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9] to gv_fea
a
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 calculate the probability of a feature belongs to any particular class, we apply laplace smoothing

- $(\text{numerator} + 10 \cdot \alpha) / (\text{denominator} + 90 \cdot \alpha)$

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 [14]: unique_genes = train_df['Gene'].value_counts()
print('Number of Unique Genes :', unique_genes.shape[0])

```

```
# the top 10 genes that occurred most  
print(unique_genes.head(10))
```

Number of Unique Genes : 235

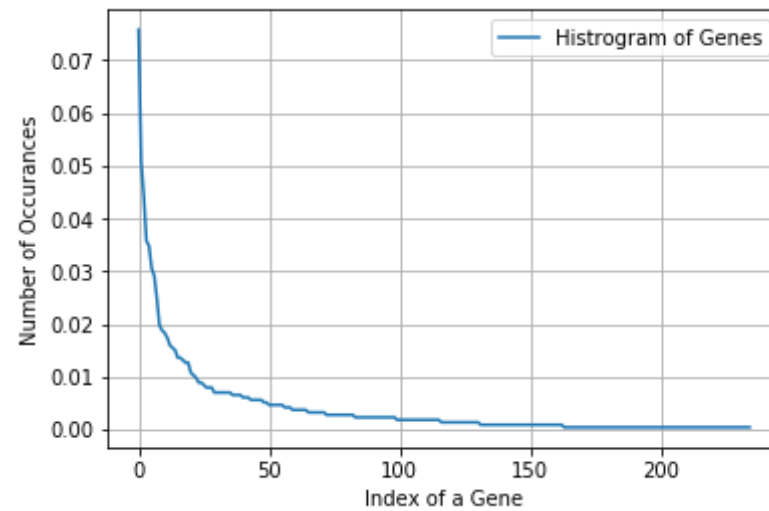
BRCA1	161
TP53	108
PTEN	94
BRCA2	76
EGFR	74
KIT	65
BRAF	62
ALK	53
ERBB2	42
PDGFRA	40

Name: Gene, dtype: int64

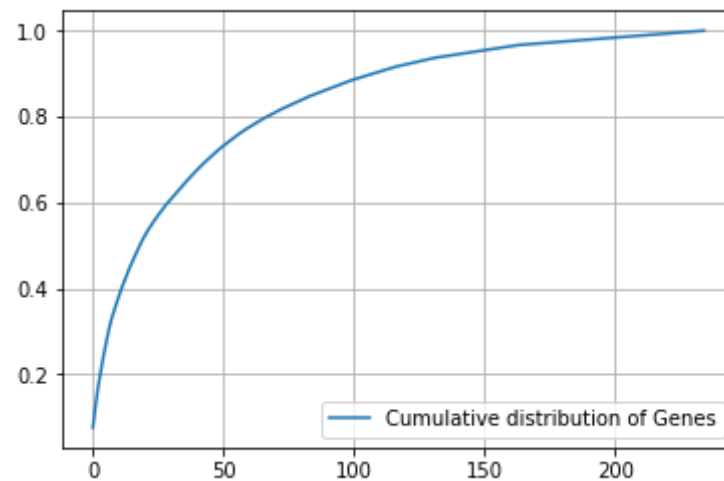
```
In [15]: print("Ans: There are", unique_genes.shape[0], "different categories of  
genes in the train data, and they are distributed as follows",)
```

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

```
In [16]: s = sum(unique_genes.values);  
h = unique_genes.values/s;  
plt.plot(h, label="Histogram of Genes")  
plt.xlabel('Index of a Gene')  
plt.ylabel('Number of Occurances')  
plt.legend()  
plt.grid()  
plt.show()
```



```
In [17]: 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 [18]: #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 [19]: print("train_gene_feature_responseCoding is converted feature using response coding method. The shape of gene feature:", train_gene_feature_responseCoding.shape)
```

train_gene_feature_responseCoding is converted feature using response coding method. The shape of gene feature: (2124, 9)

```
In [20]: # one-hot encoding of Gene feature.
gene_vectorizer = TfidfVectorizer(max_features = 2000)
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 [21]: train_df['Gene'].head()
```

```
Out[21]: 1347      AKT1
        656      CDKN2A
        2244     PTEN
        1279     HRAS
        3220     NTRK1
        Name: Gene, dtype: object
```

```
In [22]: gene_vectorizer.get_feature_names()
```

```
Out[22]: ['abl1',
          'acvr1',
          'ago2',
          'akt1',
          'akt2',
          'akt3',
          'alk',
          'apc',
          'ar',
          'araf',
          'arid1a',
          'arid1b',
          'arid2',
          'arid5b',
          'asxl1',
          'asxl2',
          'atm',
          'atr',
          'atrx',
          'aurka',
          'aurkb',
          'axin1',
          'axl',
          'b2m',
```

```
'bap1',  
'bard1',  
'bcl10',  
'bcl2',  
'bcl2l11',  
'bcor',  
'braf',  
'brca1',  
'brca2',  
'brip1',  
'btk',  
'card11',  
'carm1',  
'casp8',  
'cbl',  
'ccnd1',  
'ccnd3',  
'ccne1',  
'cdh1',  
'cdk12',  
'cdk4',  
'cdk6',  
'cdkn1a',  
'cdkn1b',  
'cdkn2a',  
'cdkn2b',  
'chek2',  
'cic',  
'crebbp',  
'ctcf',  
'ctla4',  
'ctnnb1',  
'ddr2',  
'dicer1',  
'dnmt3a',  
'egfr',  
'eif1ax',  
'elf3',  
'ep300',
```

'epas1',
'erbb2',
'erbb3',
'erbb4',
'ercc2',
'ercc4',
'erg',
'errfi1',
'esr1',
'etv1',
'etv6',
'ewsr1',
'ezh2',
'fam58a',
'fanca',
'fancc',
'fat1',
'fbxw7',
'fgf19',
'fgf3',
'fgf4',
'fgfr1',
'fgfr2',
'fgfr3',
'fgfr4',
'flt1',
'flt3',
'foxa1',
'foxl2',
'foxo1',
'foxp1',
'gata3',
'gli1',
'gna11',
'gnaq',
'gnas',
'h3f3a',
'hist1h1c',
'hla',

'hnf1a',
'hras',
'idh1',
'idh2',
'igf1r',
'ikbke',
'ikzf1',
'il7r',
'jak1',
'jak2',
'jun',
'kdm5c',
'kdm6a',
'kdr',
'keap1',
'kit',
'kmt2a',
'kmt2c',
'kmt2d',
'knstrn',
'kras',
'lats1',
'lats2',
'map2k1',
'map2k2',
'map2k4',
'map3k1',
'mapk1',
'mdm4',
'med12',
'mef2b',
'men1',
'met',
'mga',
'mlh1',
'mpl',
'msh2',
'msh6',
'mtor',


```
'myc',  
'mycn',  
'myd88',  
'myod1',  
'nf1',  
'nf2',  
'nfe2l2',  
'nfkb1a',  
'nkx2',  
'notch1',  
'notch2',  
'nras',  
'nsd1',  
'ntrk1',  
'ntrk2',  
'ntrk3',  
'nup93',  
'pax8',  
'pbrm1',  
'pdgfra',  
'pdgfrb',  
'pik3ca',  
'pik3cb',  
'pik3cd',  
'pik3r1',  
'pik3r2',  
'pim1',  
'pms1',  
'pms2',  
'pole',  
'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',  
'rnf43',  
'ros1',  
'rras2',  
'runx1',  
'rxra',  
'sdhb',  
'sdhc',  
'setd2',  
'sf3b1',  
'shq1',  
'smad2',  
'smad3',  
'smad4',  
'smarca4',  
'smarcb1',  
'smo',  
'sos1',  
'sox9',  
'spop',  
'src',  
'srsf2',  
'stag2',  
'stat3',
```

```
'stk11',  
'tert',  
'tet1',  
'tet2',  
'tgfbr1',  
'tgfbr2',  
'tmprss2',  
'tp53',  
'tp53bp1',  
'tsc1',  
'tsc2',  
'u2af1',  
'vegfa',  
'vhl',  
'xpo1',  
'yap1']
```

```
In [23]: print("train_gene_feature_onehotCoding is converted feature using one-hot  
encoding method. The shape of gene feature:", train_gene_feature_one  
hotCoding.shape)
```

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

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

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

```
In [24]: alpha = [10 ** x for x in range(-5, 1)] # hyperparam for SGD classifier.  
  
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.SGDClassifier.html  
# -----  
# default parameters
```

```

# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
5, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
arning_rate='optimal', eta0=0.0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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

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

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

```

```

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

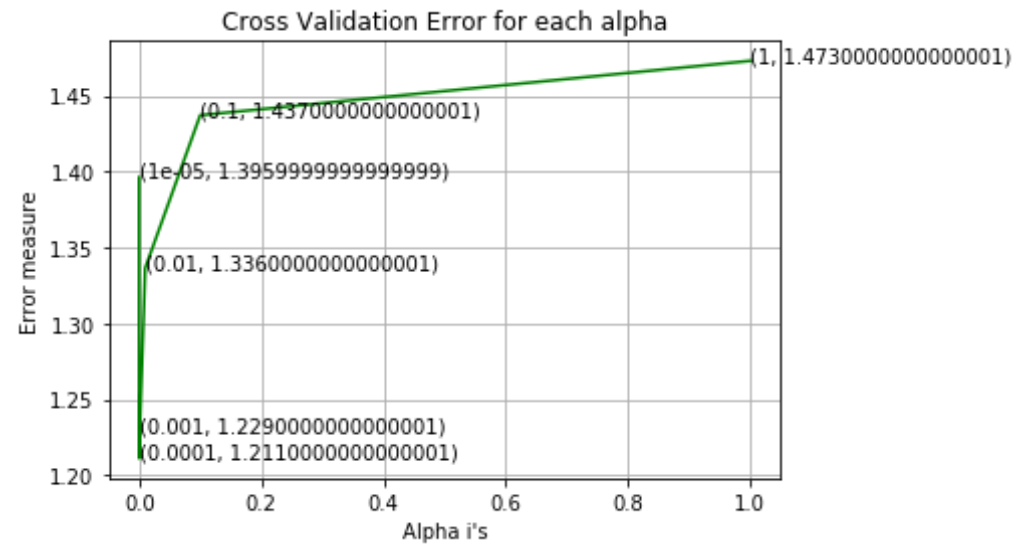
predict_y = sig_clf.predict_proba(train_gene_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log
loss is:", log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(cv_gene_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross vali
dation log loss is:", log_loss(y_cv, predict_y, labels=clf.classes_, eps
=1e-15))
predict_y = sig_clf.predict_proba(test_gene_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log l
oss is:", log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))

```

```

For values of alpha = 1e-05 The log loss is: 1.39623957674
For values of alpha = 0.0001 The log loss is: 1.21105620962
For values of alpha = 0.001 The log loss is: 1.22857516634
For values of alpha = 0.01 The log loss is: 1.33643673161
For values of alpha = 0.1 The log loss is: 1.43706915916
For values of alpha = 1 The log loss is: 1.47290584338

```



For values of best alpha = 0.0001 The train log loss is: 1.03449612543
 For values of best alpha = 0.0001 The cross validation log loss is: 1.21105620962
 For values of best alpha = 0.0001 The test log loss is: 1.19689210348

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

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

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

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

Q6. How many data points in Test and CV datasets are covered by the 23 5 genes in train dataset?

Ans

1. In test data 650 out of 665 : 97.74436090225564

2. In cross validation data 513 out of 532 : 96.42857142857143

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

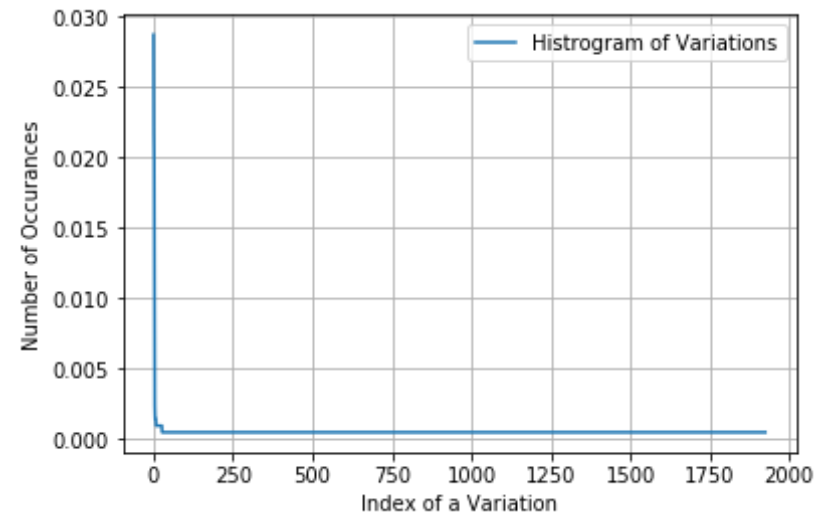
```
Number of Unique Variations : 1926
Truncating_Mutations      61
Deletion                  46
Amplification              42
Fusions                   24
Overexpression             5
G12V                      3
E17K                      3
Q61R                      3
Q209L                     2
G35R                      2
Name: Variation, dtype: int64
```

```
In [27]: print("Ans: There are", unique_variations.shape[0] ,"different categories of variations in the train data, and they are distributed as follows")
```

```
,)
```

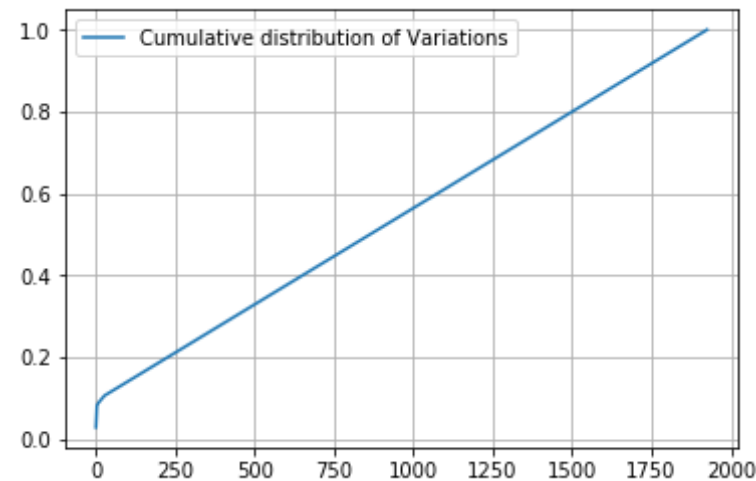
Ans: There are 1926 different categories of variations in the train data, and they are distributed as follows

```
In [28]: s = sum(unique_variations.values);  
h = unique_variations.values/s;  
plt.plot(h, label="Histogram of Variations")  
plt.xlabel('Index of a Variation')  
plt.ylabel('Number of Occurances')  
plt.legend()  
plt.grid()  
plt.show()
```



```
In [29]: c = np.cumsum(h)  
print(c)  
plt.plot(c, label='Cumulative distribution of Variations')  
plt.grid()  
plt.legend()  
plt.show()
```

```
[ 0.0287194  0.05037665  0.07015066 ...,  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 [30]: # alpha is used for laplace smoothing
alpha = 1
# train gene feature
train_variation_feature_responseCoding = np.array(get_gv_feature(alpha,
"Variation", train_df))
# test gene feature
test_variation_feature_responseCoding = np.array(get_gv_feature(alpha,
"Variation", test_df))
# cross validation gene feature
```

```
cv_variation_feature_responseCoding = np.array(get_gv_feature(alpha, "Variation", cv_df))
```

```
In [31]: print("train_variation_feature_responseCoding is a converted feature using the response coding method. The shape of Variation feature:", train_variation_feature_responseCoding.shape)
```

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

```
In [32]: # one-hot encoding of variation feature.
variation_vectorizer = TfidfVectorizer(max_features = 2000)
train_variation_feature_onehotCoding = variation_vectorizer.fit_transform(train_df['Variation'])
test_variation_feature_onehotCoding = variation_vectorizer.transform(test_df['Variation'])
cv_variation_feature_onehotCoding = variation_vectorizer.transform(cv_df['Variation'])
```

```
In [33]: print("train_variation_feature_onehotEncoded is converted feature using the one-hot encoding method. The shape of Variation feature:", train_variation_feature_onehotCoding.shape)
```

train_variation_feature_onehotEncoded is converted feature using the one-hot encoding method. The shape of Variation feature: (2124, 1954)

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

Let's build a model just like the earlier!

```
In [34]: alpha = [10 ** x for x in range(-5, 1)]

# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
# -----
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
```

```

5, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
arning_rate='optimal', eta0=0.0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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

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

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

```

```

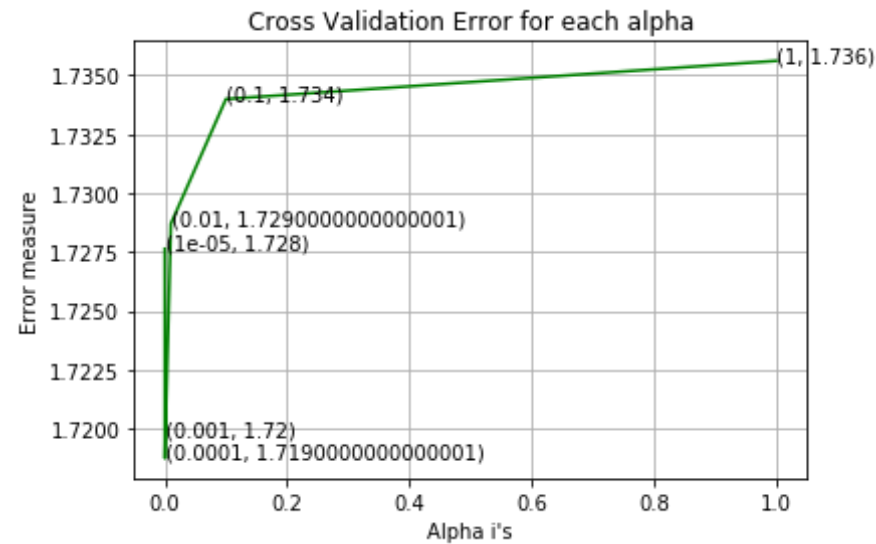
plt.ylabel("Error measure")
plt.show()

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

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

```

For values of alpha = 1e-05 The log loss is: 1.72762922547
For values of alpha = 0.0001 The log loss is: 1.71873258957
For values of alpha = 0.001 The log loss is: 1.71973111568
For values of alpha = 0.01 The log loss is: 1.72869413723
For values of alpha = 0.1 The log loss is: 1.73398742525
For values of alpha = 1 The log loss is: 1.73561116711



For values of best alpha = 0.0001 The train log loss is: 0.714280266807

For values of best alpha = 0.0001 The cross validation log loss is: 1.71873258957

For values of best alpha = 0.0001 The test log loss is: 1.6999428338

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

Q12. How many data points are covered by total 1926 genes in test and cross validation data sets?

Ans

1. In test data 63 out of 665 : 9.473684210526317
2. In cross validation data 56 out of 532 : 10.526315789473683

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 predicting y_i ?
5. Is the text feature stable across train, test and CV datasets?

```
In [36]: # 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 [37]: 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,0)+90)))
        text_feature_responseCoding[row_index][i] = math.exp(sum_prob/len(row['TEXT'].split()))
        row_index += 1
    return text_feature_responseCoding

```

```

In [38]: # building a CountVectorizer with all the words that occurred minimum 3
         times in train data
text_vectorizer = TfidfVectorizer(max_features = 2000, 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 of features) vector
train_text_fea_counts = train_text_feature_onehotCoding.sum(axis=0).A1

# zip(list(text_features), text_fea_counts) will zip a word with its number
# of times it occurred
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 : 2000

```

In [39]: dict_list = []
         # dict_list = [] contains 9 dictionaries 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 build 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 [40]: #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 [41]: # https://stackoverflow.com/a/16202486
# we convert each row values such that they sum to 1
train_text_feature_responseCoding = (train_text_feature_responseCoding.T/train_text_feature_responseCoding.sum(axis=1)).T
test_text_feature_responseCoding = (test_text_feature_responseCoding.T/test_text_feature_responseCoding.sum(axis=1)).T
cv_text_feature_responseCoding = (cv_text_feature_responseCoding.T/cv_text_feature_responseCoding.sum(axis=1)).T
```

```
In [42]: # 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 [43]: #https://stackoverflow.com/a/2258273/4084039
sorted_text_fea_dict = dict(sorted(text_fea_dict.items(), key=lambda x:
x[1] , reverse=True))
sorted_text_occur = np.array(list(sorted_text_fea_dict.values()))
```

```
In [45]: # Train a Logistic regression+Calibration model using text features whi
cha re on-hot encoded
alpha = [10 ** x for x in range(-5, 1)]

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

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

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

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

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

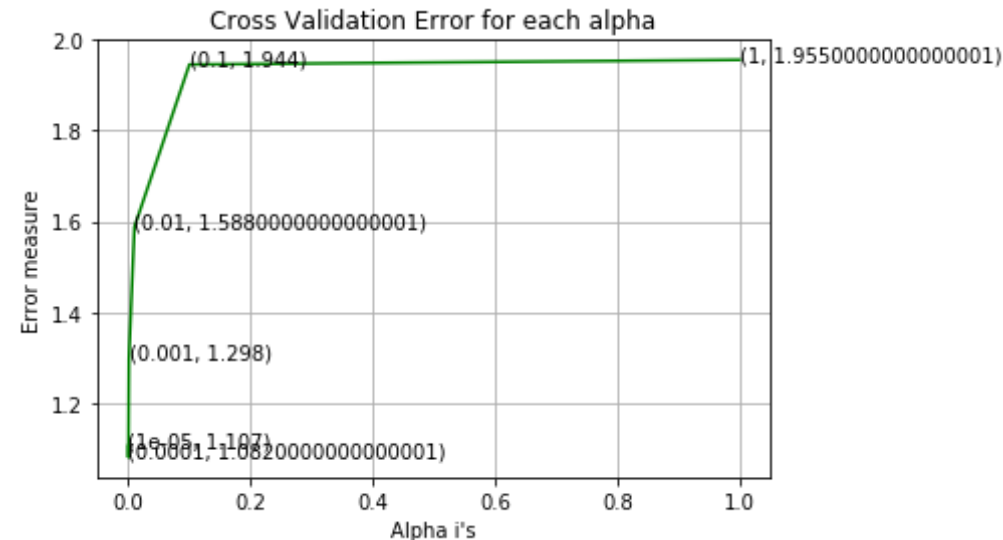
best_alpha = np.argmin(cv_log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log',
random_state=42)
clf.fit(train_text_feature_onehotCoding, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_text_feature_onehotCoding, y_train)

predict_y = sig_clf.predict_proba(train_text_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log
loss is:", log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(cv_text_feature_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross vali
dation log loss is:", log_loss(y_cv, predict_y, labels=clf.classes_, eps
=1e-15))
predict_y = sig_clf.predict_proba(test_text_feature_onehotCoding)

```

```
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:", log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

For values of alpha = 1e-05 The log loss is: 1.10688313427
 For values of alpha = 0.0001 The log loss is: 1.08179082821
 For values of alpha = 0.001 The log loss is: 1.29777661932
 For values of alpha = 0.01 The log loss is: 1.58772973135
 For values of alpha = 0.1 The log loss is: 1.9443622974
 For values of alpha = 1 The log loss is: 1.95462153649



For values of best alpha = 0.0001 The train log loss is: 0.72385618962
 2
 For values of best alpha = 0.0001 The cross validation log loss is: 1.08179082821
 For values of best alpha = 0.0001 The test log loss is: 1.10145361293

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

Ans. Yes, it seems like!

In [46]: `def get_intersec_text(df):`

```

df_text_vec = TfidfVectorizer(max_features = 2000, 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 [47]: 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")

```

93.55 % of word of test data appeared in train data
92.85 % of word of Cross Validation appeared in train data

4. Machine Learning Models

```

In [48]: #Data preparation for ML models.

#Misc. fonctionns for ML models

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

    # for calculating log_loss we willl provide the array of probabilities belongs to each class

```

```

print("Log loss :", log_loss(test_y, sig_clf.predict_proba(test_x)))
# calculating the number of data points that are misclassified
print("Number of mis-classified points :", np.count_nonzero((pred_y
- test_y))/test_y.shape[0])
plot_confusion_matrix(test_y, pred_y)

```

```

In [49]: 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 [50]: # 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 = TfidfVectorizer(max_features = 2000)
            var_count_vec = TfidfVectorizer(max_features = 2000)
            text_count_vec = TfidfVectorizer(max_features = 2000, 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, yes_no))

```

```

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(word,yes_no))
    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,yes_no))

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

```

Stacking the three types of features

```

In [51]: # merging gene, variance and text features

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

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

```

```

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

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

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

train_gene_var_responseCoding = np.hstack((train_gene_feature_responseC
oding, train_variation_feature_responseCoding))
test_gene_var_responseCoding = np.hstack((test_gene_feature_responseCod
ing, test_variation_feature_responseCoding))
cv_gene_var_responseCoding = np.hstack((cv_gene_feature_responseCoding,
cv_variation_feature_responseCoding))

train_x_responseCoding = np.hstack((train_gene_var_responseCoding, trai
n_text_feature_responseCoding))
test_x_responseCoding = np.hstack((test_gene_var_responseCoding, test_t
ext_feature_responseCoding))
cv_x_responseCoding = np.hstack((cv_gene_var_responseCoding, cv_text_fe
ature_responseCoding))

```

```

In [52]: print("One hot encoding features :")
print("(number of data points * number of features) in train data = ",
train_x_onehotCoding.shape)
print("(number of data points * number of features) in test data = ", t
est_x_onehotCoding.shape)
print("(number of data points * number of features) in cross validation
data = ", cv_x_onehotCoding.shape)

```

```

One hot encoding features :
(number of data points * number of features) in train data = (2124, 41
89)

```

```
(number of data points * number of features) in test data = (665, 4189)
(number of data points * number of features) in cross validation data = (532, 4189)
```

```
In [53]: print(" Response encoding features :")
print("(number of data points * number of features) in train data = ",
train_x_responseCoding.shape)
print("(number of data points * number of features) in test data = ", test_x_responseCoding.shape)
print("(number of data points * number of features) in cross validation data = ", cv_x_responseCoding.shape)
```

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

# some of methods of MultinomialNB()
# fit(X, y[, sample_weight])    Fit Naive Bayes classifier according to
```



```

X, y
# predict(X)    Perform classification on an array of test vectors X.
# predict_log_proba(X) Return log-probability estimates for the test v
ector X.
# -----
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/naive-bayes-algorithm-1/
# -----

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

alpha = [0.00001, 0.0001, 0.001, 0.1, 1, 10, 100,1000]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = MultinomialNB(alpha=i)
    clf.fit(train_x_onehotCoding, train_y)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_x_onehotCoding, train_y)
    sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
    cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.

```

```

classes_, eps=1e-15))
    # to avoid rounding error while multiplying probabilities we use log
    -probability estimates
    print("Log Loss :", log_loss(cv_y, sig_clf_probs))

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

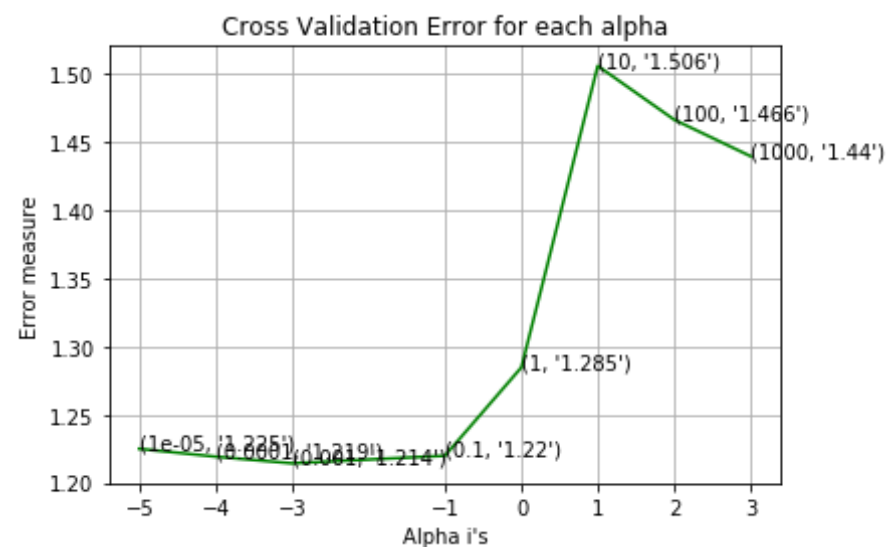
best_alpha = np.argmin(cv_log_error_array)
clf = MultinomialNB(alpha=alpha[best_alpha])
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log
loss is:", log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross vali
dation log loss is:", log_loss(y_cv, predict_y, labels=clf.classes_, eps
=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log l
oss is:", log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))

for alpha = 1e-05
Log Loss : 1.22493168291
for alpha = 0.0001

```

Log Loss : 1.2191648933
 for alpha = 0.001
 Log Loss : 1.21426267687
 for alpha = 0.1
 Log Loss : 1.21981238964
 for alpha = 1
 Log Loss : 1.28460551017
 for alpha = 10
 Log Loss : 1.50591858158
 for alpha = 100
 Log Loss : 1.46639724922
 for alpha = 1000
 Log Loss : 1.43957985142



For values of best alpha = 0.001 The train log loss is: 0.53996153846
 For values of best alpha = 0.001 The cross validation log loss is: 1.21426267687
 For values of best alpha = 0.001 The test log loss is: 1.21680806607

4.1.1.2. Testing the model with best hyper paramters

```

In [55]: # find more about Multinomial Naive base function here http://scikit-learn.org/stable/modules/generated/sklearn.naive\_bayes.MultinomialNB.html
# -----
# default paramters
# sklearn.naive_bayes.MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)

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

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

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 probabilities we use log-probability estimates
print("Log Loss :", log_loss(cv_y, sig_clf_probs))
print("Number of missclassified point :", np.count_nonzero((sig_clf.predict(cv_x_onehotCoding) - cv_y))/cv_y.shape[0])
plot_confusion_matrix(cv_y, sig_clf.predict(cv_x_onehotCoding.toarray()))
```

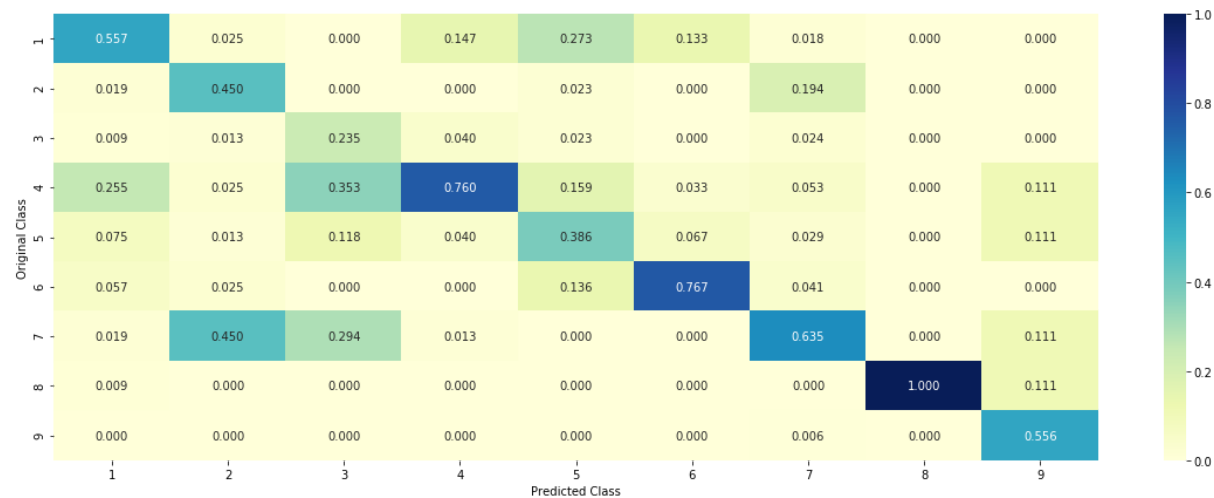
Log Loss : 1.21426267687

Number of missclassified point : 0.41729323308270677

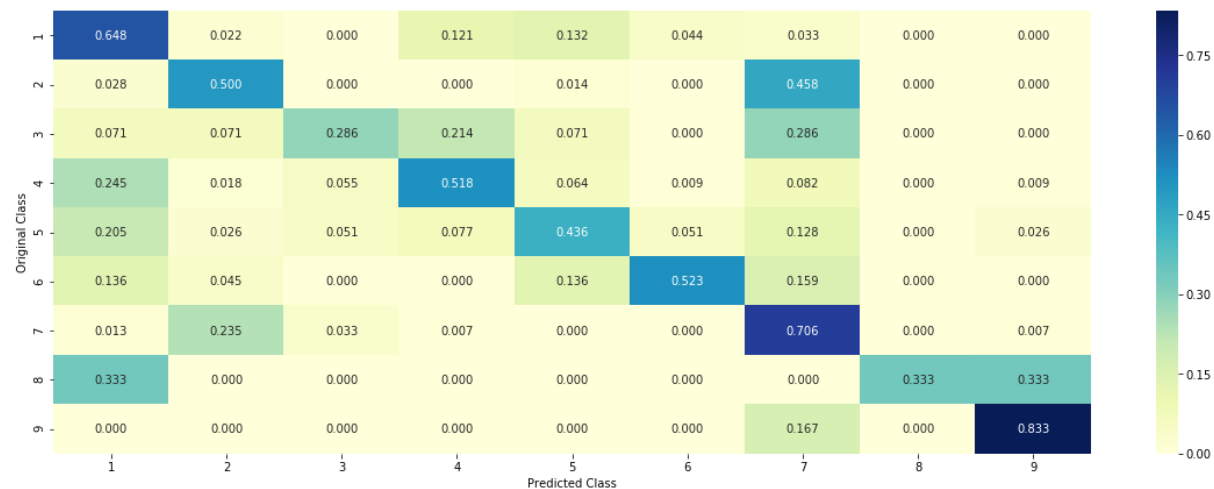
----- Confusion matrix -----



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



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



4.1.1.3. Feature Importance, Correctly classified point

```
In [56]: test_point_index = 1
no_feature = 100
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
```

```

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

```

```

Predicted Class : 1
Predicted Class Probabilities: [[ 0.7032  0.0459  0.0119  0.0751  0.037
1  0.0377  0.0818  0.0038  0.0034]]
Actual Class : 4

```

```

-----
9 Text feature [protein] present in test data point [True]
11 Text feature [one] present in test data point [True]
12 Text feature [type] present in test data point [True]
13 Text feature [dna] present in test data point [True]
14 Text feature [results] present in test data point [True]
15 Text feature [loss] present in test data point [True]
16 Text feature [function] present in test data point [True]
17 Text feature [role] present in test data point [True]
18 Text feature [also] present in test data point [True]
19 Text feature [wild] present in test data point [True]
20 Text feature [two] present in test data point [True]
21 Text feature [control] present in test data point [True]
22 Text feature [using] present in test data point [True]
23 Text feature [therefore] present in test data point [True]
24 Text feature [table] present in test data point [True]
25 Text feature [functions] present in test data point [True]
26 Text feature [however] present in test data point [True]
30 Text feature [expression] present in test data point [True]
31 Text feature [region] present in test data point [True]
32 Text feature [containing] present in test data point [True]
33 Text feature [possible] present in test data point [True]
34 Text feature [result] present in test data point [True]
35 Text feature [shown] present in test data point [True]
36 Text feature [binding] present in test data point [True]
37 Text feature [either] present in test data point [True]

```

38 Text feature [analysis] present in test data point [True]
39 Text feature [reduced] present in test data point [True]
40 Text feature [affect] present in test data point [True]
41 Text feature [human] present in test data point [True]
42 Text feature [used] present in test data point [True]
43 Text feature [specific] present in test data point [True]
44 Text feature [determined] present in test data point [True]
45 Text feature [well] present in test data point [True]
46 Text feature [whether] present in test data point [True]
47 Text feature [present] present in test data point [True]
48 Text feature [may] present in test data point [True]
49 Text feature [large] present in test data point [True]
50 Text feature [involved] present in test data point [True]
51 Text feature [mediated] present in test data point [True]
52 Text feature [effect] present in test data point [True]
53 Text feature [three] present in test data point [True]
54 Text feature [25] present in test data point [True]
55 Text feature [transcriptional] present in test data point [True]
56 Text feature [thus] present in test data point [True]
57 Text feature [previous] present in test data point [True]
58 Text feature [suggest] present in test data point [True]
59 Text feature [discussion] present in test data point [True]
60 Text feature [data] present in test data point [True]
61 Text feature [observed] present in test data point [True]
62 Text feature [fig] present in test data point [True]
63 Text feature [whereas] present in test data point [True]
64 Text feature [sequence] present in test data point [True]
65 Text feature [important] present in test data point [True]
66 Text feature [least] present in test data point [True]
67 Text feature [additional] present in test data point [True]
68 Text feature [several] present in test data point [True]
69 Text feature [conserved] present in test data point [True]
70 Text feature [four] present in test data point [True]
71 Text feature [cell] present in test data point [True]
72 Text feature [gene] present in test data point [True]
73 Text feature [ability] present in test data point [True]
74 Text feature [addition] present in test data point [True]
75 Text feature [critical] present in test data point [True]
76 Text feature [cancer] present in test data point [True]


```

77 Text feature [corresponding] present in test data point [True]
78 Text feature [studies] present in test data point [True]
79 Text feature [based] present in test data point [True]
80 Text feature [described] present in test data point [True]
81 Text feature [similar] present in test data point [True]
82 Text feature [previously] present in test data point [True]
83 Text feature [indicated] present in test data point [True]
84 Text feature [performed] present in test data point [True]
85 Text feature [amino] present in test data point [True]
86 Text feature [although] present in test data point [True]
87 Text feature [analyzed] present in test data point [True]
88 Text feature [specifically] present in test data point [True]
89 Text feature [directly] present in test data point [True]
90 Text feature [compared] present in test data point [True]
91 Text feature [following] present in test data point [True]
92 Text feature [likely] present in test data point [True]
93 Text feature [together] present in test data point [True]
94 Text feature [15] present in test data point [True]
95 Text feature [within] present in test data point [True]
96 Text feature [indicate] present in test data point [True]
97 Text feature [deletion] present in test data point [True]
98 Text feature [10] present in test data point [True]
99 Text feature [proteins] present in test data point [True]
Out of the top 100 features 87 are present in query point

```

4.1.1.4. Feature Importance, Incorrectly classified point

```

In [57]: test_point_index = 100
no_feature = 100
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(
test_x_onehotCoding[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:, :no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index]

```

```
],test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

Predicted Class : 4

Predicted Class Probabilities: [[0.058 0.0402 0.0105 0.747 0.032
5 0.0333 0.0721 0.0033 0.003]]

Actual Class : 1

9 Text feature [activity] present in test data point [True]
10 Text feature [protein] present in test data point [True]
14 Text feature [function] present in test data point [True]
15 Text feature [experiments] present in test data point [True]
16 Text feature [missense] present in test data point [True]
17 Text feature [acid] present in test data point [True]
19 Text feature [ability] present in test data point [True]
20 Text feature [whereas] present in test data point [True]
24 Text feature [results] present in test data point [True]
25 Text feature [described] present in test data point [True]
26 Text feature [determined] present in test data point [True]
27 Text feature [important] present in test data point [True]
28 Text feature [shown] present in test data point [True]
29 Text feature [mammalian] present in test data point [True]
30 Text feature [catalytic] present in test data point [True]
31 Text feature [whether] present in test data point [True]
32 Text feature [related] present in test data point [True]
33 Text feature [amino] present in test data point [True]
34 Text feature [indicated] present in test data point [True]
35 Text feature [suppressor] present in test data point [True]
36 Text feature [mutations] present in test data point [True]
37 Text feature [also] present in test data point [True]
38 Text feature [loss] present in test data point [True]
39 Text feature [functions] present in test data point [True]
40 Text feature [functional] present in test data point [True]
41 Text feature [although] present in test data point [True]
42 Text feature [two] present in test data point [True]
43 Text feature [type] present in test data point [True]
44 Text feature [reduced] present in test data point [True]
46 Text feature [may] present in test data point [True]
47 Text feature [determine] present in test data point [True]
48 Text feature [phosphatase] present in test data point [True]
49 Text feature [phosphatase] present in test data point [True]

49 Text feature [levels] present in test data point [True]
50 Text feature [correspond] present in test data point [True]
52 Text feature [vivo] present in test data point [True]
53 Text feature [three] present in test data point [True]
57 Text feature [bind] present in test data point [True]
59 Text feature [previously] present in test data point [True]
60 Text feature [30] present in test data point [True]
61 Text feature [associated] present in test data point [True]
62 Text feature [containing] present in test data point [True]
64 Text feature [loops] present in test data point [True]
65 Text feature [either] present in test data point [True]
66 Text feature [expressed] present in test data point [True]
67 Text feature [indicate] present in test data point [True]
70 Text feature [suggest] present in test data point [True]
71 Text feature [within] present in test data point [True]
72 Text feature [critical] present in test data point [True]
73 Text feature [therefore] present in test data point [True]
75 Text feature [thus] present in test data point [True]
76 Text feature [analyzed] present in test data point [True]
77 Text feature [similar] present in test data point [True]
78 Text feature [effects] present in test data point [True]
79 Text feature [generated] present in test data point [True]
80 Text feature [fully] present in test data point [True]
82 Text feature [made] present in test data point [True]
83 Text feature [show] present in test data point [True]
86 Text feature [cellular] present in test data point [True]
88 Text feature [addition] present in test data point [True]
89 Text feature [result] present in test data point [True]
90 Text feature [analysis] present in test data point [True]
92 Text feature [however] present in test data point [True]
93 Text feature [except] present in test data point [True]
94 Text feature [mutants] present in test data point [True]
95 Text feature [terminal] present in test data point [True]
96 Text feature [system] present in test data point [True]
97 Text feature [examine] present in test data point [True]
98 Text feature [one] present in test data point [True]
Out of the top 100 features 68 are present in query point

4.2. K Nearest Neighbour Classification

4.2.1. Hyper parameter tuning

```
In [58]: # find more about KNeighborsClassifier() here http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
# -----
# default parameter
# KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2,
# metric='minkowski', metric_params=None, n_jobs=1, **kwargs)

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

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

```

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

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

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

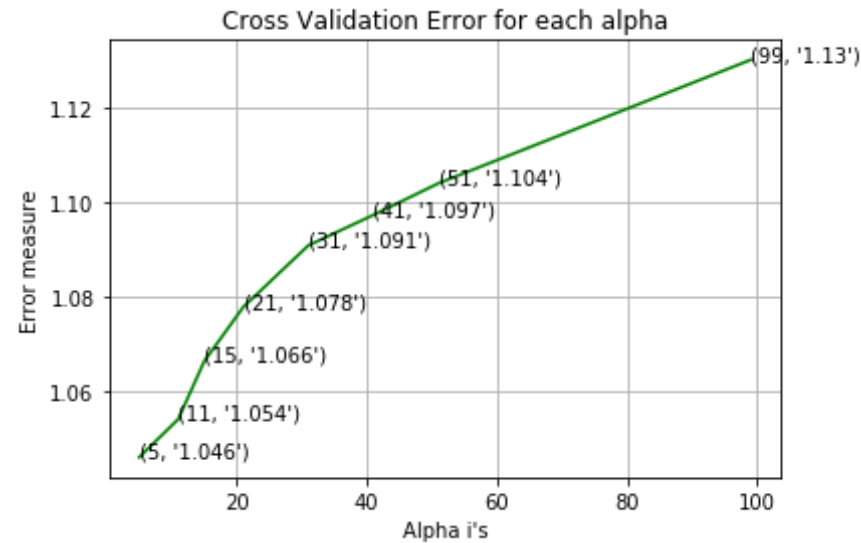
best_alpha = np.argmin(cv_log_error_array)
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_responseCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log

```

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

```
for alpha = 5
Log Loss : 1.04595011861
for alpha = 11
Log Loss : 1.05402233576
for alpha = 15
Log Loss : 1.06629694488
for alpha = 21
Log Loss : 1.07777603261
for alpha = 31
Log Loss : 1.0908269505
for alpha = 41
Log Loss : 1.09727769916
for alpha = 51
Log Loss : 1.10398589006
for alpha = 99
Log Loss : 1.13014428869
```



For values of best alpha = 5 The train log loss is: 0.494622657544
 For values of best alpha = 5 The cross validation log loss is: 1.04595011861
 For values of best alpha = 5 The test log loss is: 1.0502700101

4.2.2. Testing the model with best hyper paramters

In [59]:

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

# methods of
# fit(X, y) : Fit the model using X as training data and y as target va
lues
# predict(X):Predict the class labels for the provided data
# predict_proba(X):Return probability estimates for the test data X.
#-----
```

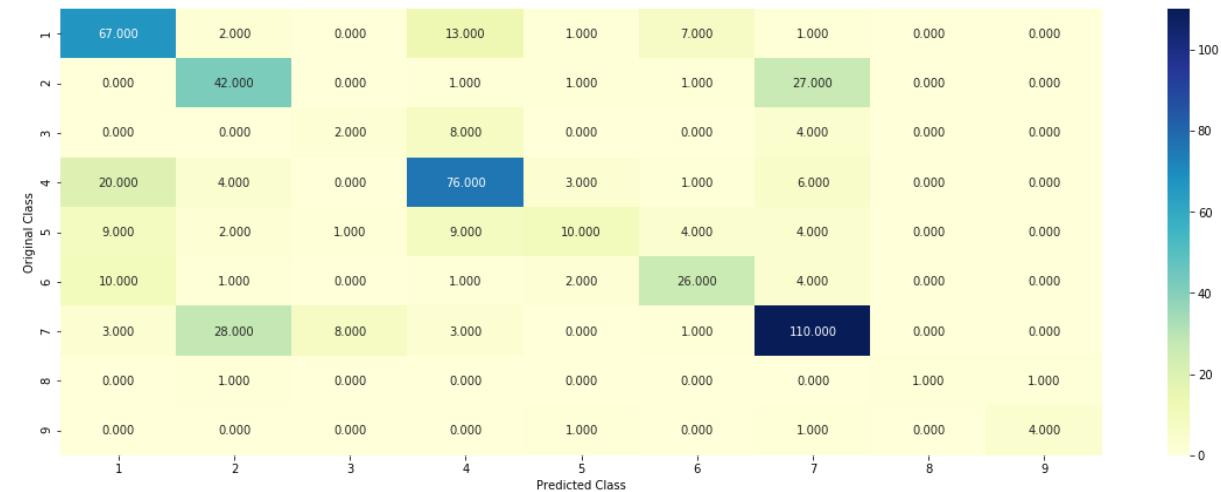
```
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/k-nearest-neighbors-geometric-intuition-with-a-toy-example-1/
```

```
#-----  
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])  
predict_and_plot_confusion_matrix(train_x_responseCoding, train_y, cv_x  
_responseCoding, cv_y, clf)
```

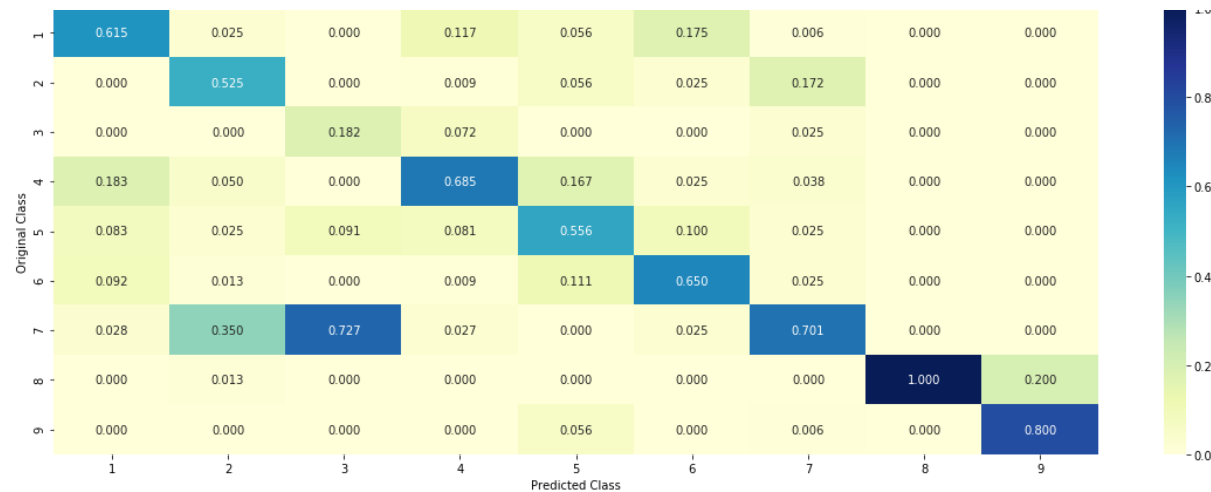
Log loss : 1.04595011861

Number of mis-classified points : 0.36466165413533835

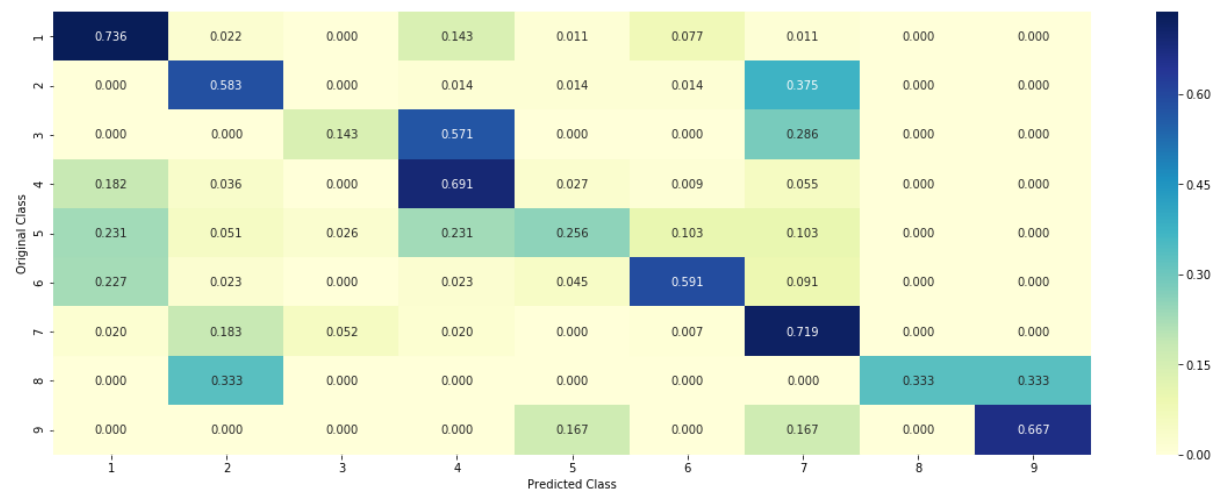
----- Confusion matrix -----



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



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



4.2.3.Sample Query point -1

```
In [60]: clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
          clf.fit(train_x_responseCoding, train_y)
          sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
```

```

sig_clf.fit(train_x_responseCoding, train_y)

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

```

```

Predicted Class : 1
Actual Class : 4
The 5 nearest neighbours of the test points belongs to classes [1 1 1 1 1]
Frequency of nearest points : Counter({1: 5})

```

4.2.4. Sample Query Point-2

```

In [61]: clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)

test_point_index = 100

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

```

```

Predicted Class : 4

```

Actual Class : 1
the k value for knn is 5 and the nearest neighbours of the test points
belongs to classes [4 4 4 4 4]
Frequency of nearest points : Counter({4: 5})

4.3. Logistic Regression

4.3.1. With Class balancing

4.3.1.1. Hyper paramter tuning

```
In [62]: # read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.SGDClassifier.html
# -----
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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

# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html
```

```

# -----
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, metho
d='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight])    Fit the calibrated model
# get_params([deep])           Get parameters for this estimator.
# predict(X)                    Predict the target of new samples.
# predict_proba(X)              Posterior probabilities of classification
#-----
# video link:
#-----

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

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

```

```

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

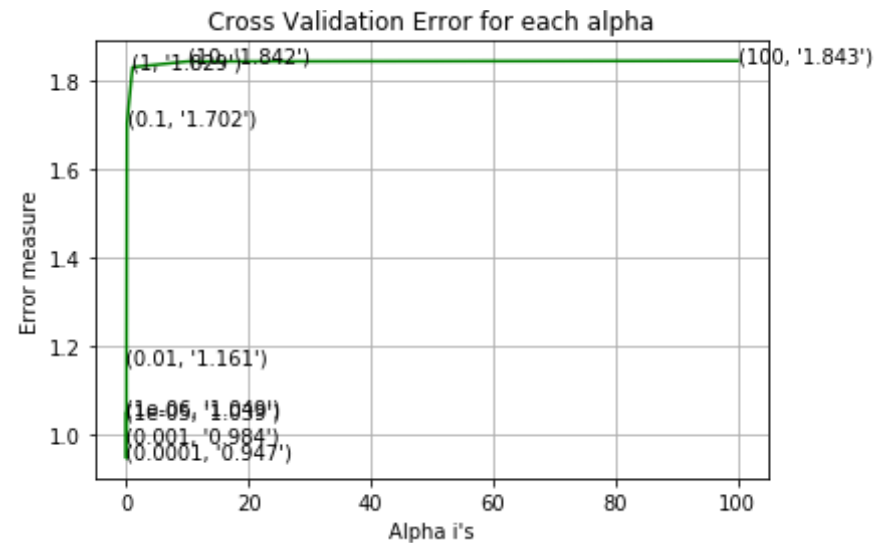
predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log
loss is:", log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross vali
dation log loss is:", log_loss(y_cv, predict_y, labels=clf.classes_, eps
=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log l
oss is:", log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))

```

```

for alpha = 1e-06
Log Loss : 1.04882754677
for alpha = 1e-05
Log Loss : 1.0387167851
for alpha = 0.0001
Log Loss : 0.946760417846
for alpha = 0.001
Log Loss : 0.98385630929
for alpha = 0.01
Log Loss : 1.16129714618
for alpha = 0.1
Log Loss : 1.70240833209
for alpha = 1
Log Loss : 1.82904966173
for alpha = 10
Log Loss : 1.84199869833
for alpha = 100
Log Loss : 1.84336599949

```



For values of best alpha = 0.0001 The train log loss is: 0.423831670433

For values of best alpha = 0.0001 The cross validation log loss is: 0.946760417846

For values of best alpha = 0.0001 The test log loss is: 0.992775387113

4.3.1.2. Testing the model with best hyper paramters

In [63]:

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

# some of methods
# fit(X, y[, coef_init, intercept_init, ...])    Fit linear model with Stochastic Gradient Descent.
```

```
# predict(X)    Predict class labels for samples in X.

#-----
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/geometric-intuition-1/
#-----
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], p
enalty='l2', loss='log', random_state=42)
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y, cv_x_o
nehotCoding, cv_y, clf)
```

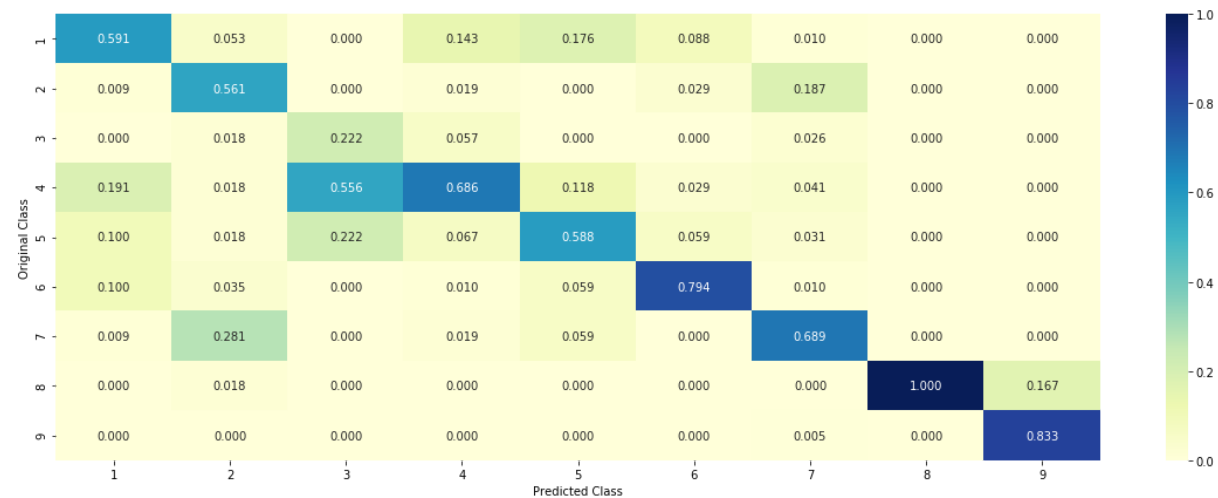
Log loss : 0.946760417846

Number of mis-classified points : 0.34774436090225563

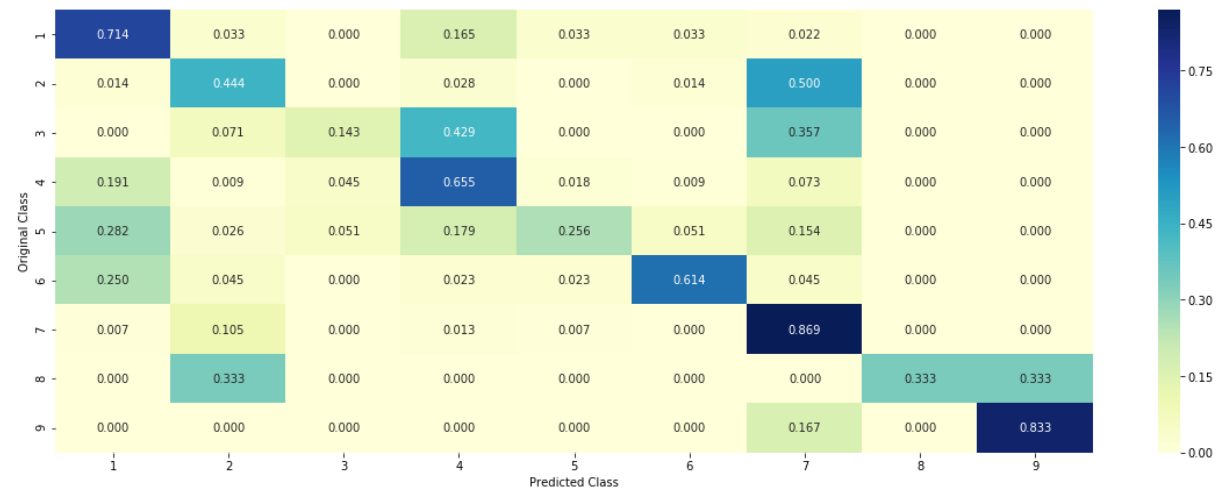
----- Confusion matrix -----



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



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



4.3.1.3. Feature Importance

```
In [64]: def get_imp_feature_names(text, indices, removed_ind = []):
          word_present = 0
          tabulte_list = []
```



```

increasingorder_ind = 0
for i in indices:
    if i < train_gene_feature_onehotCoding.shape[1]:
        tabulte_list.append([increasingorder_ind, "Gene", "Yes"])
    elif i < 18:
        tabulte_list.append([increasingorder_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([increasingorder_ind, train_text_features
[i], yes_no])
            increasingorder_ind += 1
        print(word_present, "most important features are present in our que
ry point")
        print("-"*50)
        print("The features that are most important of the ", predicted_cls[
0], " class:")
        print(tabulate(tabulte_list, headers=["Index", 'Feature name', 'Pre
sent or Not']))

```

4.3.1.3.1. Correctly Classified point

```

In [65]: # from tabulate import tabulate
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], p
enalty='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[test_point_index]), 4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:, :no_feature]
print("-"*50)

```

```
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

Predicted Class : 1

Predicted Class Probabilities: [[0.8836 0.0063 0.0041 0.0714 0.0117 0.0123 0.0066 0.0013 0.0027]]

Actual Class : 4

89 Text feature [surface] present in test data point [True]
110 Text feature [plays] present in test data point [True]
131 Text feature [nucleus] present in test data point [True]
162 Text feature [intron] present in test data point [True]
178 Text feature [mediate] present in test data point [True]
194 Text feature [deficient] present in test data point [True]
205 Text feature [panel] present in test data point [True]
213 Text feature [define] present in test data point [True]
218 Text feature [suppress] present in test data point [True]
231 Text feature [repeat] present in test data point [True]
236 Text feature [infected] present in test data point [True]
238 Text feature [parallel] present in test data point [True]
255 Text feature [lost] present in test data point [True]
275 Text feature [germ] present in test data point [True]
282 Text feature [wide] present in test data point [True]
285 Text feature [exposed] present in test data point [True]
286 Text feature [causes] present in test data point [True]
288 Text feature [mainly] present in test data point [True]
289 Text feature [corresponding] present in test data point [True]
290 Text feature [panels] present in test data point [True]
291 Text feature [novo] present in test data point [True]
292 Text feature [specificity] present in test data point [True]
295 Text feature [6b] present in test data point [True]
300 Text feature [patterns] present in test data point [True]
303 Text feature [knockdown] present in test data point [True]
305 Text feature [matched] present in test data point [True]
307 Text feature [influence] present in test data point [True]
308 Text feature [monoclonal] present in test data point [True]
312 Text feature [transcriptional] present in test data point [True]
316 Text feature [susceptibility] present in test data point [True]
318 Text feature [quantified] present in test data point [True]

320 Text feature [difficult] present in test data point [True]
325 Text feature [region] present in test data point [True]
326 Text feature [colonies] present in test data point [True]
327 Text feature [efficiency] present in test data point [True]
329 Text feature [rt] present in test data point [True]
330 Text feature [detectable] present in test data point [True]
331 Text feature [alter] present in test data point [True]
332 Text feature [identify] present in test data point [True]
335 Text feature [nuclear] present in test data point [True]
341 Text feature [upon] present in test data point [True]
343 Text feature [splicing] present in test data point [True]
345 Text feature [kb] present in test data point [True]
346 Text feature [adhesion] present in test data point [True]
348 Text feature [fraction] present in test data point [True]
350 Text feature [via] present in test data point [True]
351 Text feature [digestion] present in test data point [True]
352 Text feature [page] present in test data point [True]
353 Text feature [reporter] present in test data point [True]
354 Text feature [reveal] present in test data point [True]
356 Text feature [signal] present in test data point [True]
358 Text feature [scale] present in test data point [True]
360 Text feature [area] present in test data point [True]
362 Text feature [tail] present in test data point [True]
363 Text feature [kda] present in test data point [True]
364 Text feature [functions] present in test data point [True]
365 Text feature [future] present in test data point [True]
366 Text feature [amount] present in test data point [True]
370 Text feature [next] present in test data point [True]
372 Text feature [sirna] present in test data point [True]
376 Text feature [ld] present in test data point [True]
377 Text feature [function] present in test data point [True]
378 Text feature [4d] present in test data point [True]
379 Text feature [order] present in test data point [True]
380 Text feature [transiently] present in test data point [True]
382 Text feature [obtain] present in test data point [True]
388 Text feature [deletion] present in test data point [True]
390 Text feature [94] present in test data point [True]
391 Text feature [tabdownload] present in test data point [True]
393 Text feature [embedded] present in test data point [True]

397 Text feature [37] present in test data point [True]
398 Text feature [ability] present in test data point [True]
399 Text feature [colorectal] present in test data point [True]
401 Text feature [contain] present in test data point [True]
406 Text feature [controls] present in test data point [True]
409 Text feature [figureopen] present in test data point [True]
410 Text feature [displayed] present in test data point [True]
412 Text feature [reduced] present in test data point [True]
414 Text feature [bovine] present in test data point [True]
416 Text feature [population] present in test data point [True]
418 Text feature [loss] present in test data point [True]
423 Text feature [analyses] present in test data point [True]
424 Text feature [turn] present in test data point [True]
425 Text feature [96] present in test data point [True]
427 Text feature [arrest] present in test data point [True]
428 Text feature [acids] present in test data point [True]
430 Text feature [structure] present in test data point [True]
431 Text feature [contact] present in test data point [True]
433 Text feature [regulate] present in test data point [True]
434 Text feature [like] present in test data point [True]
435 Text feature [signals] present in test data point [True]
437 Text feature [smad3] present in test data point [True]
438 Text feature [rather] present in test data point [True]
439 Text feature [blood] present in test data point [True]
442 Text feature [59] present in test data point [True]
443 Text feature [sections] present in test data point [True]
444 Text feature [loading] present in test data point [True]
446 Text feature [affect] present in test data point [True]
447 Text feature [processes] present in test data point [True]
448 Text feature [molecules] present in test data point [True]
449 Text feature [notably] present in test data point [True]
450 Text feature [defined] present in test data point [True]
451 Text feature [1c] present in test data point [True]
452 Text feature [actin] present in test data point [True]
453 Text feature [appears] present in test data point [True]
454 Text feature [com] present in test data point [True]
455 Text feature [gel] present in test data point [True]
456 Text feature [capacity] present in test data point [True]
457 Text feature [reduce] present in test data point [True]

```
458 Text feature [vivo] present in test data point [True]
459 Text feature [vitro] present in test data point [True]
462 Text feature [revealed] present in test data point [True]
464 Text feature [immunoblotting] present in test data point [True]
465 Text feature [05] present in test data point [True]
468 Text feature [encoding] present in test data point [True]
470 Text feature [extracted] present in test data point [True]
471 Text feature [center] present in test data point [True]
476 Text feature [sl] present in test data point [True]
478 Text feature [commonly] present in test data point [True]
480 Text feature [change] present in test data point [True]
482 Text feature [thereby] present in test data point [True]
484 Text feature [reduction] present in test data point [True]
485 Text feature [complexes] present in test data point [True]
486 Text feature [lb] present in test data point [True]
487 Text feature [peptide] present in test data point [True]
488 Text feature [sds] present in test data point [True]
489 Text feature [fold] present in test data point [True]
490 Text feature [tgf] present in test data point [True]
491 Text feature [ratios] present in test data point [True]
492 Text feature [colon] present in test data point [True]
495 Text feature [importantly] present in test data point [True]
496 Text feature [biological] present in test data point [True]
498 Text feature [smad] present in test data point [True]
499 Text feature [labeled] present in test data point [True]
Out of the top 500 features 134 are present in query point
```

4.3.1.3.2. *Incorrectly Classified point*

```
In [66]: test_point_index = 100
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(
test_x_onehotCoding[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:, :no_feature]
print("-"*50)
```

```
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)
```

Predicted Class : 1

Predicted Class Probabilities: [[0.4919 0.0125 0.0041 0.4586 0.0092 0.0128 0.0079 0.0015 0.0014]]

Actual Class : 1

89 Text feature [surface] present in test data point [True]
110 Text feature [plays] present in test data point [True]
162 Text feature [intron] present in test data point [True]
164 Text feature [truncation] present in test data point [True]
194 Text feature [deficient] present in test data point [True]
205 Text feature [panel] present in test data point [True]
206 Text feature [repeats] present in test data point [True]
213 Text feature [define] present in test data point [True]
224 Text feature [hydrophobic] present in test data point [True]
231 Text feature [repeat] present in test data point [True]
233 Text feature [populations] present in test data point [True]
249 Text feature [sequenced] present in test data point [True]
255 Text feature [lost] present in test data point [True]
275 Text feature [germ] present in test data point [True]
285 Text feature [exposed] present in test data point [True]
286 Text feature [causes] present in test data point [True]
288 Text feature [mainly] present in test data point [True]
289 Text feature [corresponding] present in test data point [True]
305 Text feature [matched] present in test data point [True]
307 Text feature [influence] present in test data point [True]
316 Text feature [susceptibility] present in test data point [True]
318 Text feature [quantified] present in test data point [True]
322 Text feature [frameshift] present in test data point [True]
325 Text feature [region] present in test data point [True]
327 Text feature [efficiency] present in test data point [True]
330 Text feature [detectable] present in test data point [True]
332 Text feature [identify] present in test data point [True]
335 Text feature [nuclear] present in test data point [True]
341 Text feature [upon] present in test data point [True]
345 Text feature [kb] present in test data point [True]
348 Text feature [fraction] present in test data point [True]

349 Text feature [excluded] present in test data point [True]
350 Text feature [via] present in test data point [True]
354 Text feature [reveal] present in test data point [True]
356 Text feature [signal] present in test data point [True]
358 Text feature [scale] present in test data point [True]
360 Text feature [area] present in test data point [True]
362 Text feature [tail] present in test data point [True]
364 Text feature [functions] present in test data point [True]
370 Text feature [next] present in test data point [True]
376 Text feature [ld] present in test data point [True]
377 Text feature [function] present in test data point [True]
379 Text feature [order] present in test data point [True]
388 Text feature [deletion] present in test data point [True]
390 Text feature [94] present in test data point [True]
397 Text feature [37] present in test data point [True]
398 Text feature [ability] present in test data point [True]
399 Text feature [colorectal] present in test data point [True]
403 Text feature [subunits] present in test data point [True]
406 Text feature [controls] present in test data point [True]
412 Text feature [reduced] present in test data point [True]
416 Text feature [population] present in test data point [True]
418 Text feature [loss] present in test data point [True]
423 Text feature [analyses] present in test data point [True]
426 Text feature [http] present in test data point [True]
430 Text feature [structure] present in test data point [True]
432 Text feature [nonsense] present in test data point [True]
434 Text feature [like] present in test data point [True]
435 Text feature [signals] present in test data point [True]
438 Text feature [rather] present in test data point [True]
439 Text feature [blood] present in test data point [True]
441 Text feature [conducted] present in test data point [True]
442 Text feature [59] present in test data point [True]
443 Text feature [sections] present in test data point [True]
445 Text feature [base] present in test data point [True]
446 Text feature [affect] present in test data point [True]
450 Text feature [defined] present in test data point [True]
451 Text feature [lc] present in test data point [True]
453 Text feature [appears] present in test data point [True]
454 Text feature [com] present in test data point [True]

```
455 Text feature [gel] present in test data point [True]
458 Text feature [vivo] present in test data point [True]
462 Text feature [revealed] present in test data point [True]
470 Text feature [extracted] present in test data point [True]
471 Text feature [center] present in test data point [True]
473 Text feature [somatic] present in test data point [True]
476 Text feature [sl] present in test data point [True]
480 Text feature [change] present in test data point [True]
482 Text feature [thereby] present in test data point [True]
483 Text feature [interacting] present in test data point [True]
484 Text feature [reduction] present in test data point [True]
485 Text feature [complexes] present in test data point [True]
486 Text feature [lb] present in test data point [True]
488 Text feature [sds] present in test data point [True]
489 Text feature [fold] present in test data point [True]
491 Text feature [ratios] present in test data point [True]
492 Text feature [colon] present in test data point [True]
495 Text feature [importantly] present in test data point [True]
499 Text feature [labeled] present in test data point [True]
Out of the top 500 features 89 are present in query point
```

4.3.2. Without Class balancing

4.3.2.1. Hyper paramter tuning

```
In [67]: # read more about SGDClassifier() at http://scikit-learn.org/stable/mod
         ules/generated/sklearn.linear_model.SGDClassifier.html
         # -----
         # default parameters
         # SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
         5, fit_intercept=True, max_iter=None, tol=None,
         # shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
         arning_rate='optimal', eta0=0.0, power_t=0.5,
         # class_weight=None, warm_start=False, average=False, n_iter=None)

         # some of methods
```



```

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

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

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

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

```

```

classes_, eps=1e-15))
    print("Log Loss :",log_loss(cv_y, sig_clf_probs))

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

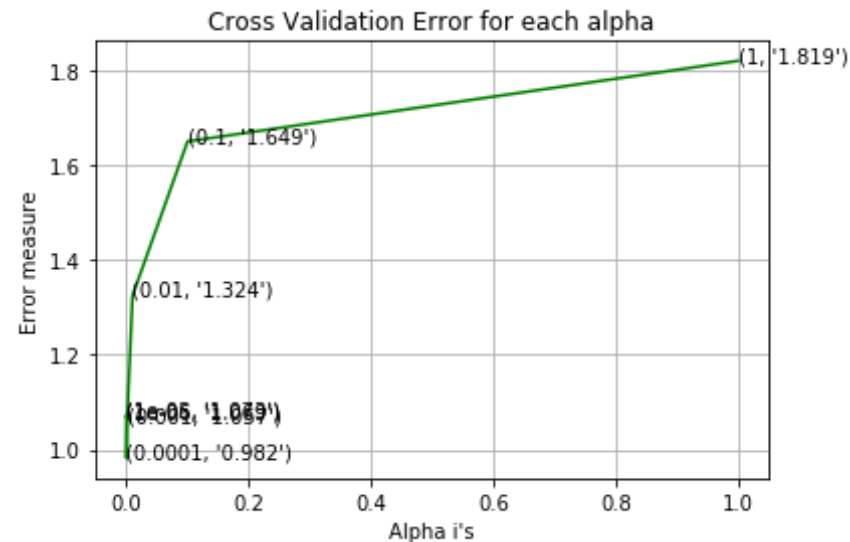
best_alpha = np.argmin(cv_log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log',
random_state=42)
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log
loss is:",log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross vali
dation log loss is:",log_loss(y_cv, predict_y, labels=clf.classes_, eps
=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log l
oss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))

for alpha = 1e-06
Log Loss : 1.06911519196
for alpha = 1e-05
Log Loss : 1.07289062494
for alpha = 0.0001
Log Loss : 0.981776499777
for alpha = 0.001

```

Log Loss : 1.05705170049
 for alpha = 0.01
 Log Loss : 1.32420444791
 for alpha = 0.1
 Log Loss : 1.64910812837
 for alpha = 1
 Log Loss : 1.8188711033



For values of best alpha = 0.0001 The train log loss is: 0.416533318356
 For values of best alpha = 0.0001 The cross validation log loss is: 0.981776499777
 For values of best alpha = 0.0001 The test log loss is: 1.00899889995

4.3.2.2. Testing model with best hyper parameters

In [68]:

```
# read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
# -----
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
```

```

5, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, le
arning_rate='optimal', eta0=0.0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

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

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

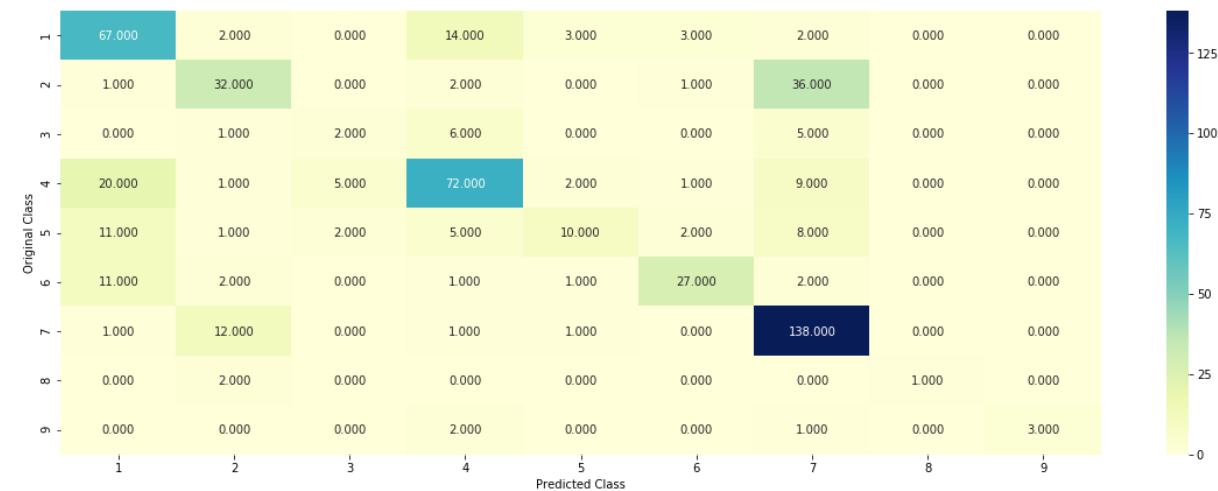
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log',
random_state=42)
predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y, cv_x_o
nehotCoding, cv_y, clf)

```

Log loss : 0.981776499777

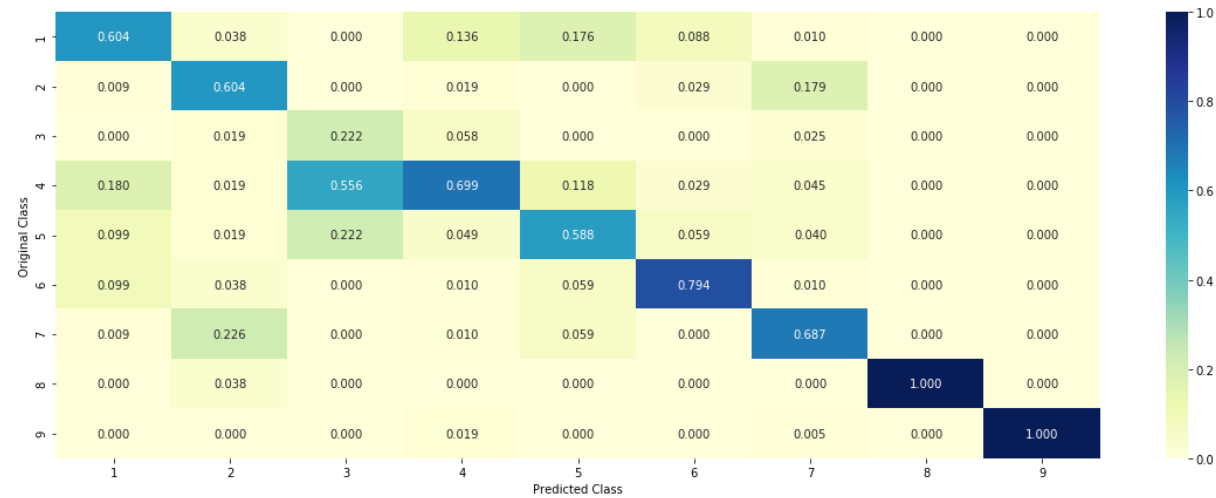
Number of mis-classified points : 0.3383458646616541

----- Confusion matrix -----



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

--



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



4.3.2.3. Feature Importance, Correctly Classified point

```
In [69]: 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[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index
],test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test
_point_index], no_feature)

```

```

Predicted Class : 1
Predicted Class Probabilities: [[ 0.8691  0.0068  0.0026  0.0851  0.012
6  0.0131  0.0073  0.0015  0.002 ]]
Actual Class : 4

```

```

-----
100 Text feature [surface] present in test data point [True]
127 Text feature [plays] present in test data point [True]
154 Text feature [nucleus] present in test data point [True]
193 Text feature [panel] present in test data point [True]
195 Text feature [intron] present in test data point [True]
198 Text feature [mediate] present in test data point [True]
202 Text feature [infected] present in test data point [True]
215 Text feature [parallel] present in test data point [True]
222 Text feature [deficient] present in test data point [True]
226 Text feature [define] present in test data point [True]
232 Text feature [suppress] present in test data point [True]
233 Text feature [repeat] present in test data point [True]
249 Text feature [lost] present in test data point [True]
279 Text feature [causes] present in test data point [True]
280 Text feature [6b] present in test data point [True]
282 Text feature [panels] present in test data point [True]
291 Text feature [corresponding] present in test data point [True]
293 Text feature [germ] present in test data point [True]
295 Text feature [patterns] present in test data point [True]
297 Text feature [novo] present in test data point [True]
298 Text feature [wide] present in test data point [True]
302 Text feature [exposed] present in test data point [True]

```

303 Text feature [detectable] present in test data point [True]
304 Text feature [mainly] present in test data point [True]
305 Text feature [rt] present in test data point [True]
307 Text feature [knockdown] present in test data point [True]
308 Text feature [alter] present in test data point [True]
309 Text feature [matched] present in test data point [True]
311 Text feature [monoclonal] present in test data point [True]
313 Text feature [specificity] present in test data point [True]
314 Text feature [influence] present in test data point [True]
317 Text feature [colonies] present in test data point [True]
318 Text feature [susceptibility] present in test data point [True]
320 Text feature [splicing] present in test data point [True]
321 Text feature [quantified] present in test data point [True]
323 Text feature [transcriptional] present in test data point [True]
326 Text feature [upon] present in test data point [True]
327 Text feature [via] present in test data point [True]
328 Text feature [identify] present in test data point [True]
330 Text feature [region] present in test data point [True]
333 Text feature [fraction] present in test data point [True]
336 Text feature [difficult] present in test data point [True]
340 Text feature [order] present in test data point [True]
342 Text feature [signal] present in test data point [True]
344 Text feature [kda] present in test data point [True]
346 Text feature [future] present in test data point [True]
350 Text feature [transiently] present in test data point [True]
355 Text feature [efficiency] present in test data point [True]
357 Text feature [reveal] present in test data point [True]
358 Text feature [adhesion] present in test data point [True]
360 Text feature [page] present in test data point [True]
361 Text feature [function] present in test data point [True]
362 Text feature [next] present in test data point [True]
363 Text feature [obtain] present in test data point [True]
365 Text feature [reporter] present in test data point [True]
366 Text feature [nuclear] present in test data point [True]
368 Text feature [area] present in test data point [True]
370 Text feature [scale] present in test data point [True]
371 Text feature [amount] present in test data point [True]
374 Text feature [deletion] present in test data point [True]
376 Text feature [functions] present in test data point [True]

380 Text feature [tail] present in test data point [True]
381 Text feature [digestion] present in test data point [True]
382 Text feature [ability] present in test data point [True]
383 Text feature [arrest] present in test data point [True]
384 Text feature [sirna] present in test data point [True]
385 Text feature [kb] present in test data point [True]
386 Text feature [acids] present in test data point [True]
388 Text feature [tabdownload] present in test data point [True]
392 Text feature [94] present in test data point [True]
393 Text feature [sections] present in test data point [True]
394 Text feature [ld] present in test data point [True]
397 Text feature [4d] present in test data point [True]
398 Text feature [embedded] present in test data point [True]
399 Text feature [37] present in test data point [True]
401 Text feature [96] present in test data point [True]
404 Text feature [59] present in test data point [True]
405 Text feature [controls] present in test data point [True]
406 Text feature [like] present in test data point [True]
409 Text feature [colorectal] present in test data point [True]
410 Text feature [figureopen] present in test data point [True]
411 Text feature [loading] present in test data point [True]
412 Text feature [reduced] present in test data point [True]
413 Text feature [signals] present in test data point [True]
418 Text feature [population] present in test data point [True]
419 Text feature [lc] present in test data point [True]
421 Text feature [vitro] present in test data point [True]
423 Text feature [actin] present in test data point [True]
424 Text feature [molecules] present in test data point [True]
427 Text feature [blood] present in test data point [True]
429 Text feature [com] present in test data point [True]
431 Text feature [bovine] present in test data point [True]
432 Text feature [colon] present in test data point [True]
434 Text feature [turn] present in test data point [True]
436 Text feature [contact] present in test data point [True]
437 Text feature [rather] present in test data point [True]
440 Text feature [analyses] present in test data point [True]
441 Text feature [smad3] present in test data point [True]
442 Text feature [sl] present in test data point [True]
443 Text feature [displayed] present in test data point [True]


```
444 Text feature [vivo] present in test data point [True]
446 Text feature [affect] present in test data point [True]
448 Text feature [encoding] present in test data point [True]
449 Text feature [contain] present in test data point [True]
451 Text feature [center] present in test data point [True]
452 Text feature [loss] present in test data point [True]
453 Text feature [revealed] present in test data point [True]
455 Text feature [structure] present in test data point [True]
456 Text feature [appears] present in test data point [True]
457 Text feature [05] present in test data point [True]
459 Text feature [reduce] present in test data point [True]
460 Text feature [type] present in test data point [True]
461 Text feature [lb] present in test data point [True]
464 Text feature [ratios] present in test data point [True]
465 Text feature [defined] present in test data point [True]
466 Text feature [fold] present in test data point [True]
467 Text feature [gel] present in test data point [True]
468 Text feature [labeled] present in test data point [True]
470 Text feature [tgf] present in test data point [True]
474 Text feature [regulate] present in test data point [True]
475 Text feature [alone] present in test data point [True]
479 Text feature [commonly] present in test data point [True]
480 Text feature [sds] present in test data point [True]
482 Text feature [processes] present in test data point [True]
483 Text feature [extracted] present in test data point [True]
484 Text feature [change] present in test data point [True]
486 Text feature [slightly] present in test data point [True]
488 Text feature [notably] present in test data point [True]
491 Text feature [biological] present in test data point [True]
492 Text feature [reduction] present in test data point [True]
494 Text feature [21] present in test data point [True]
496 Text feature [immunoblotting] present in test data point [True]
497 Text feature [importantly] present in test data point [True]
498 Text feature [suppression] present in test data point [True]
499 Text feature [appear] present in test data point [True]
Out of the top 500 features 135 are present in query point
```

4.3.2.4. Feature Importance, Inorrectly Classified point

```
In [70]: test_point_index = 100
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(
test_x_onehotCoding[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index]
,test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test
_point_index], no_feature)
```

```
Predicted Class : 4
Predicted Class Probabilities: [[ 0.4622  0.0128  0.0022  0.4925  0.008
7 0.0112  0.0084  0.001  0.0011]]
Actual Class : 1
```

```
-----
44 Text feature [suppressor] present in test data point [True]
50 Text feature [inactivating] present in test data point [True]
90 Text feature [material] present in test data point [True]
91 Text feature [missense] present in test data point [True]
110 Text feature [density] present in test data point [True]
115 Text feature [mammalian] present in test data point [True]
134 Text feature [tumorigenesis] present in test data point [True]
168 Text feature [main] present in test data point [True]
173 Text feature [iii] present in test data point [True]
179 Text feature [dimerization] present in test data point [True]
190 Text feature [nonsense] present in test data point [True]
196 Text feature [show] present in test data point [True]
204 Text feature [inactivation] present in test data point [True]
221 Text feature [isolated] present in test data point [True]
228 Text feature [deleted] present in test data point [True]
242 Text feature [loss] present in test data point [True]
247 Text feature [along] present in test data point [True]
252 Text feature [almost] present in test data point [True]
256 Text feature [mb] present in test data point [True]
262 Text feature [free] present in test data point [True]
265 Text feature [unable] present in test data point [True]
```

270 Text feature [apparent] present in test data point [True]
273 Text feature [washed] present in test data point [True]
288 Text feature [dominant] present in test data point [True]
296 Text feature [right] present in test data point [True]
303 Text feature [sigma] present in test data point [True]
312 Text feature [lack] present in test data point [True]
316 Text feature [germline] present in test data point [True]
319 Text feature [disruption] present in test data point [True]
321 Text feature [homozygous] present in test data point [True]
327 Text feature [risk] present in test data point [True]
330 Text feature [motif] present in test data point [True]
344 Text feature [despite] present in test data point [True]
358 Text feature [cannot] present in test data point [True]
363 Text feature [analysed] present in test data point [True]
364 Text feature [incidence] present in test data point [True]
369 Text feature [1998] present in test data point [True]
373 Text feature [functional] present in test data point [True]
378 Text feature [contribution] present in test data point [True]
382 Text feature [represent] present in test data point [True]
383 Text feature [homologous] present in test data point [True]
387 Text feature [suggest] present in test data point [True]
388 Text feature [fisher] present in test data point [True]
395 Text feature [account] present in test data point [True]
396 Text feature [left] present in test data point [True]
397 Text feature [deletion] present in test data point [True]
399 Text feature [solution] present in test data point [True]
404 Text feature [high] present in test data point [True]
408 Text feature [crc] present in test data point [True]
411 Text feature [regulates] present in test data point [True]
412 Text feature [lkb1] present in test data point [True]
415 Text feature [age] present in test data point [True]
420 Text feature [level] present in test data point [True]
421 Text feature [resulting] present in test data point [True]
426 Text feature [called] present in test data point [True]
439 Text feature [comprehensive] present in test data point [True]
440 Text feature [ii] present in test data point [True]
444 Text feature [due] present in test data point [True]
450 Text feature [genome] present in test data point [True]
454 Text feature [substrate] present in test data point [True]

```
455 Text feature [functionally] present in test data point [True]
457 Text feature [accessible] present in test data point [True]
458 Text feature [stained] present in test data point [True]
461 Text feature [thought] present in test data point [True]
466 Text feature [documented] present in test data point [True]
470 Text feature [62] present in test data point [True]
471 Text feature [min] present in test data point [True]
474 Text feature [affected] present in test data point [True]
477 Text feature [frameshift] present in test data point [True]
478 Text feature [described] present in test data point [True]
479 Text feature [responsible] present in test data point [True]
480 Text feature [suggesting] present in test data point [True]
481 Text feature [truncated] present in test data point [True]
483 Text feature [determine] present in test data point [True]
484 Text feature [moderate] present in test data point [True]
486 Text feature [catalytic] present in test data point [True]
491 Text feature [sds] present in test data point [True]
492 Text feature [representative] present in test data point [True]
494 Text feature [specifically] present in test data point [True]
495 Text feature [version] present in test data point [True]
498 Text feature [phosphatase] present in test data point [True]
499 Text feature [45] present in test data point [True]
Out of the top 500 features 82 are present in query point
```

4.4. Linear Support Vector Machines

4.4.1. Hyper parameter tuning

```
In [71]: # read more about support vector machines with linear kernels here http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

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

```

on_function_shape='ovr', random_state=None)

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

# find more about CalibratedClassifierCV here at http://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html
# -----
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, metho
d='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='bal
anced')
    clf = SGDClassifier( class_weight='balanced', alpha=i, penalty='l2'
, loss='hinge', random_state=42)
    clf.fit(train_x_onehotCoding, train_y)

```

```

sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)
sig_clf_probs = sig_clf.predict_proba(cv_x_onehotCoding)
cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=clf.
classes_, eps=1e-15))
print("Log Loss :", log_loss(cv_y, sig_clf_probs))

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

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

predict_y = sig_clf.predict_proba(train_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log
loss is:", log_loss(y_train, predict_y, labels=clf.classes_, eps=1e-15
))
predict_y = sig_clf.predict_proba(cv_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross vali
dation log loss is:", log_loss(y_cv, predict_y, labels=clf.classes_, eps
=1e-15))
predict_y = sig_clf.predict_proba(test_x_onehotCoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log l
oss is:", log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))

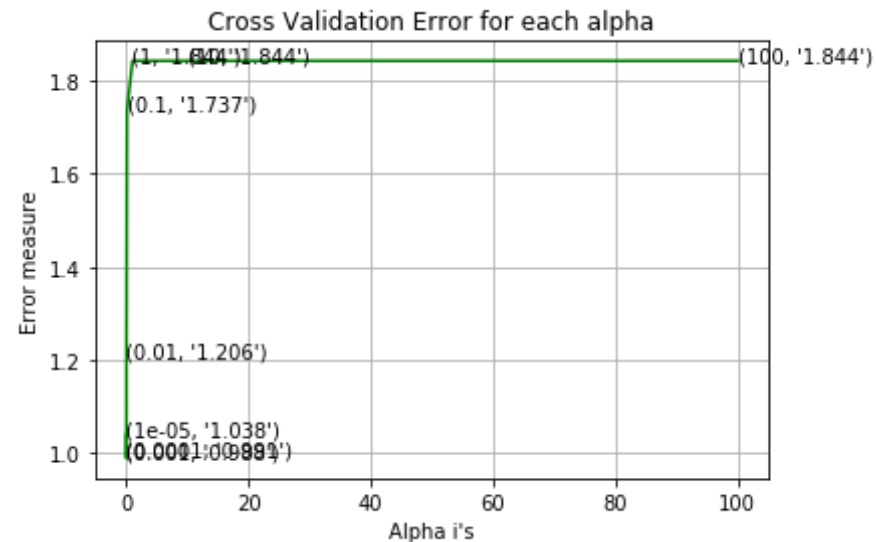
```

for C = 1e-05

```

Log Loss : 1.03798788188
for C = 0.0001
Log Loss : 0.991041892625
for C = 0.001
Log Loss : 0.987574714156
for C = 0.01
Log Loss : 1.20553878126
for C = 0.1
Log Loss : 1.7366760497
for C = 1
Log Loss : 1.84360036802
for C = 10
Log Loss : 1.84360077518
for C = 100
Log Loss : 1.84360080337

```



```

For values of best alpha = 0.001 The train log loss is: 0.536465147627
For values of best alpha = 0.001 The cross validation log loss is: 0.9
87574714156
For values of best alpha = 0.001 The test log loss is: 1.04853851575

```

4.4.2. Testing model with best hyper parameters

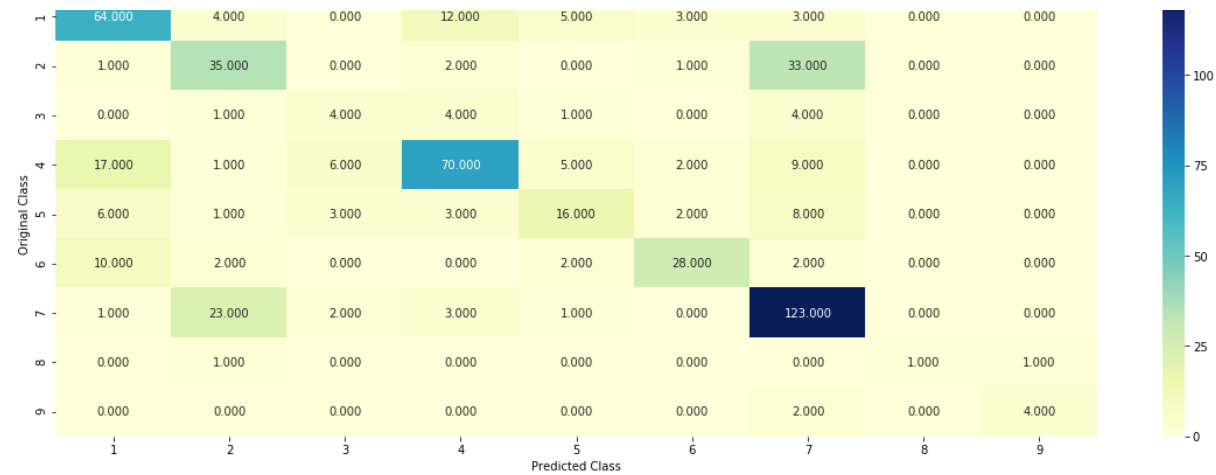
```
In [72]: # read more about support vector machines with linear kernels here http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

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

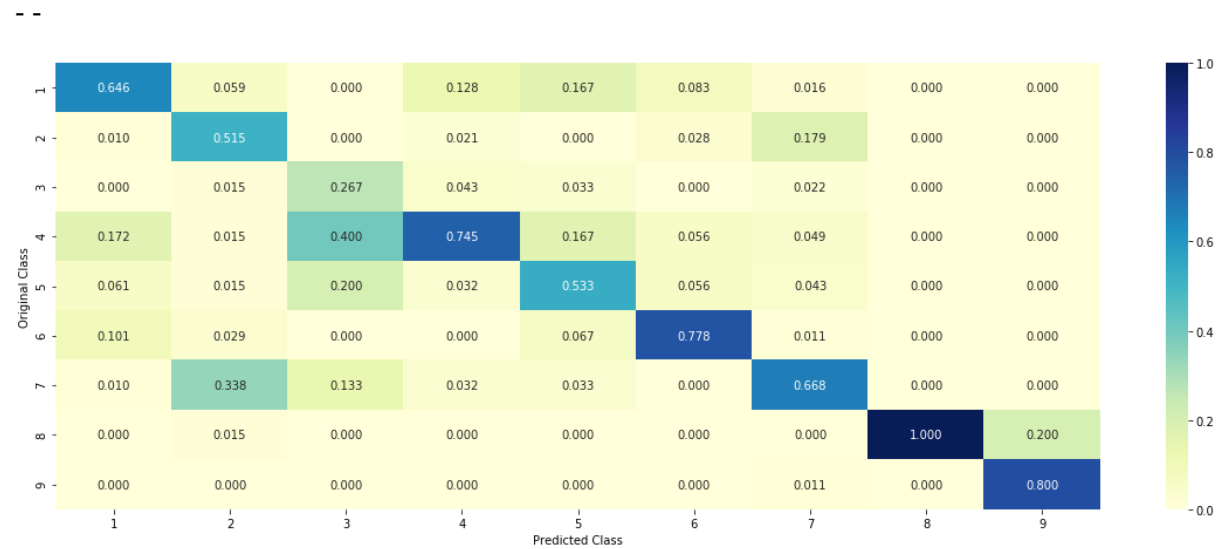
# Some of methods of SVM()
# fit(X, y, [sample_weight])    Fit the SVM model according to the give
# n training data.
# predict(X)    Perform classification on samples in X.
# -----
# video link: https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/mathematical-derivation-copy-8/
#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

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

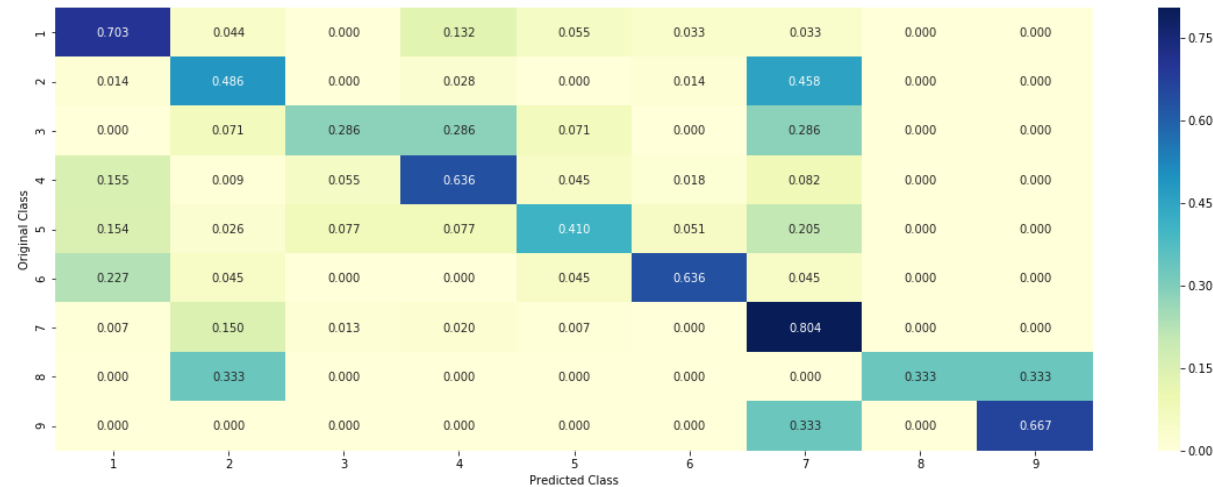
Log loss : 0.987574714156
Number of mis-classified points : 0.35150375939849626
----- Confusion matrix -----
```

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



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



4.3.3. Feature Importance

4.3.3.1. For Correctly classified point

```
In [73]: clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='hinge'
, random_state=42)
clf.fit(train_x_onehotCoding,train_y)
test_point_index = 1
# test_point_index = 100
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(
test_x_onehotCoding[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index]
,test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test
_point_index], no_feature)
```

```
Predicted Class : 1
Predicted Class Probabilities: [[ 0.8891  0.0171  0.0056  0.0245  0.025
 0.0199  0.0146  0.0014  0.0028]]
Actual Class : 4
```

```
-----
9 Text feature [surface] present in test data point [True]
14 Text feature [suppress] present in test data point [True]
16 Text feature [plays] present in test data point [True]
17 Text feature [mediate] present in test data point [True]
20 Text feature [nucleus] present in test data point [True]
221 Text feature [intron] present in test data point [True]
222 Text feature [deficient] present in test data point [True]
223 Text feature [adhesion] present in test data point [True]
225 Text feature [lost] present in test data point [True]
228 Text feature [exposed] present in test data point [True]
280 Text feature [panel] present in test data point [True]
282 Text feature [importantly] present in test data point [True]
285 Text feature [difficult] present in test data point [True]
286 Text feature [wide] present in test data point [True]
287 Text feature [repeat] present in test data point [True]
288 Text feature [mainly] present in test data point [True]
290 Text feature [alter] present in test data point [True]
292 Text feature [bcl] present in test data point [True]
296 Text feature [fraction] present in test data point [True]
300 Text feature [transiently] present in test data point [True]
302 Text feature [next] present in test data point [True]
303 Text feature [vivo] present in test data point [True]
304 Text feature [ability] present in test data point [True]
305 Text feature [signal] present in test data point [True]
307 Text feature [germ] present in test data point [True]
308 Text feature [matched] present in test data point [True]
309 Text feature [ld] present in test data point [True]
310 Text feature [parallel] present in test data point [True]
312 Text feature [notably] present in test data point [True]
314 Text feature [causing] present in test data point [True]
315 Text feature [splicing] present in test data point [True]
316 Text feature [scale] present in test data point [True]
317 Text feature [area] present in test data point [True]
318 Text feature [efficiency] present in test data point [True]
```

321 Text feature [thereby] present in test data point [True]
348 Text feature [reveal] present in test data point [True]
350 Text feature [corresponding] present in test data point [True]
351 Text feature [infected] present in test data point [True]
352 Text feature [define] present in test data point [True]
354 Text feature [96] present in test data point [True]
355 Text feature [via] present in test data point [True]
356 Text feature [novo] present in test data point [True]
357 Text feature [specificity] present in test data point [True]
358 Text feature [4d] present in test data point [True]
360 Text feature [identify] present in test data point [True]
361 Text feature [detectable] present in test data point [True]
362 Text feature [monoclonal] present in test data point [True]
363 Text feature [sirna] present in test data point [True]
364 Text feature [rather] present in test data point [True]
365 Text feature [regulate] present in test data point [True]
366 Text feature [susceptibility] present in test data point [True]
367 Text feature [knockdown] present in test data point [True]
368 Text feature [05] present in test data point [True]
369 Text feature [1c] present in test data point [True]
370 Text feature [suppression] present in test data point [True]
371 Text feature [contact] present in test data point [True]
372 Text feature [acids] present in test data point [True]
373 Text feature [37] present in test data point [True]
374 Text feature [affect] present in test data point [True]
375 Text feature [region] present in test data point [True]
378 Text feature [quantified] present in test data point [True]
379 Text feature [actin] present in test data point [True]
381 Text feature [functions] present in test data point [True]
383 Text feature [blood] present in test data point [True]
384 Text feature [panels] present in test data point [True]
385 Text feature [center] present in test data point [True]
387 Text feature [transcriptional] present in test data point [True]
393 Text feature [signals] present in test data point [True]
394 Text feature [embedded] present in test data point [True]
396 Text feature [vitro] present in test data point [True]
397 Text feature [rt] present in test data point [True]
398 Text feature [order] present in test data point [True]
399 Text feature [analyses] present in test data point [True]

400 Text feature [loss] present in test data point [True]
404 Text feature [patterns] present in test data point [True]
405 Text feature [peptide] present in test data point [True]
406 Text feature [structure] present in test data point [True]
407 Text feature [upon] present in test data point [True]
408 Text feature [like] present in test data point [True]
409 Text feature [percentage] present in test data point [True]
410 Text feature [1997] present in test data point [True]
411 Text feature [transfection] present in test data point [True]
412 Text feature [future] present in test data point [True]
415 Text feature [cultured] present in test data point [True]
416 Text feature [might] present in test data point [True]
417 Text feature [1b] present in test data point [True]
419 Text feature [essential] present in test data point [True]
421 Text feature [regulatory] present in test data point [True]
422 Text feature [tail] present in test data point [True]
423 Text feature [molecules] present in test data point [True]
424 Text feature [encoding] present in test data point [True]
425 Text feature [amount] present in test data point [True]
426 Text feature [2002] present in test data point [True]
427 Text feature [94] present in test data point [True]
428 Text feature [digestion] present in test data point [True]
429 Text feature [driven] present in test data point [True]
431 Text feature [harboring] present in test data point [True]
433 Text feature [qiagen] present in test data point [True]
434 Text feature [page] present in test data point [True]
437 Text feature [6b] present in test data point [True]
438 Text feature [causes] present in test data point [True]
439 Text feature [contain] present in test data point [True]
440 Text feature [sl] present in test data point [True]
441 Text feature [obtain] present in test data point [True]
442 Text feature [revealed] present in test data point [True]
443 Text feature [capacity] present in test data point [True]
444 Text feature [perhaps] present in test data point [True]
447 Text feature [function] present in test data point [True]
448 Text feature [gel] present in test data point [True]
449 Text feature [colorectal] present in test data point [True]
450 Text feature [kb] present in test data point [True]
451 Text feature [type] present in test data point [True]

```

453 Text feature [processes] present in test data point [True]
454 Text feature [smad3] present in test data point [True]
455 Text feature [wild] present in test data point [True]
457 Text feature [nuclear] present in test data point [True]
458 Text feature [resulted] present in test data point [True]
459 Text feature [though] present in test data point [True]
460 Text feature [additional] present in test data point [True]
462 Text feature [reporter] present in test data point [True]
463 Text feature [reduced] present in test data point [True]
487 Text feature [pair] present in test data point [True]
488 Text feature [established] present in test data point [True]
489 Text feature [paraffin] present in test data point [True]
490 Text feature [binding] present in test data point [True]
491 Text feature [interestingly] present in test data point [True]
493 Text feature [commonly] present in test data point [True]
494 Text feature [2008] present in test data point [True]
495 Text feature [additionally] present in test data point [True]
496 Text feature [particularly] present in test data point [True]
497 Text feature [assay] present in test data point [True]
Out of the top 500 features 131 are present in query point

```

4.3.3.2. For Incorrectly classified point

```

In [74]: test_point_index = 100
no_feature = 500
predicted_cls = sig_clf.predict(test_x_onehotCoding[test_point_index])
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(
test_x_onehotCoding[test_point_index]),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*50)
get_impfeature_names(indices[0], test_df['TEXT'].iloc[test_point_index],
test_df['Gene'].iloc[test_point_index],test_df['Variation'].iloc[test_point_index], no_feature)

```

```

Predicted Class : 1
Predicted Class Probabilities: [[ 0.5853  0.0204  0.0045  0.3527  0.010

```

```
9 0.0133 0.0107 0.0009 0.0013]]
```

```
Actual Class : 1
```

```
-----  
9 Text feature [surface] present in test data point [True]  
16 Text feature [plays] present in test data point [True]  
18 Text feature [repeats] present in test data point [True]  
219 Text feature [hydrophobic] present in test data point [True]  
221 Text feature [intron] present in test data point [True]  
222 Text feature [deficient] present in test data point [True]  
225 Text feature [lost] present in test data point [True]  
227 Text feature [truncation] present in test data point [True]  
228 Text feature [exposed] present in test data point [True]  
280 Text feature [panel] present in test data point [True]  
282 Text feature [importantly] present in test data point [True]  
284 Text feature [populations] present in test data point [True]  
287 Text feature [repeat] present in test data point [True]  
288 Text feature [mainly] present in test data point [True]  
296 Text feature [fraction] present in test data point [True]  
297 Text feature [sequenced] present in test data point [True]  
302 Text feature [next] present in test data point [True]  
303 Text feature [vivo] present in test data point [True]  
304 Text feature [ability] present in test data point [True]  
305 Text feature [signal] present in test data point [True]  
307 Text feature [germ] present in test data point [True]  
308 Text feature [matched] present in test data point [True]  
309 Text feature [ld] present in test data point [True]  
314 Text feature [causing] present in test data point [True]  
316 Text feature [scale] present in test data point [True]  
317 Text feature [area] present in test data point [True]  
318 Text feature [efficiency] present in test data point [True]  
321 Text feature [thereby] present in test data point [True]  
348 Text feature [reveal] present in test data point [True]  
350 Text feature [corresponding] present in test data point [True]  
352 Text feature [define] present in test data point [True]  
355 Text feature [via] present in test data point [True]  
360 Text feature [identify] present in test data point [True]  
361 Text feature [detectable] present in test data point [True]  
364 Text feature [rather] present in test data point [True]  
366 Text feature [susceptibility] present in test data point [True]
```

369 Text feature [1c] present in test data point [True]
373 Text feature [37] present in test data point [True]
374 Text feature [affect] present in test data point [True]
375 Text feature [region] present in test data point [True]
378 Text feature [quantified] present in test data point [True]
381 Text feature [functions] present in test data point [True]
383 Text feature [blood] present in test data point [True]
385 Text feature [center] present in test data point [True]
386 Text feature [position] present in test data point [True]
388 Text feature [base] present in test data point [True]
393 Text feature [signals] present in test data point [True]
398 Text feature [order] present in test data point [True]
399 Text feature [analyses] present in test data point [True]
400 Text feature [loss] present in test data point [True]
401 Text feature [subunits] present in test data point [True]
402 Text feature [somatic] present in test data point [True]
403 Text feature [excluded] present in test data point [True]
406 Text feature [structure] present in test data point [True]
407 Text feature [upon] present in test data point [True]
408 Text feature [like] present in test data point [True]
409 Text feature [percentage] present in test data point [True]
410 Text feature [1997] present in test data point [True]
414 Text feature [conducted] present in test data point [True]
417 Text feature [1b] present in test data point [True]
419 Text feature [essential] present in test data point [True]
421 Text feature [regulatory] present in test data point [True]
422 Text feature [tail] present in test data point [True]
426 Text feature [2002] present in test data point [True]
427 Text feature [94] present in test data point [True]
429 Text feature [driven] present in test data point [True]
435 Text feature [1995] present in test data point [True]
438 Text feature [causes] present in test data point [True]
440 Text feature [sl] present in test data point [True]
442 Text feature [revealed] present in test data point [True]
444 Text feature [perhaps] present in test data point [True]
445 Text feature [mapped] present in test data point [True]
447 Text feature [function] present in test data point [True]
448 Text feature [gel] present in test data point [True]
449 Text feature [colorectal] present in test data point [True]


```
450 Text feature [kb] present in test data point [True]
451 Text feature [type] present in test data point [True]
457 Text feature [nuclear] present in test data point [True]
458 Text feature [resulted] present in test data point [True]
460 Text feature [additional] present in test data point [True]
463 Text feature [reduced] present in test data point [True]
488 Text feature [established] present in test data point [True]
490 Text feature [binding] present in test data point [True]
491 Text feature [interestingly] present in test data point [True]
494 Text feature [2008] present in test data point [True]
496 Text feature [particularly] present in test data point [True]
497 Text feature [assay] present in test data point [True]
Out of the top 500 features 87 are present in query point
```

4.5 Random Forest Classifier

4.5.1. Hyper paramter tuning (With One hot Encoding)

```
In [75]: # -----
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='g
ini', max_depth=None, min_samples_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='aut
o', max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, r
andom_state=None, verbose=0, warm_start=False,
# class_weight=None)

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

# some of attributes of RandomForestClassifier()
# feature_importances_ : array of shape = [n_features]
```

```

# The feature importances (the higher, the more important the feature).

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

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

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

```

```

        print("Log Loss :", log_loss(cv_y, sig_clf_probs))

'''fig, ax = plt.subplots()
features = np.dot(np.array(alpha)[: ,None], np.array(max_depth)[None]).ravel()
ax.plot(features, cv_log_error_array, c='g')
for i, txt in enumerate(np.round(cv_log_error_array,3)):
    ax.annotate((alpha[int(i/2)], max_depth[int(i%2)], str(txt)), (features[i], cv_log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()
'''

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

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

```

for n_estimators = 100 and max depth = 5

Log Loss : 1.18558569367

for n_estimators = 100 and max depth = 10

```

Log Loss : 1.18980709914
for n_estimators = 200 and max depth = 5
Log Loss : 1.17447697162
for n_estimators = 200 and max depth = 10
Log Loss : 1.18124777722
for n_estimators = 500 and max depth = 5
Log Loss : 1.16770463685
for n_estimators = 500 and max depth = 10
Log Loss : 1.17968242912
for n_estimators = 1000 and max depth = 5
Log Loss : 1.16772439938
for n_estimators = 1000 and max depth = 10
Log Loss : 1.17928335521
for n_estimators = 2000 and max depth = 5
Log Loss : 1.16867486376
for n_estimators = 2000 and max depth = 10
Log Loss : 1.17891194565
For values of best estimator = 500 The train log loss is: 0.8576854852
14
For values of best estimator = 500 The cross validation log loss is:
1.16770439012
For values of best estimator = 500 The test log loss is: 1.19719361688

```

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

```

In [76]: # -----
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='g
ini', max_depth=None, min_samples_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='aut
o', max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, r
andom_state=None, verbose=0, warm_start=False,
# class_weight=None)

# Some of methods of RandomForestClassifier()
# fit(X, y, [sample_weight])    Fit the SVM model according to the give
n training data.

```

```
# predict(X)      Perform classification on samples in X.
# predict_proba (X)      Perform classification on samples in X.

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

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

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

Log loss : 1.16768984667

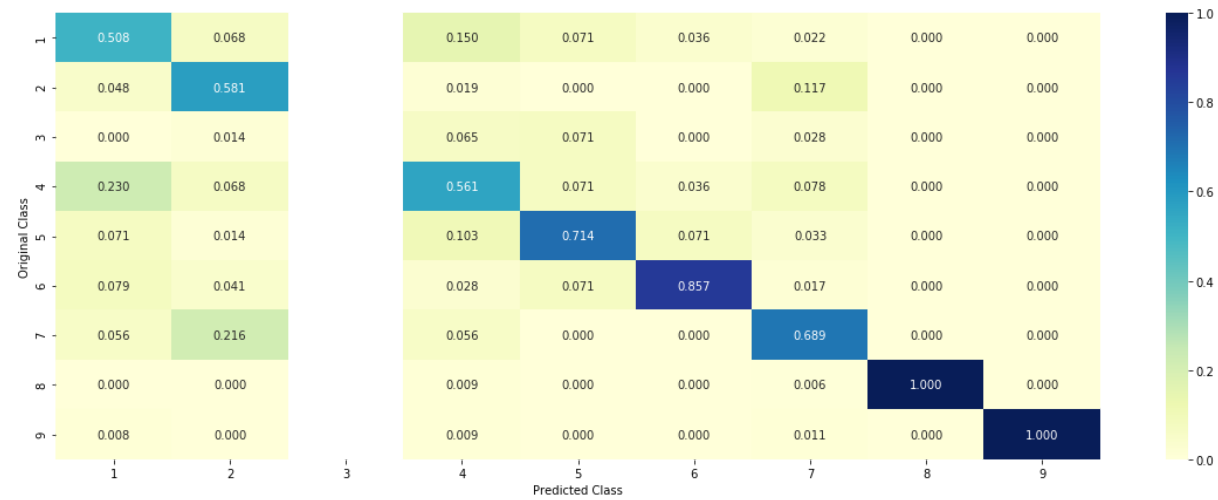
Number of mis-classified points : 0.38345864661654133

----- Confusion matrix -----



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

--



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



4.5.3. Feature Importance

4.5.3.1. Correctly Classified point

```
In [77]: # test_point_index = 10
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_depth=max_depth[int(best_alpha%2)], random_state=42, n_jobs=-1)
clf.fit(train_x_onehotCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_onehotCoding, train_y)

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

Predicted Class : 7

Predicted Class Probabilities: [[0.2792 0.0876 0.0202 0.1405 0.0563 0.0456 0.3547 0.0078 0.008]]

Actual Class : 4

```
-----
0 Text feature [kinase] present in test data point [True]
1 Text feature [activating] present in test data point [True]
2 Text feature [phosphorylation] present in test data point [True]
3 Text feature [inhibitors] present in test data point [True]
4 Text feature [treatment] present in test data point [True]
5 Text feature [function] present in test data point [True]
6 Text feature [activated] 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 [tyrosine] present in test data point [True]
10 Text feature [oncogenic] present in test data point [True]
13 Text feature [inhibitor] present in test data point [True]
14 Text feature [pten] present in test data point [True]
15 Text feature [loss] present in test data point [True]
```

16 Text feature [missense] present in test data point [True]
17 Text feature [therapeutic] present in test data point [True]
19 Text feature [constitutive] present in test data point [True]
20 Text feature [receptor] present in test data point [True]
21 Text feature [activate] present in test data point [True]
22 Text feature [signaling] present in test data point [True]
23 Text feature [akt] present in test data point [True]
25 Text feature [inhibited] present in test data point [True]
26 Text feature [constitutively] present in test data point [True]
27 Text feature [protein] present in test data point [True]
28 Text feature [transforming] present in test data point [True]
29 Text feature [cells] present in test data point [True]
30 Text feature [growth] present in test data point [True]
31 Text feature [therapy] present in test data point [True]
32 Text feature [phosphatase] present in test data point [True]
33 Text feature [proteins] present in test data point [True]
34 Text feature [clinical] present in test data point [True]
35 Text feature [months] present in test data point [True]
36 Text feature [defective] present in test data point [True]
38 Text feature [downstream] present in test data point [True]
39 Text feature [kinases] present in test data point [True]
42 Text feature [classified] present in test data point [True]
48 Text feature [advanced] present in test data point [True]
51 Text feature [57] present in test data point [True]
52 Text feature [treated] present in test data point [True]
53 Text feature [functional] present in test data point [True]
57 Text feature [mammalian] present in test data point [True]
58 Text feature [neutral] present in test data point [True]
60 Text feature [nuclear] present in test data point [True]
61 Text feature [retained] present in test data point [True]
62 Text feature [potential] present in test data point [True]
63 Text feature [ring] present in test data point [True]
64 Text feature [extracellular] present in test data point [True]
66 Text feature [inhibition] present in test data point [True]
68 Text feature [cell] present in test data point [True]
69 Text feature [survival] present in test data point [True]
71 Text feature [brca] present in test data point [True]
73 Text feature [resistance] present in test data point [True]
75 Text feature [functions] present in test data point [True]


```

78 Text feature [carry] present in test data point [True]
80 Text feature [ovarian] present in test data point [True]
82 Text feature [patients] present in test data point [True]
85 Text feature [null] present in test data point [True]
86 Text feature [inactivation] present in test data point [True]
87 Text feature [oncogene] present in test data point [True]
89 Text feature [phospho] present in test data point [True]
92 Text feature [dna] present in test data point [True]
93 Text feature [dose] present in test data point [True]
94 Text feature [expected] present in test data point [True]
95 Text feature [p53] present in test data point [True]
98 Text feature [sensitivity] present in test data point [True]
Out of the top 100 features 65 are present in query point

```

4.5.3.2. Inorrectly 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[test_point_index]),4))
print("Actuall Class :", test_y[test_point_index])
indices = np.argsort(-clf.feature_importances_)
print("-"*50)
get_impfeature_names(indices[:no_feature], test_df['TEXT'].iloc[test_po
int_index],test_df['Gene'].iloc[test_point_index],test_df['Variation'].
iloc[test_point_index], no_feature)

```

```

Predicted Class : 1
Predicted Class Probabilities: [[ 0.2764  0.1706  0.0222  0.1633  0.067
8 0.0662  0.222  0.0073  0.0041]]
Actuall Class : 1
-----
0 Text feature [kinase] present in test data point [True]
1 Text feature [activating] present in test data point [True]
2 Text feature [phosphorylation] present in test data point [True]
5 Text feature [function] present in test data point [True]

```

6 Text feature [activated] 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]
10 Text feature [oncogenic] present in test data point [True]
11 Text feature [erk] present in test data point [True]
12 Text feature [nonsense] present in test data point [True]
15 Text feature [loss] present in test data point [True]
16 Text feature [missense] present in test data point [True]
17 Text feature [therapeutic] present in test data point [True]
21 Text feature [activate] present in test data point [True]
22 Text feature [signaling] present in test data point [True]
27 Text feature [protein] present in test data point [True]
29 Text feature [cells] present in test data point [True]
30 Text feature [growth] present in test data point [True]
32 Text feature [phosphatase] present in test data point [True]
34 Text feature [clinical] present in test data point [True]
39 Text feature [kinases] present in test data point [True]
51 Text feature [57] present in test data point [True]
52 Text feature [treated] present in test data point [True]
53 Text feature [functional] present in test data point [True]
57 Text feature [mammalian] present in test data point [True]
60 Text feature [nuclear] present in test data point [True]
62 Text feature [potential] present in test data point [True]
66 Text feature [inhibition] present in test data point [True]
68 Text feature [cell] present in test data point [True]
69 Text feature [survival] present in test data point [True]
70 Text feature [nscclc] present in test data point [True]
75 Text feature [functions] present in test data point [True]
80 Text feature [ovarian] present in test data point [True]
81 Text feature [frameshift] present in test data point [True]
82 Text feature [patients] present in test data point [True]
86 Text feature [inactivation] present in test data point [True]
89 Text feature [phospho] present in test data point [True]
92 Text feature [dna] present in test data point [True]
94 Text feature [expected] present in test data point [True]
95 Text feature [p53] present in test data point [True]
97 Text feature [predicted] present in test data point [True]
98 Text feature [sensitivity] present in test data point [True]
Out of the top 100 features 42 are present in query point

4.5.3. Hyper paramter tuning (With Response Coding)

```
In [79]: # -----
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='g
ini', max_depth=None, min_samples_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='aut
o', max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, r
andom_state=None, verbose=0, warm_start=False,
# class_weight=None)

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

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

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

# find more about CalibratedClassifierCV here at http://scikit-learn.or
g/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.h
tml
# -----
# default paramters
# sklearn.calibration.CalibratedClassifierCV(base_estimator=None, metho
d='sigmoid', cv=3)
#
# some of the methods of CalibratedClassifierCV()
# fit(X, y[, sample_weight])    Fit the calibrated model
```

```

# get_params([deep])    Get parameters for this estimator.
# predict(X)            Predict the target of new samples.
# predict_proba(X)      Posterior probabilities of classification
#-----
# video link:
#-----

alpha = [10,50,100,200,500,1000]
max_depth = [2,3,5,10]
cv_log_error_array = []
for i in alpha:
    for j in max_depth:
        print("for n_estimators =", i,"and max depth = ", j)
        clf = RandomForestClassifier(n_estimators=i, criterion='gini',
max_depth=j, random_state=42, n_jobs=-1)
        clf.fit(train_x_responseCoding, train_y)
        sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
        sig_clf.fit(train_x_responseCoding, train_y)
        sig_clf_probs = sig_clf.predict_proba(cv_x_responseCoding)
        cv_log_error_array.append(log_loss(cv_y, sig_clf_probs, labels=
clf.classes_, eps=1e-15))
        print("Log Loss :",log_loss(cv_y, sig_clf_probs))
    ...

fig, ax = plt.subplots()
features = np.dot(np.array(alpha)[: ,None],np.array(max_depth)[None]).ra
vel()
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)), (featur
es[i],cv_log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()
...

best_alpha = np.argmin(cv_log_error_array)
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/4)], cri

```

```

terion='gini', max_depth=max_depth[int(best_alpha%4)], random_state=42,
n_jobs=-1)
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)

predict_y = sig_clf.predict_proba(train_x_responseCoding)
print('For values of best alpha = ', alpha[int(best_alpha/4)], "The tra
in log loss is:", log_loss(y_train, predict_y, labels=clf.classes_, eps=
1e-15))
predict_y = sig_clf.predict_proba(cv_x_responseCoding)
print('For values of best alpha = ', alpha[int(best_alpha/4)], "The cro
ss validation log loss is:", log_loss(y_cv, predict_y, labels=clf.classe
s_, eps=1e-15))
predict_y = sig_clf.predict_proba(test_x_responseCoding)
print('For values of best alpha = ', alpha[int(best_alpha/4)], "The tes
t log loss is:", log_loss(y_test, predict_y, labels=clf.classes_, eps=1e
-15))

```

```

for n_estimators = 10 and max depth = 2
Log Loss : 2.07046324995
for n_estimators = 10 and max depth = 3
Log Loss : 1.70430353017
for n_estimators = 10 and max depth = 5
Log Loss : 1.54295699236
for n_estimators = 10 and max depth = 10
Log Loss : 1.98689124705
for n_estimators = 50 and max depth = 2
Log Loss : 1.73712620922
for n_estimators = 50 and max depth = 3
Log Loss : 1.54032766207
for n_estimators = 50 and max depth = 5
Log Loss : 1.56409335312
for n_estimators = 50 and max depth = 10
Log Loss : 1.96474512195
for n_estimators = 100 and max depth = 2
Log Loss : 1.6074701839
for n_estimators = 100 and max depth = 3
Log Loss : 1.55012054876
for n_estimators = 100 and max depth = 5

```

```

Log Loss : 1.47093081171
for n_estimators = 100 and max depth = 10
Log Loss : 1.89482073231
for n_estimators = 200 and max depth = 2
Log Loss : 1.67221942662
for n_estimators = 200 and max depth = 3
Log Loss : 1.55934879172
for n_estimators = 200 and max depth = 5
Log Loss : 1.52174503341
for n_estimators = 200 and max depth = 10
Log Loss : 1.84659028089
for n_estimators = 500 and max depth = 2
Log Loss : 1.74108279809
for n_estimators = 500 and max depth = 3
Log Loss : 1.62269549187
for n_estimators = 500 and max depth = 5
Log Loss : 1.5184303446
for n_estimators = 500 and max depth = 10
Log Loss : 1.92095598328
for n_estimators = 1000 and max depth = 2
Log Loss : 1.71888308571
for n_estimators = 1000 and max depth = 3
Log Loss : 1.62320594454
for n_estimators = 1000 and max depth = 5
Log Loss : 1.50118225484
for n_estimators = 1000 and max depth = 10
Log Loss : 1.87884073401
For values of best alpha = 100 The train log loss is: 0.0504528058041
For values of best alpha = 100 The cross validation log loss is: 1.470
93081171
For values of best alpha = 100 The test log loss is: 1.35205612677

```

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

```

In [80]: # -----
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='g
ini', max_depth=None, min_samples_split=2,

```

```

# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',
# max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1,
# random_state=None, verbose=0, warm_start=False,
# class_weight=None)

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

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

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

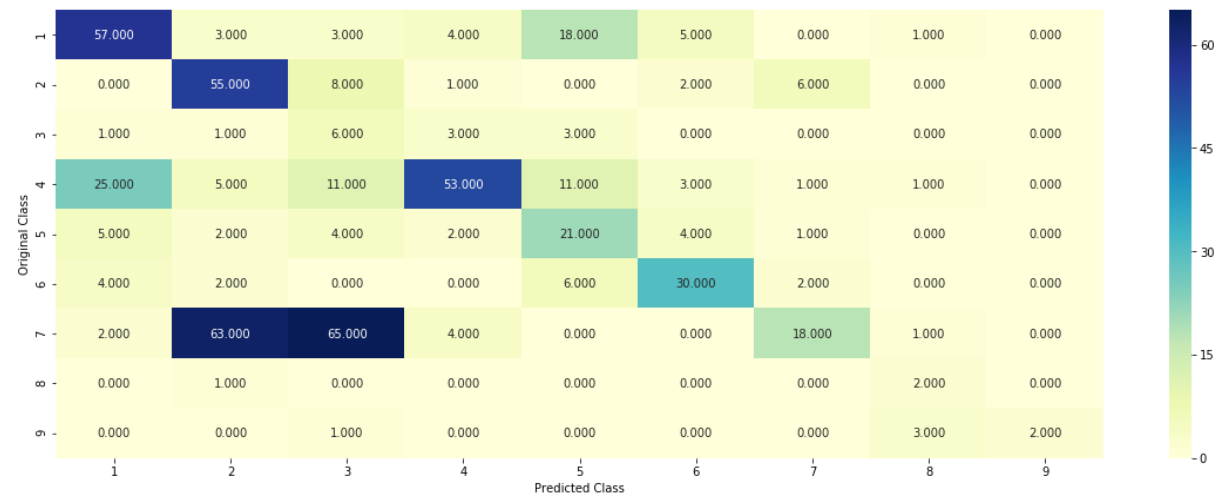
clf = RandomForestClassifier(max_depth=max_depth[int(best_alpha%4)], n_estimators=alpha[int(best_alpha/4)], criterion='gini', max_features='auto', random_state=42)
predict_and_plot_confusion_matrix(train_x_responseCoding, train_y, cv_x_responseCoding, cv_y, clf)

```

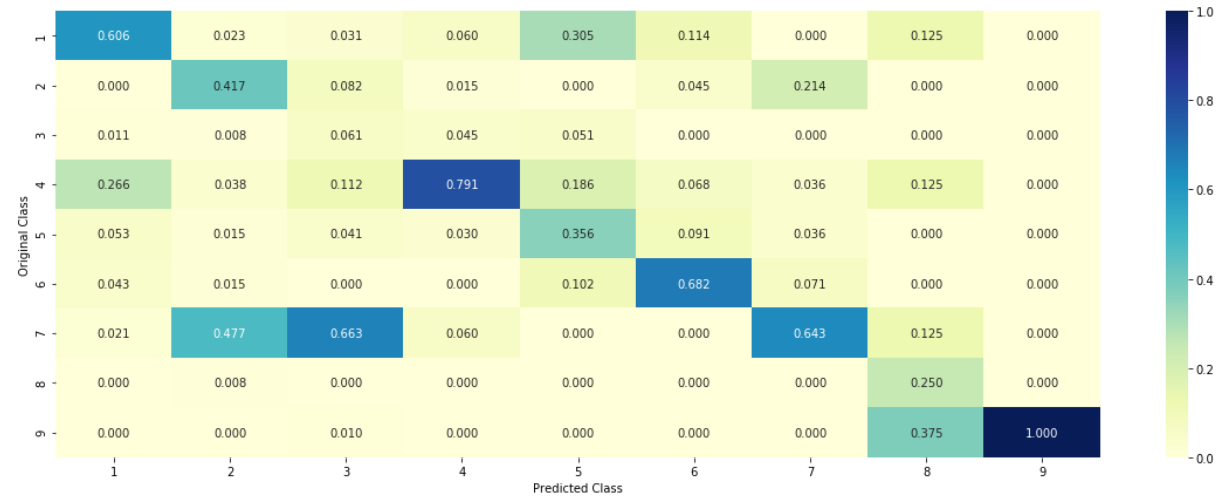
Log loss : 1.47093081171

Number of mis-classified points : 0.5413533834586466

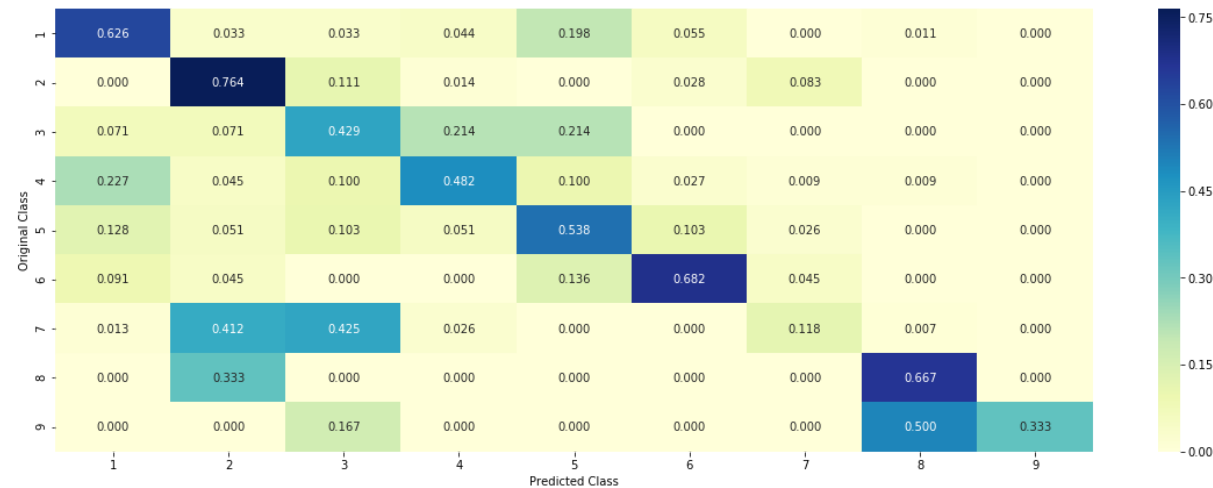
----- Confusion matrix -----



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



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



4.5.5. Feature Importance

4.5.5.1. Correctly Classified point

```
In [81]: clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/4)], criterion='gini', max_depth=max_depth[int(best_alpha%4)], random_state=42, n_jobs=-1)
clf.fit(train_x_responseCoding, train_y)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(train_x_responseCoding, train_y)

test_point_index = 1
no_feature = 27
predicted_cls = sig_clf.predict(test_x_responseCoding[test_point_index].reshape(1,-1))
print("Predicted Class :", predicted_cls[0])
print("Predicted Class Probabilities:", np.round(sig_clf.predict_proba(test_x_responseCoding[test_point_index].reshape(1,-1)),4))
```

```

print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.feature_importances_)
print("-"*50)
for i in indices:
    if i<9:
        print("Gene is important feature")
    elif i<18:
        print("Variation is important feature")
    else:
        print("Text is important feature")

```

```

Predicted Class : 1
Predicted Class Probabilities: [[ 0.9557  0.0014  0.001  0.0348  0.001
 2  0.0021  0.0011  0.0013  0.0013]]
Actual Class : 4

```

```

-----
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
Gene is important feature
Gene is important feature
Variation is important feature
Gene is important feature
Text is important feature
Variation is important feature
Gene is important feature
Text is important feature
Variation is important feature
Gene is important feature
Text is important feature

```

```
Gene is important feature
Text is important feature
Gene is important feature
```

4.5.5.2. Incorrectly Classified point

```
In [82]: 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_responseCoding[test_point_index].reshape(1,-1)),4))
print("Actual Class :", test_y[test_point_index])
indices = np.argsort(-clf.feature_importances_)
print("-"*50)
for i in indices:
    if i<9:
        print("Gene is important feature")
    elif i<18:
        print("Variation is important feature")
    else:
        print("Text is important feature")
```

```
Predicted Class : 4
Predicted Class Probabilities: [[ 0.196  0.0016  0.001  0.7942  0.001
 0.0025  0.0011  0.0015  0.0012]]
Actual Class : 1
```

```
-----
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
```

```
text is important feature
Gene is important feature
Text is important feature
Text is important feature
Gene is important feature
Gene is important feature
Variation is important feature
Gene is important feature
Text is important feature
Variation is important feature
Gene is important feature
Text is important feature
Variation is important feature
Gene is important feature
Text is important feature
Gene is important feature
Text is important feature
Gene is important feature
```

4.7 Stack the models

4.7.1 testing with hyper parameter tuning

```
In [83]: # read more about SGDClassifier() at http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.SGDClassifier.html
# -----
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=None, tol=None,
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5,
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, ...])    Fit linear model with Stochastic Gradient Descent.
```

```

# predict(X)    Predict class labels for samples in X.

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

# read more about support vector machines with linear kernels here http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
# -----
# default parameters
# SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001,
# cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)

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

# read more about support vector machines with linear kernels here http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
# -----
# default parameters
# sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2,
# min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0,
# min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False,
# class_weight=None)

```

```

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

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

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

clf1 = SGDClassifier(alpha=0.001, penalty='l2', loss='log', class_weight='balanced', random_state=0)
clf1.fit(train_x_onehotCoding, train_y)
sig_clf1 = CalibratedClassifierCV(clf1, method="sigmoid")

clf2 = SGDClassifier(alpha=1, penalty='l2', loss='hinge', class_weight='balanced', random_state=0)
clf2.fit(train_x_onehotCoding, train_y)
sig_clf2 = CalibratedClassifierCV(clf2, method="sigmoid")

clf3 = MultinomialNB(alpha=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_x_onehotCoding))))
sig_clf2.fit(train_x_onehotCoding, train_y)
print("Support vector machines : Log Loss: %0.2f" % (log_loss(cv_y, sig_clf2.predict_proba(cv_x_onehotCoding))))
sig_clf3.fit(train_x_onehotCoding, train_y)

```

```

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

```

Logistic Regression : Log Loss: 0.99
 Support vector machines : Log Loss: 1.84
 Naive Bayes : Log Loss: 1.21

```

-----
Stacking Classifier : for the value of alpha: 0.000100 Log Loss: 2.177
Stacking Classifier : for the value of alpha: 0.001000 Log Loss: 2.030
Stacking Classifier : for the value of alpha: 0.010000 Log Loss: 1.493
Stacking Classifier : for the value of alpha: 0.100000 Log Loss: 1.158
Stacking Classifier : for the value of alpha: 1.000000 Log Loss: 1.388
Stacking Classifier : for the value of alpha: 10.000000 Log Loss: 1.850

```

4.7.2 testing the model with the best hyper parameters

```

In [84]: lr = LogisticRegression(C=0.1)
sclf = StackingClassifier(classifiers=[sig_clf1, sig_clf2, sig_clf3], meta_classifier=lr, use_probas=True)
sclf.fit(train_x_onehotCoding, train_y)

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

log_error = log_loss(cv_y, sclf.predict_proba(cv_x_onehotCoding))

```

```

print("Log loss (CV) on the stacking classifier :",log_error)

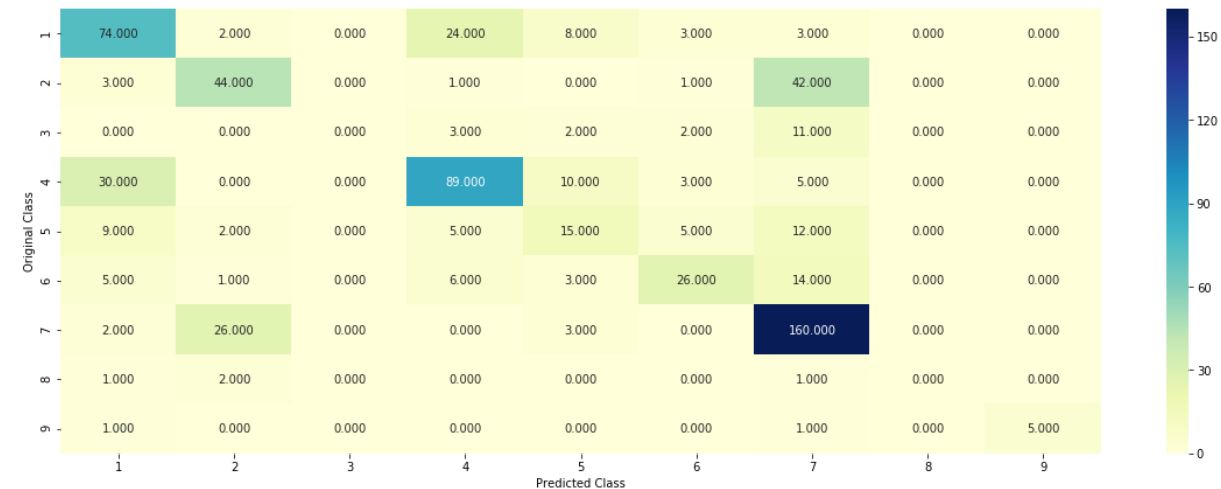
log_error = log_loss(test_y, sclf.predict_proba(test_x_onehotCoding))
print("Log loss (test) on the stacking classifier :",log_error)

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

```

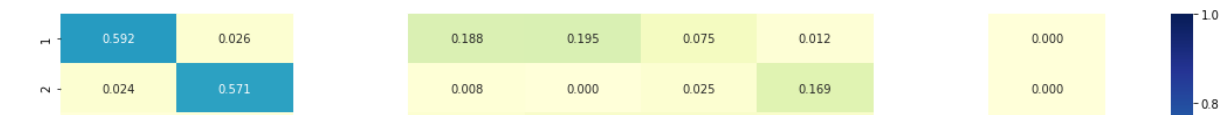
Log loss (train) on the stacking classifier : 0.560751825611
 Log loss (CV) on the stacking classifier : 1.15819186597
 Log loss (test) on the stacking classifier : 1.15940485015
 Number of missclassified point : 0.37894736842105264

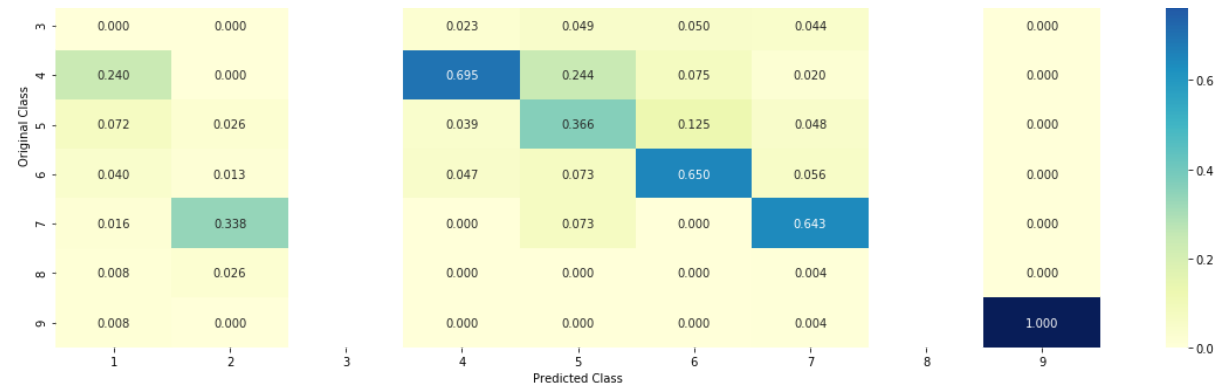
----- Confusion matrix -----



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

--





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



4.7.3 Maximum Voting classifier

```
In [85]: #Refer:http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html
from sklearn.ensemble import VotingClassifier
vclf = VotingClassifier(estimators=[('lr', sig_clf1), ('svc', sig_clf2), ('rf', sig_clf3)], voting='soft')
vclf.fit(train_x_onehotCoding, train_y)
```

```

print("Log loss (train) on the VotingClassifier :", log_loss(train_y, v
clf.predict_proba(train_x_onehotCoding)))
print("Log loss (CV) on the VotingClassifier :", log_loss(cv_y, vclf.pr
edict_proba(cv_x_onehotCoding)))
print("Log loss (test) on the VotingClassifier :", log_loss(test_y, vcl
f.predict_proba(test_x_onehotCoding)))
print("Number of missclassified point :", np.count_nonzero((vclf.predic
t(test_x_onehotCoding)- test_y))/test_y.shape[0])
plot_confusion_matrix(test_y=test_y, predict_y=vclf.predict(test_x_oneh
otCoding))

```

Log loss (train) on the VotingClassifier : 0.824152170697

Log loss (CV) on the VotingClassifier : 1.15953161575

Log loss (test) on the VotingClassifier : 1.19273200743

Number of missclassified point : 0.3804511278195489

----- Confusion matrix -----



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

--



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

