Personalized Cancer Diagnosis

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import warnings
import time
import math
import re
warnings.simplefilter('ignore')
from collections import Counter, defaultdict
from scipy.sparse import hstack
from scipy.sparse import vstack
from mlxtend.classifier import StackingClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.svm import SVC
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
```

In [2]:

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Reading Data

In [3]:

```
read_initital_data = pd.read_csv("../PERSONALISED_TRAINING_DATA/DATA_TRAIN")
```

In [4]:

```
read_initital_data.head()
```

Out[4]:

	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

In [5]:

```
read_initital_data.shape
```

Out[5]:

(3321, 4)

(3321, 4)

In [6]:

```
read_initital_data_text = pd.read_csv("../PERSONALISED_TRAINING_DATA/TEXT_TRAIN_DATA", sep = "\|\|"
, names = ["ID", "Text"])
```

In [7]:

```
read_initital_data_text = read_initital_data_text.drop(0, axis = 0)
```

In [8]:

```
read_initital_data_text = read_initital_data_text.reset_index(drop = True)
```

In [9]:

```
read_initital_data_text.head()
```

Out[9]:

	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...

In [10]:

```
read_initital_data_text.shape
```

Out[10]:

(3321, 2)

In [11]:

```
read_initital_data_text = read_initital_data_text.drop(["ID"], axis = 1)
```

In [12]:

```
new_data_final = pd.concat([read_initital_data, read_initital_data_text], axis = 1)
```

In [13]:

```
new_data_final.head()
```

Out[13]:

	ID	Gene	Variation	Class	Text
0	0	FAM58A	Truncating Mutations	1	Cyclin-dependent kinases (CDKs) regulate a var...
1	1	CBL	W802*	2	Abstract Background Non-small cell lung canc...
2	2	CBL	Q249E	2	Abstract Background Non-small cell lung canc...
3	3	CBL	N454D	3	Recent evidence has demonstrated that acquired...
4	4	CBL	L399V	4	Oncogenic mutations in the monomeric Casitas B...

In [14]:

```
new_data_final_label = new_data_final["Class"]
```

In [15]:

```
new_data_final = new_data_final.drop(["Class"], axis = 1)
```

In [16]:

```
new_data_final.head()
```

Out[16]:

	ID	Gene	Variation	Text
0	0	FAM58A	Truncating Mutations	Cyclin-dependent kinases (CDKs) regulate a var...
1	1	CBL	W802*	Abstract Background Non-small cell lung canc...
2	2	CBL	Q249E	Abstract Background Non-small cell lung canc...
3	3	CBL	N454D	Recent evidence has demonstrated that acquired...
4	4	CBL	L399V	Oncogenic mutations in the monomeric Casitas B...

In [17]:

```
new_data_final = pd.concat([new_data_final, new_data_final_label], axis = 1)
```

In [18]:

```
new_data_final.head()
```

Out[18]:

	ID	Gene	Variation	Text	Class
0	0	FAM58A	Truncating Mutations	Cyclin-dependent kinases (CDKs) regulate a var...	1
1	1	CBL	W802*	Abstract Background Non-small cell lung canc...	2
2	2	CBL	Q249E	Abstract Background Non-small cell lung canc...	2
3	3	CBL	N454D	Recent evidence has demonstrated that acquired...	3
4	4	CBL	L399V	Oncogenic mutations in the monomeric Casitas B...	4

In [19]:

```
for indx, txt in new_data_final.iterrows():
    if type(txt["Text"]) is not str:
        print(indx)
```

```
1109
1277
1407
1639
2755
```

In [20]:

```
for indx, txt in new_data_final.iterrows():
    if type(txt["Text"]) is not str:
        new_data_final.drop(indx, axis = 0, inplace = True)
```

In [21]:

```
new_data_final.shape
```

Out[21]:

```
(3316, 5)
```

In [22]:

```
def final_preprocess_text_nlp(data_text, idxno, colval):
    if type(data_text) is not int:
        textString = ""
        #special symbol removal
        data_text1 = re.sub('[^a-zA-Z0-9\n]', ' ', data_text)
        #whitespace removal
        data_text2 = re.sub('\s+', ' ', data_text1)
        #conversion lowercase
        str_lower_text += data_text2.lower()
        new_data_final[colval][idxno] = str_lower_text
```

In [23]:

```
time_tick = time.clock()
for index, txtinfdata in new_data_final.iterrows():
    final_preprocess_text_nlp(txtinfdata["Text"], index, "Text")
print("Total time for Preprocessing the text data = "+str(time.clock() - time_tick)+"Sec")
```

```
Total time for Preprocessing the text data = 174.12587938723027Sec
```

In [24]:

```
new_data_final.head()
```

Out[24]:

	ID	Gene	Variation	Text	Class
0	0	FAM58A	Truncating Mutations	cyclin dependent kinases cdks regulate a varie...	1
1	1	CBL	W802*	abstract background non small cell lung cance...	2
2	2	CBL	Q249E	abstract background non small cell lung cance...	2
3	3	CBL	N454D	recent evidence has demonstrated that acquired...	3
4	4	CBL	L399V	oncogenic mutations in the monomeric casitas b...	4

In [25]:

```
new_data_final_label = new_data_final["Class"]
```

In [26]:

```
new_data_final.shape, new_data_final_label.shape
```

Out[26]:

```
((3316, 5), (3316,))
```

Train Test split with respect to cross-validation by doing 64,20,16 splits

In [27]:

```
#underscore replacement
```

```
new_data_final["Gene"] = new_data_final["Gene"].str.replace("\s+", "_")
new_data_final["Variation"] = new_data_final["Variation"].str.replace("\s+", "_")
```

In [28]:

```
new_data_final.head()
```

Out[28]:

	ID	Gene	Variation	Text	Class
0	0	FAM58A	Truncating_Mutations	cyclin dependent kinases cdks regulate a varie...	1
1	1	CBL	W802*	abstract background non small cell lung cance...	2
2	2	CBL	Q249E	abstract background non small cell lung cance...	2
3	3	CBL	N454D	recent evidence has demonstrated that acquired...	3
4	4	CBL	L399V	oncogenic mutations in the monomeric casitas b...	4

In [29]:

```
X_Train, X_Test, Y_Train, Y_Test = train_test_split(new_data_final, new_data_final_label, stratify=
new_data_final_label, test_size=0.2)
final_train, cross_val_data, cross_val_y, label_cross_val_data = train_test_split(X_Train, Y_Train,
stratify=Y_Train, test_size=0.2)
```

In [30]:

```
X_Train.shape, X_Test.shape, Y_Train.shape, Y_Test.shape
```

Out[30]:

```
((2652, 5), (664, 5), (2652,), (664,))
```

In [31]:

```
print('Number of data points in train data:', final_train.shape[0])
print('Number of data points in test data:', X_Test.shape[0])
print('Number of data points in cross validation data:', cross_val_data.shape[0])
```

```
Number of data points in train data: 2121
Number of data points in test data: 664
Number of data points in cross validation data: 531
```

Distribution of Class labels in Train, CV and Test data

In [32]:

```
new_train_dist_class = final_train["Class"].value_counts().sort_index()
cross_val_dist_class = cross_val_data["Class"].value_counts().sort_index()
new_test_dist_class = X_Test["Class"].value_counts().sort_index()
```

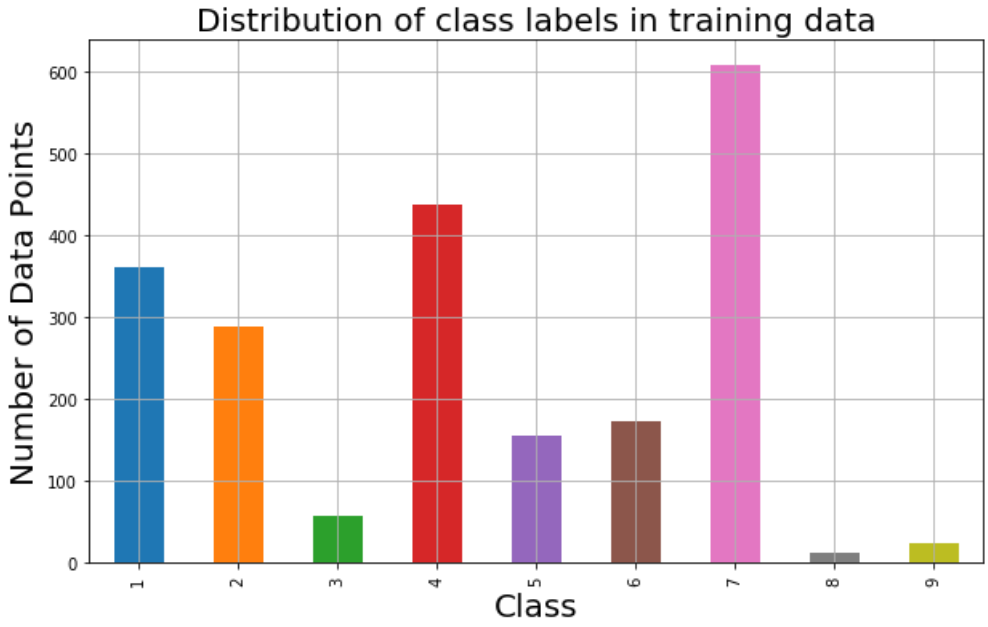
In [33]:

```
new_train_dist_sorted = sorted(new_train_dist_class.items(), key = lambda d: d[1], reverse = True)
cross_val_dist_sorted = sorted(cross_val_dist_class.items(), key = lambda d: d[1], reverse = True)
new_test_dist_sorted = sorted(new_test_dist_class.items(), key = lambda d: d[1], reverse = True)
```

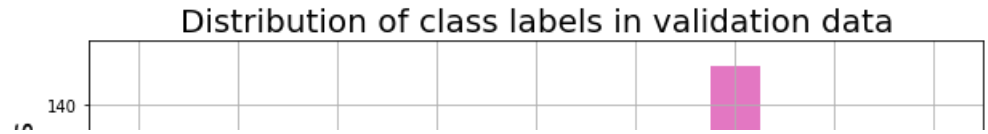
In [34]:

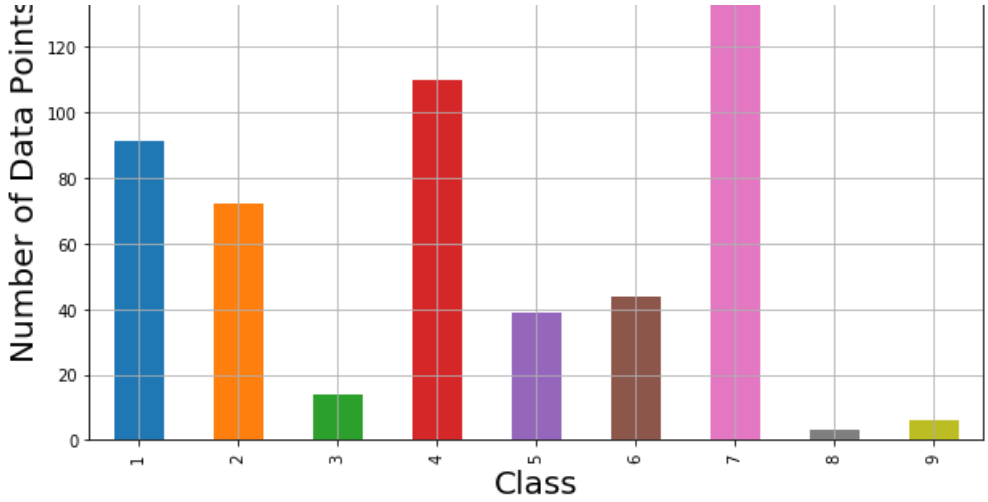
```
plt.figure(figsize = (10, 6))
new_train_dist_class.plot(kind = "bar")
plt.grid()
plt.title("Distribution of class labels in training data", fontsize = 20)
plt.xlabel("Class", fontsize = 20)
```

```
plt.ylabel("Number of Data Points", fontsize = 20)
plt.show()

for i in new_train_dist_sorted:
    print("Number of training data points in class "+str(i[0])+" = "+str(i[1])+ "("+str(np.round(((
i[1]/final_train.shape[0])*100), 4))+"%)")

print("-"*80)

plt.figure(figsize = (10, 6))
cross_val_dist_class.plot(kind = "bar")
plt.grid()
plt.title("Distribution of class labels in validation data", fontsize = 20)
plt.xlabel("Class", fontsize = 20)
plt.ylabel("Number of Data Points", fontsize = 20)
plt.show()

for i in cross_val_dist_sorted:
    print("Number of CV data points in class "+str(i[0])+" = "+str(i[1])+ "("+str(np.round(((i[1]/c
ross_val_data.shape[0])*100), 4))+"%)")

print("-"*80)

plt.figure(figsize = (10, 6))
new_test_dist_sorted.plot(kind = "bar")
plt.grid()
plt.title("Distribution of class labels in test data", fontsize = 20)
plt.xlabel("Class", fontsize = 20)
plt.ylabel("Number of Data Points", fontsize = 20)
plt.show()

for i in TestData_distribution_sorted:
    print("Number of test data points in class "+str(i[0])+" = "+str(i[1])+ "("+str(np.round(((i[1]
/X_Test.shape[0])*100), 4))+"%)")
```

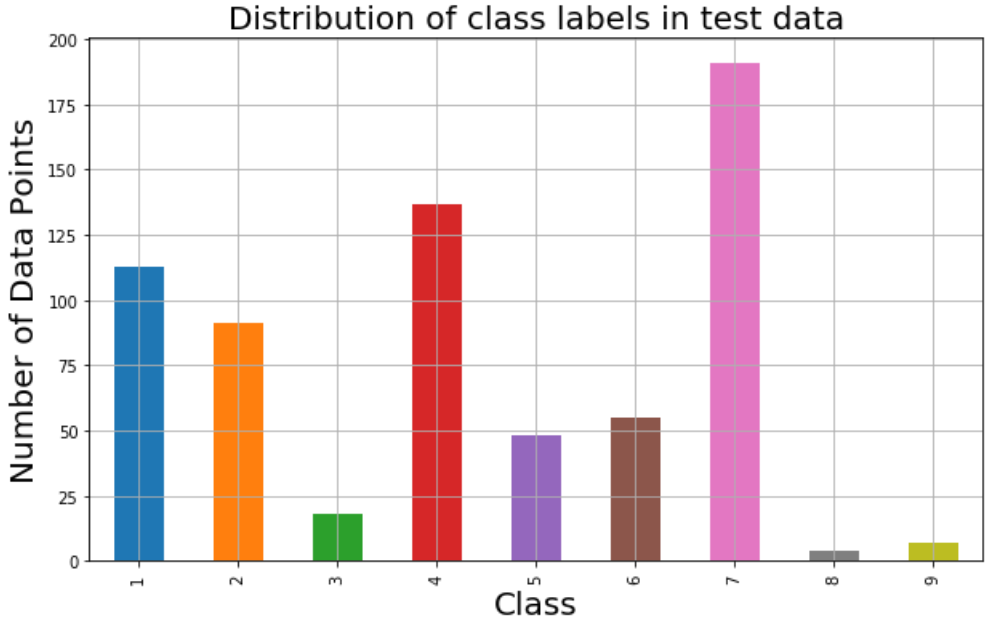



```
Number of training data points in class 7 = 609(28.7129%)
Number of training data points in class 4 = 439(20.6978%)
Number of training data points in class 1 = 362(17.0674%)
Number of training data points in class 2 = 289(13.6256%)
Number of training data points in class 6 = 174(8.2037%)
Number of training data points in class 5 = 155(7.3079%)
Number of training data points in class 3 = 57(2.6874%)
Number of training data points in class 9 = 24(1.1315%)
Number of training data points in class 8 = 12(0.5658%)
--------------------------------------------------------------------------------
```






```
Number of CV data points in class 7 = 152(28.6252%)
Number of CV data points in class 4 = 110(20.7156%)
Number of CV data points in class 1 = 91(17.1375%)
Number of CV data points in class 2 = 72(13.5593%)
Number of CV data points in class 6 = 44(8.2863%)
Number of CV data points in class 5 = 39(7.3446%)
Number of CV data points in class 3 = 14(2.6365%)
Number of CV data points in class 9 = 6(1.1299%)
Number of CV data points in class 8 = 3(0.565%)
------------------------------------------------------------------------------------
```




```
Number of test data points in class 7 = 191(28.7651%)
Number of test data points in class 4 = 137(20.6325%)
Number of test data points in class 1 = 113(17.0181%)
Number of test data points in class 2 = 91(13.7048%)
Number of test data points in class 6 = 55(8.2831%)
Number of test data points in class 5 = 48(7.2289%)
Number of test data points in class 3 = 18(2.7108%)
Number of test data points in class 9 = 7(1.0542%)
Number of test data points in class 8 = 4(0.6024%)
```

Predict using randomly generated model

In [35]:

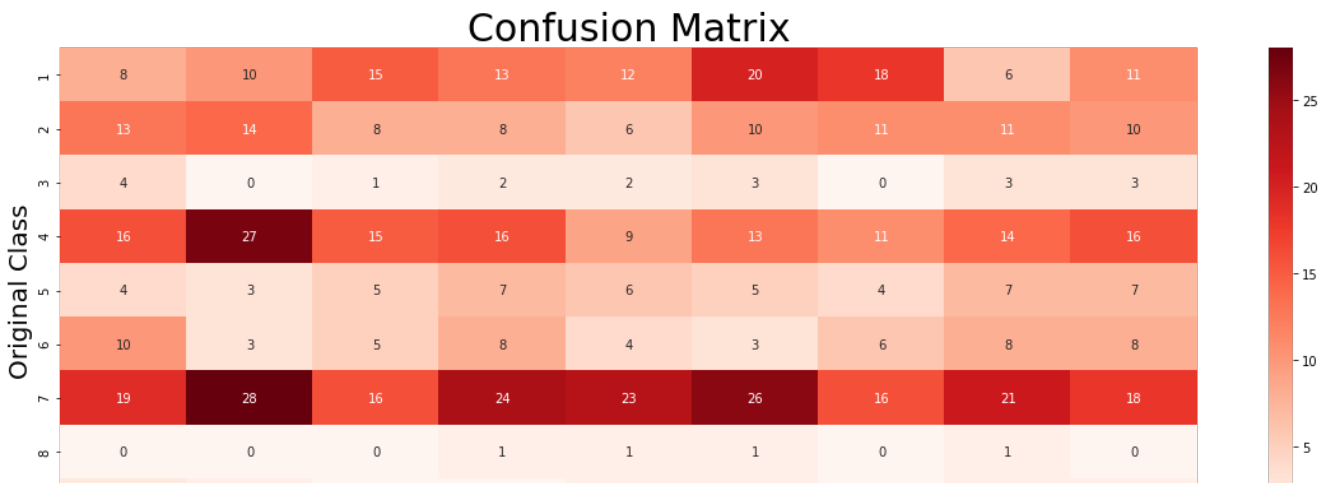
```
def generate_conf_mat(testlabel_y, lab_pred):
    calmatrx = confusion_matrix(testlabel_y, lab_pred)

    precval = calmatrx/calmatrx.sum(axis = 0)
```

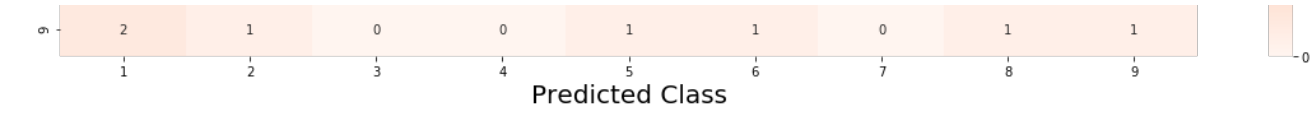
```
    rec_val = (calmatrx.T/calmatrx.sum(axis = 1)).T


    labels = [i for i in range(1, 10)]

    plt.figure(figsize=(20,7))
    sns.heatmap(calmatrx, cmap = "Reds", annot = True, xticklabels=labels, yticklabels=labels)
    plt.title("Confusion Matrix", fontsize = 30)
    plt.xlabel('Predicted Class', fontsize = 20)
    plt.ylabel('Original Class', fontsize = 20)
    plt.show()

    print("-"*125)

    plt.figure(figsize=(20,7))
    sns.heatmap(precval, cmap = "Reds", annot = True, fmt = ".4f", xticklabels=labels, yticklabels=
labels)
    plt.title("Precision Matrix", fontsize = 30)
    plt.xlabel('Predicted Class', fontsize = 20)
    plt.ylabel('Original Class', fontsize = 20)
    plt.show()

    print("-"*125)

    plt.figure(figsize=(20,7))
    sns.heatmap(rec_val, cmap = "Reds", annot = True, fmt = ".3f", xticklabels=labels, yticklabels=
labels)
    plt.title("Recall Matrix", fontsize = 30)
    plt.xlabel('Predicted Class', fontsize = 20)
    plt.ylabel('Original Class', fontsize = 20)
    plt.show()
```

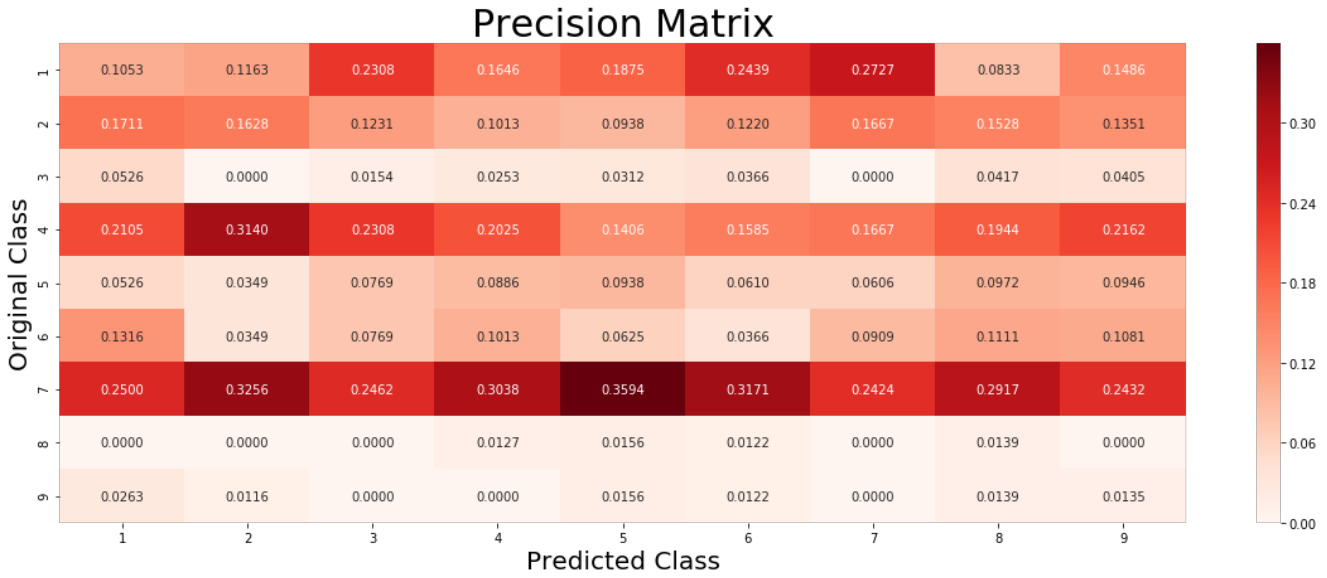
In [36]:

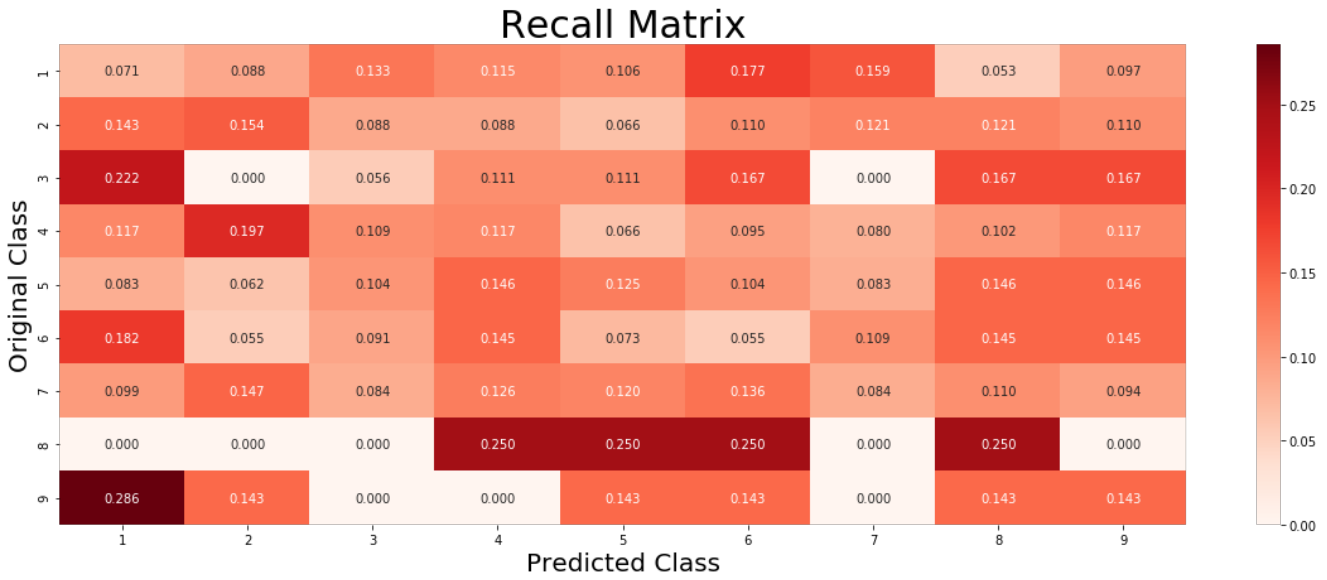
```
len_testdata = X_Test.shape[0]
len_crossval_data = cross_val_data.shape[0]

cross_val_final_pred = np.zeros((len_crossval_data,9))
for i in range(len_crossval_data):
    cross_val_prob = np.random.rand(1,9)
    cross_val_final_pred[i] = (cross_val_prob/sum(sum(cross_val_prob)))[0]
print("Log loss on Cross Validation Data using Random Model "+str(log_loss(label_cross_val_data,cr
oss_val_final_pred)))

test_final_pred_prob = np.zeros((len_testdata,9))
for i in range(len_testdata):
    rand_probs_test = np.random.rand(1,9)
    test_final_pred_prob[i] = (rand_probs_test/sum(sum(rand_probs_test)))[0]
print("Log loss on Test Data using Random Model "+str(log_loss(Y_Test, test_final_pred_prob)))

pred_test_labels = np.argmax(test_final_pred_prob, axis = 1)
generate_conf_mat(Y_Test, pred_test_labels+1)
```

```
Log loss on Cross Validation Data using Random Model 2.4908584438935715
Log loss on Test Data using Random Model 2.5051221373926853
```









In [37]:

```
def feat_code_resp(alpha, feature, df):

    value_count = final_train[feature].value_counts()
    rc_values = dict()

    for i, denominator in value_count.items():
        vec = []
        for k in range(1, 10):
            cls_cnt = final_train.loc[(final_train["Class"]==k) & (final_train[feature]==i)]
            vec.append((cls_cnt.shape[0] + 10*alpha)/denominator + (90*alpha))
        rc_values[i] = vec
    return rc_values


def res_val(alpha, feature, df):
    resp_values = feat_code_resp(alpha, feature, df)
```

```
    resp_val_count = final_train[feature].value_counts()
    res_feat_val = []
    for index, row in df.iterrows():
        if row[feature] in dict(resp_val_count).keys():
            res_feat_val.append(resp_values[row[feature]])
        else:
            res_feat_val.append([1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9,1/9])
    return res_feat_val
```

By doing laplace smoothing, we can calculate laplace smoothing.

Univariate Analysis

In [38]:

```
number_gene_distinct = final_train["Gene"].value_counts()
print("Number of Unique genes = "+str(number_gene_distinct.shape[0]))
print(number_gene_distinct.head(10))
```

```
Number of Unique genes = 231
BRCA1     176
TP53      111
EGFR       81
PTEN       79
BRCA2      77
BRAF       58
KIT        57
ERBB2      46
ALK        42
PDGFRA     42
Name: Gene, dtype: int64
```

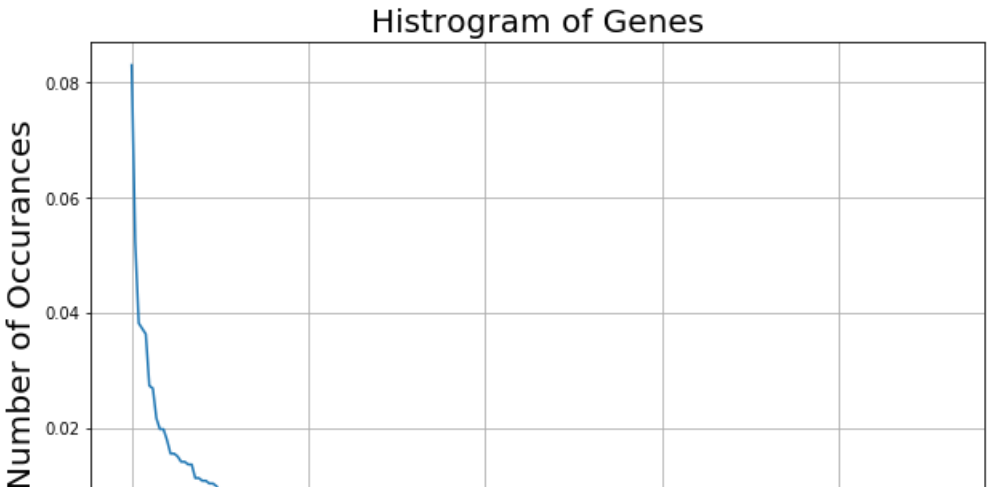
In [39]:

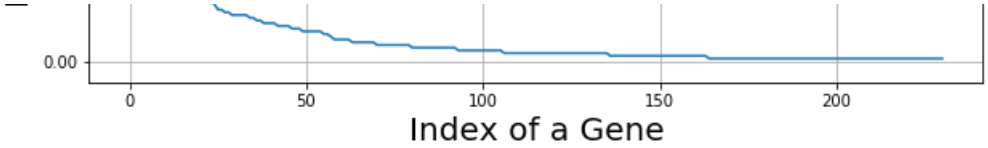
```
print("Ans: There are", number_gene_distinct.shape[0] ,"different categories of genes in the train data, and they are distibuted as follows:")
```

```
Ans: There are 231 different categories of genes in the train data, and they are distibuted as follows:
```

In [40]:

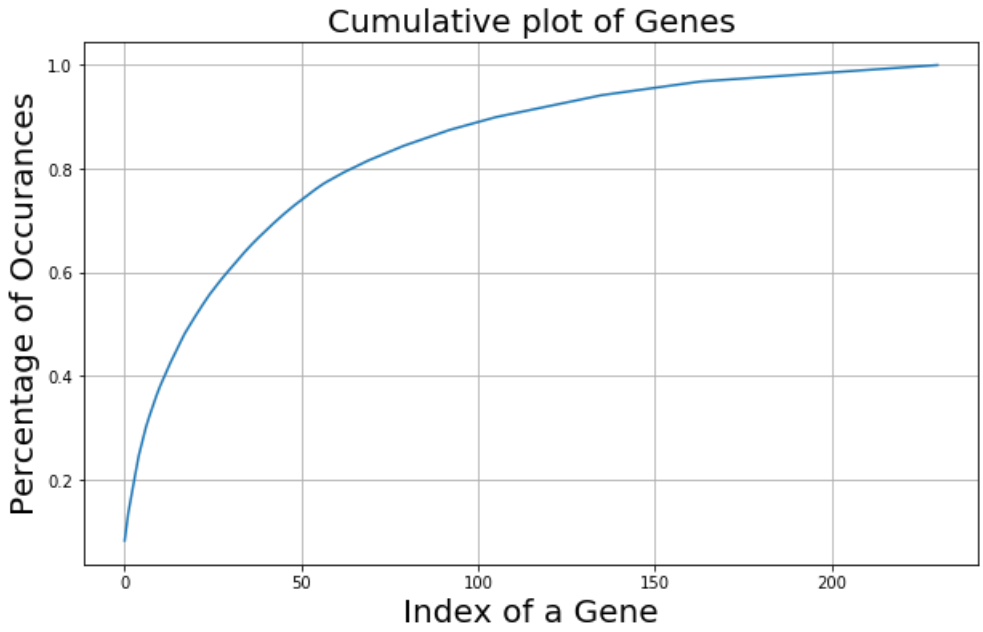
```
sum_gene = sum(number_gene_distinct.values)
dist_gen = number_gene_distinct.values/sum_gene
plt.figure(figsize = (10, 6))
plt.plot(dist_gen)
plt.title("Histrogram of Genes", fontsize = 20)
plt.xlabel('Index of a Gene', fontsize = 20)
plt.ylabel('Number of Occurances', fontsize = 20)
plt.grid()
plt.show()
```





In [41]:

```
sum_gene = sum(number_gene_distinct.values)
dist_gen = number_gene_distinct.values/sum_gene
plt.figure(figsize = (10, 6))
plt.plot(np.cumsum(dist_gen))
plt.title("Cumulative plot of Genes", fontsize = 20)
plt.xlabel('Index of a Gene', fontsize = 20)
plt.ylabel('Percentage of Occurances', fontsize = 20)
plt.grid()
plt.show()
```



Doing One hot encoding

In [42]:

```
alpha = 1
gene_train_encoding = np.array(res_val(alpha, "Gene", final_train))
gene_test_encoding = np.array(res_val(alpha, "Gene", X_Test))
gene_crossval_encoding = np.array(res_val(alpha, "Gene", cross_val_data))
```

In [43]:

```
gene_train_encoding = (gene_train_encoding.T/gene_train_encoding.sum(axis=1)).T
gene_test_encoding = (gene_test_encoding.T/gene_test_encoding.sum(axis=1)).T
gene_crossval_encoding = (gene_crossval_encoding.T/gene_crossval_encoding.sum(axis=1)).T
```

In [44]:

```
print("Size of response encoded features in train data = "+str(gene_train_encoding.shape))
print("Size of response encoded features in test data = "+str(gene_test_encoding.shape))
print("Size of response encoded features in CV data = "+str(gene_crossval_encoding.shape))
```

```
Size of response encoded features in train data = (2121, 9)
Size of response encoded features in test data = (664, 9)
Size of response encoded features in CV data = (531, 9)
```

In [45]:

```
vectorgene = CountVectorizer()
gene_train_one_hot_encoded = vectorgene.fit_transform(final_train['Gene'])
gene_test_one_hot_encoded = vectorgene.transform(X_Test['Gene'])
gene_crossval_one_hot_encoded = vectorgene.transform(cross_val_data['Gene'])
```

In [46]:

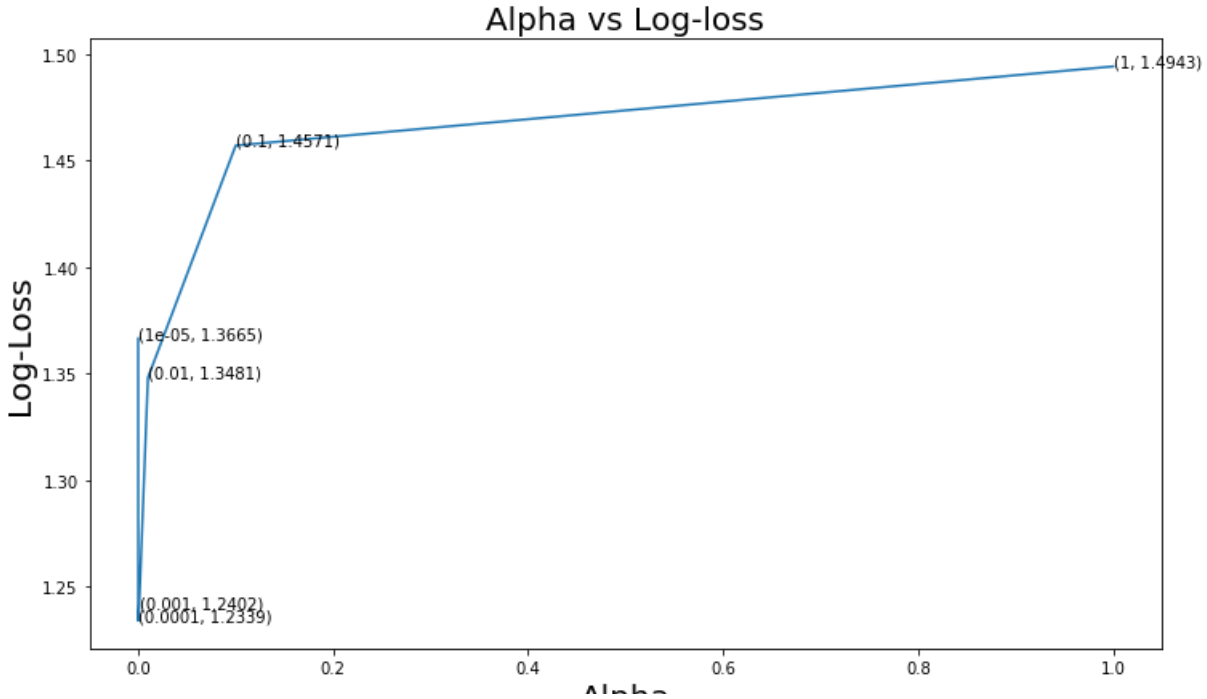
```
print("Size of one-hot encoded features in train data = "+str(gene_train_one_hot_encoded.shape))
print("Size of one-hot encoded features in test data = "+str(gene_test_one_hot_encoded.shape))
print("Size of one-hot encoded features in CV data = "+str(gene_crossval_one_hot_encoded.shape))
```

```
Size of one-hot encoded features in train data = (2121, 231)
Size of one-hot encoded features in test data = (664, 231)
Size of one-hot encoded features in CV data = (531, 231)
```

In [47]:

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

cross_val_lgloss = []
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_gene_feature_onehotCoding, cross_val_y)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    #this is a cross-validation
    classifier_generate_claibrated.fit(train_gene_feature_onehotCoding, cross_val_y)
    predicted_y = classifier_generate_claibrated.predict_proba(cv_gene_feature_onehotCoding)
    cross_val_lgloss.append(log_loss(label_cross_val_data, predicted_y, labels=clf.classes_))
    print("For alpha value of "+str(i)+" CV log loss = "+str(log_loss(label_cross_val_data, predict
ed_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```

```
For alpha value of 1e-05 CV log loss = 1.3664557036424139
For alpha value of 0.0001 CV log loss = 1.2338743693775847
For alpha value of 0.001 CV log loss = 1.240161361283013
For alpha value of 0.01 CV log loss = 1.3480508759177392
For alpha value of 0.1 CV log loss = 1.4571446712648841
For alpha value of 1 CV log loss = 1.4943304880484825
```



Alpha

In [48]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]
clf = SGDClassifier(alpha=best_alpha, penalty='l2', loss='log', random_state=42)
clf.fit(train_gene_feature_onehotCoding, cross_val_y)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(train_gene_feature_onehotCoding, cross_val_y)

train_final_pred = classifier_generate_claibrated.predict_proba(train_gene_feature_onehotCoding)
print('For values of best alpha = ', best_alpha, "the train log loss =:",log_loss(cross_val_y, tra
in_final_pred, labels=clf.classes_))

cross_val_final_pred = classifier_generate_claibrated.predict_proba(cv_gene_feature_onehotCoding)
print('For values of best alpha = ', best_alpha, "the CV log loss
=:",log_loss(label_cross_val_data, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(test_gene_feature_onehotCoding)
print('For values of best alpha = ', best_alpha, "the test log loss =:",log_loss(Y_Test,
test_final_pred, labels=clf.classes_))
```

```
For values of best alpha =  0.0001 the train log loss =: 1.0201550004769964
For values of best alpha =  0.0001 the CV log loss =: 1.2338743693775847
For values of best alpha =  0.0001 the test log loss =: 1.2192205191626133
```

In [49]:

```
print("Ques: How many common gene features are there in train, cv and test data?")
test_data_comongene = X_Test[X_Test["Gene"].isin(final_train["Gene"])].shape[0]
crossval_data_comongene = cross_val_data[cross_val_data["Gene"].isin(final_train["Gene"])].shape[0]
print("Ans:")
print("Percentage of common gene features in test and train data =
"+str(np.round((test_data_comongene/X_Test.shape[0])*100, 2))+"%")
print("Percentage of common gene features in CV and train data =
"+str(np.round((crossval_data_comongene/cross_val_data.shape[0])*100, 2))+"%")
```

```
Ques: How many common gene features are there in train, cv and test data?
Ans:
Percentage of common gene features in test and train data = 96.84%
Percentage of common gene features in CV and train data = 96.61%
```

In [50]:

```
gen_uniq_data = final_train["Variation"].value_counts()
print("Number of Unique variations = "+str(gen_uniq_data.shape[0]))
print(gen_uniq_data.head(10))
```

```
Number of Unique variations = 1926
Truncating_Mutations    57
Amplification           49
Deletion                40
Fusions                 25
G12V                     3
Q61H                     3
Overexpression           3
E17K                     3
V321M                    2
P130S                    2
Name: Variation, dtype: int64
```

In [51]:

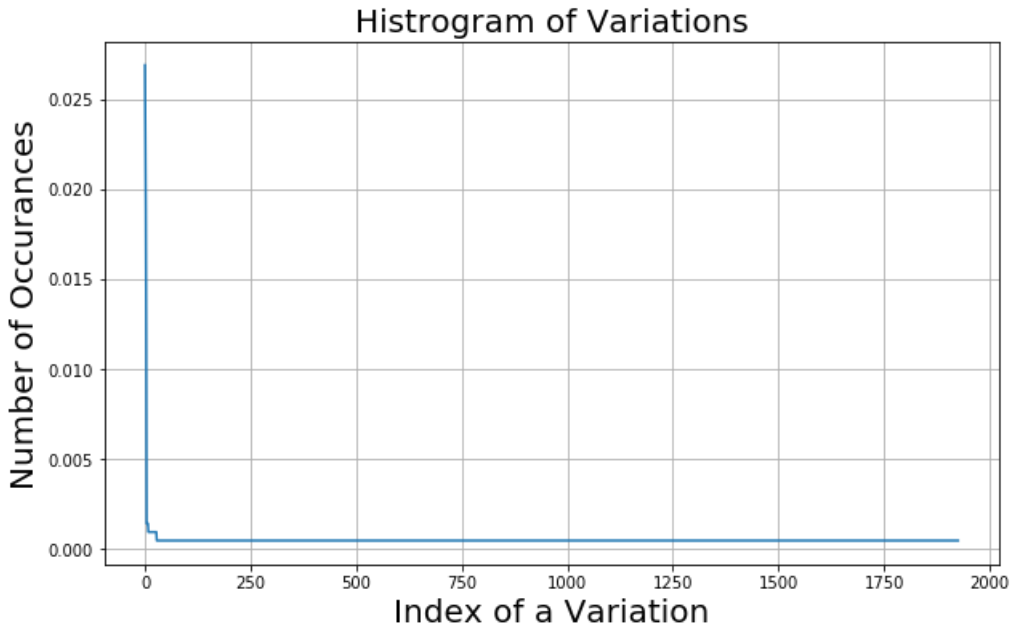
```
print("Ans: There are", gen_uniq_data.shape[0] ,"different categories of variations in the train da
ta, and they are distibuted as follows:")
```

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

In [52]:

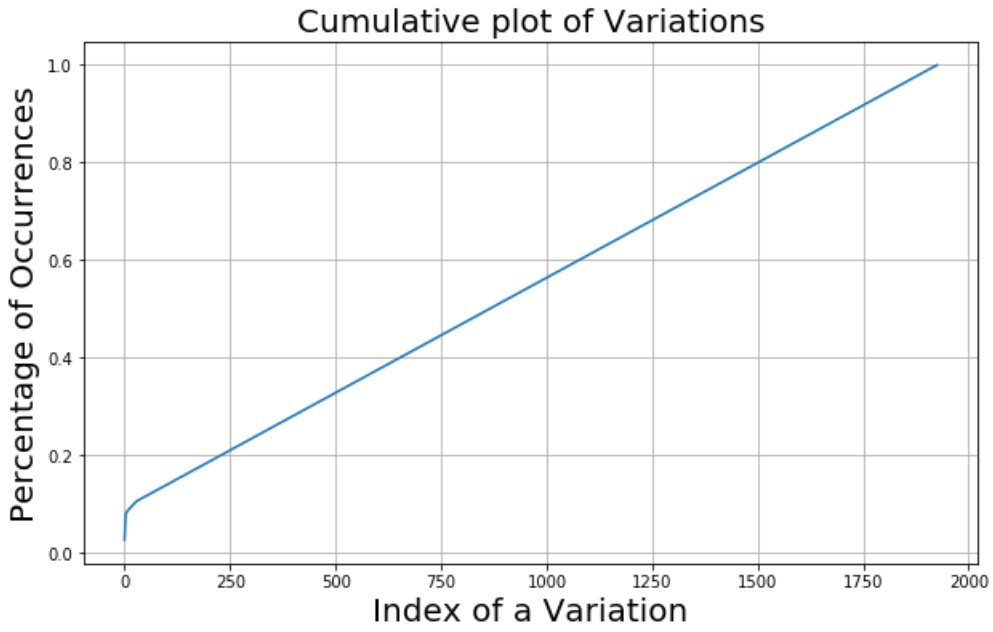
In [52]:

```
sum_gene = sum(gen_uniq_data.values)
dist_gene = gen_uniq_data.values/sum_gene
plt.figure(figsize = (10, 6))
plt.plot(dist_gene)
plt.title("Histrogram of Variations", fontsize = 20)
plt.xlabel('Index of a Variation', fontsize = 20)
plt.ylabel('Number of Occurances', fontsize = 20)
plt.grid()
plt.show()
```



In [53]:

```
sum_gene = sum(gen_uniq_data.values)
dist_gene = gen_uniq_data.values/sum_gene
c = np.cumsum(dist_gene)
plt.figure(figsize = (10, 6))
plt.plot(c)
plt.title("Cumulative plot of Variations", fontsize = 20)
plt.xlabel('Index of a Variation', fontsize = 20)
plt.ylabel('Percentage of Occurrences', fontsize = 20)
plt.grid()
plt.show()
```



In [54]:

In [54]:

```
alpha = 1
train_variation_feature_responseCoding = np.array(res_val(alpha, "Variation", final_train))
test_variation_feature_responseCoding = np.array(res_val(alpha, "Variation", X_Test))
cv_variation_feature_responseCoding = np.array(res_val(alpha, "Variation", cross_val_data))
```

In [55]:

```
train_variation_feature_responseCoding =
(train_variation_feature_responseCoding.T/train_variation_feature_responseCoding.sum(axis=1)).T
test_variation_feature_responseCoding =
(test_variation_feature_responseCoding.T/test_variation_feature_responseCoding.sum(axis=1)).T
cv_variation_feature_responseCoding =
(cv_variation_feature_responseCoding.T/cv_variation_feature_responseCoding.sum(axis=1)).T
```

In [56]:

```
print("Size of response encoded features in train data =
"+str(train_variation_feature_responseCoding.shape))
print("Size of response encoded features in test data =
"+str(test_variation_feature_responseCoding.shape))
print("Size of response encoded features in CV data = "+str(cv_variation_feature_responseCoding.sh
ape))
```

```
Size of response encoded features in train data = (2121, 9)
Size of response encoded features in test data = (664, 9)
Size of response encoded features in CV data = (531, 9)
```

In [57]:

```
variationVectorizer = CountVectorizer()
train_variation_feature_onehotCoding = variationVectorizer.fit_transform(final_train['Variation'])
test_variation_feature_onehotCoding = variationVectorizer.transform(X_Test['Variation'])
cv_variation_feature_onehotCoding = variationVectorizer.transform(cross_val_data['Variation'])
```

In [58]:

```
print("Size of one-hot encoded features in train data = "+str(train_variation_feature_onehotCoding
.shape))
print("Size of one-hot encoded features in test data = "+str(test_variation_feature_onehotCoding.s
hape))
print("Size of one-hot encoded features in CV data = "+str(cv_variation_feature_onehotCoding.shape
))
```

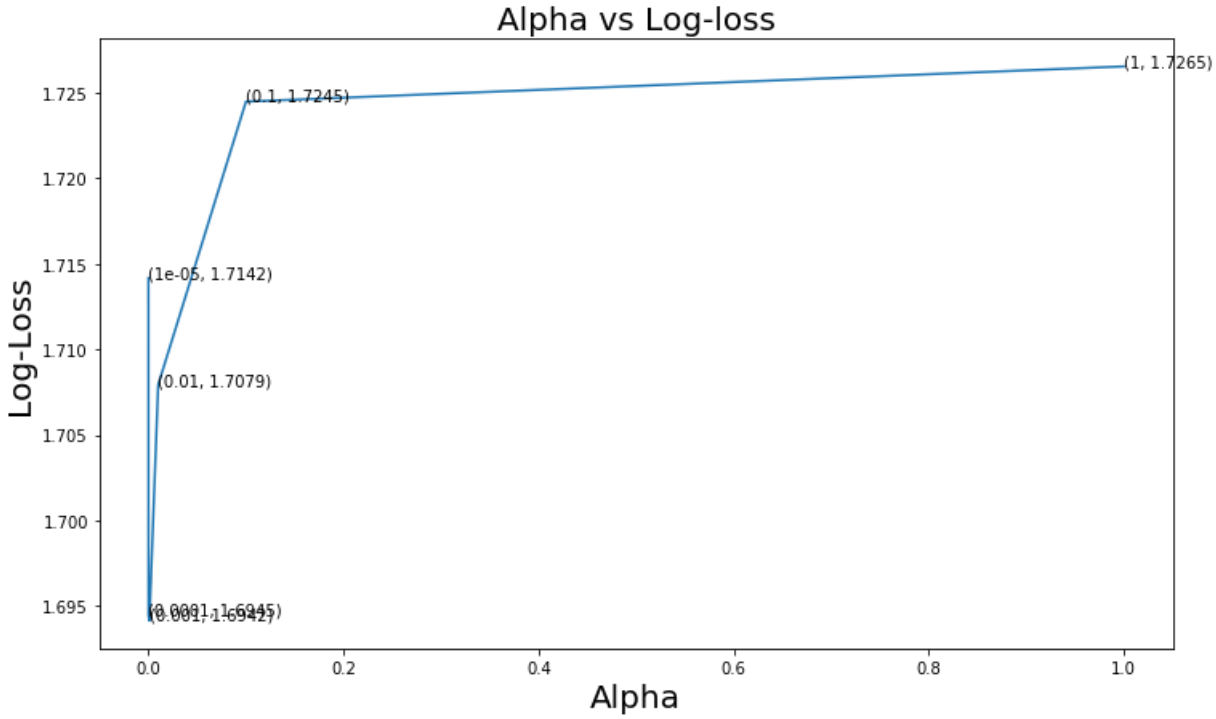
```
Size of one-hot encoded features in train data = (2121, 1960)
Size of one-hot encoded features in test data = (664, 1960)
Size of one-hot encoded features in CV data = (531, 1960)
```

In [59]:

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

cross_val_lgloss = []
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_variation_feature_onehotCoding, cross_val_y)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    classifier_generate_claibrated.fit(train_variation_feature_onehotCoding, cross_val_y)
    predicted_y = classifier_generate_claibrated.predict_proba(cv_variation_feature_onehotCoding)
    cross_val_lgloss.append(log_loss(label_cross_val_data, predicted_y, labels=clf.classes_))
    print("For alpha value of "+str(i)+" CV log loss = "+str(log_loss(label_cross_val_data, predict
ed_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
```

```
plt.show()
```

```
For alpha value of 1e-05 CV log loss = 1.7141515246311514
For alpha value of 0.0001 CV log loss = 1.6944516886030232
For alpha value of 0.001 CV log loss = 1.6941554287218221
For alpha value of 0.01 CV log loss = 1.7079272121104785
For alpha value of 0.1 CV log loss = 1.7244541941635538
For alpha value of 1 CV log loss = 1.7265074461494674
```



In [60]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]
clf = SGDClassifier(alpha=best_alpha, penalty='l2', loss='log', random_state=42)
clf.fit(train_variation_feature_onehotCoding, cross_val_y)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(train_variation_feature_onehotCoding, cross_val_y)

train_final_pred =
classifier_generate_claibrated.predict_proba(train_variation_feature_onehotCoding)
print('For values of best alpha = ', best_alpha, "the train log loss =:",log_loss(cross_val_y, tra
in_final_pred, labels=clf.classes_))

cross_val_final_pred =
classifier_generate_claibrated.predict_proba(cv_variation_feature_onehotCoding)
print('For values of best alpha = ', best_alpha, "the CV log loss
=:",log_loss(label_cross_val_data, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(test_variation_feature_onehotCoding
)
print('For values of best alpha = ', best_alpha, "the test log loss =:",log_loss(Y_Test,
test_final_pred, labels=clf.classes_))
```

```
For values of best alpha =  0.001 the train log loss =: 1.052420759553998
For values of best alpha =  0.001 the CV log loss =: 1.6941554287218221
For values of best alpha =  0.001 the test log loss =: 1.7246954865100943
```

In [61]:

```
print("Ques: How many common variation features are there in train, cv and test data?")
test_gene_comon_check = X_Test[X_Test["Variation"].isin(TrainData["Variation"])].shape[0]
cross_val_gene_comon_check = cross_val_data[cross_val_data["Variation"].isin(TrainData["Variation"]
)].shape[0]
print("Ans:")
print("Percentage of common Variation features in test and train data =
"+str(np.round((test_gene_comon_check/X_Test.shape[0])*100, 2))+"%")
print("Percentage of common Variation features in CV and train data =
```

```
print("Percentage of common variation features in CV and train data =
"+str(np.round((cross_val_gene_comon_check/cross_val_data.shape[0])*100, 2))+"%")
```

```
Ques: How many common variation features are there in train, cv and test data?
Ans:
Percentage of common Variation features in test and train data = 10.39%
Percentage of common Variation features in CV and train data = 10.55%
```

Unique words of Train

In [62]:

```
dict_new_train = defaultdict(int)
for indx, txt in final_train.iterrows():
    for word in txt['Text'].split():
        dict_new_train[word] += 1
print("Number of unique words in train data = "+str(len(dict_new_train.keys())))
```

```
Number of unique words in train data = 126077
```

Response Coding

In [63]:

```
def word_occur(cls_text):
    dict_new_train = defaultdict(int)
    for indx, txt in cls_text.iterrows():
        for word in txt['Text'].split():
            dict_new_train[word] += 1
    return dict_new_train
```

In [64]:

```
dict_data_new1 = []
for i in range(1,10):
    cls_text = final_train[final_train['Class']==i]
    dict_data_new1.append(word_occur(cls_text))
final_dict_data = word_occur(final_train)
```

In [65]:

```
def respon_text_coding(df):
    alpha = 10
    feat_resp_code_text = np.zeros((df.shape[0], 9))
    for i in range(0,9):
        rowIndex = 0
        for idx, txt in df.iterrows():
            sumProbability = 0
            for word in txt["Text"].split():
                sumProbability +=
math.log(((dict_data_new1[i].get(word,0))+alpha)/((final_dict_data.get(word,0))+(9*alpha)))
            feat_resp_code_text[rowIndex][i] = math.exp(sumProbability/len(row["Text"].split()))
            rowIndex += 1
    return feat_resp_code_text
```

In [66]:

```
train_text_feature_responseCoding  = respon_text_coding(final_train)
test_text_feature_responseCoding  = respon_text_coding(X_Test)
cv_text_feature_responseCoding  = respon_text_coding(cross_val_data)
```

In [67]:

```
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 [68]:

```
print("Size of response coded text feature of Train Data = "+str(train_text_feature_responseCoding
.shape))
print("Size of response coded text feature of Test Data = "+str(test_text_feature_responseCoding.s
hape))
print("Size of response coded text feature of CV Data = "+str(cv_text_feature_responseCoding.shape
))
```

```
Size of response coded text feature of Train Data = (2121, 9)
Size of response coded text feature of Test Data = (664, 9)
Size of response coded text feature of CV Data = (531, 9)
```

TFIDF

In [69]:

```
tf_idf_vect = TfidfVectorizer(ngram_range = (1,2), stop_words = "english", min_df = 3, max_features
= 100000)

tf_idf_train_feat = tf_idf_vect.fit_transform(final_train['Text'])
tf_idf_train_feat = StandardScaler(with_mean = False).fit_transform(tf_idf_train_feat)

tf_idf_test_feat = tf_idf_vect.transform(X_Test['Text'])
tf_idf_test_feat = StandardScaler(with_mean = False).fit_transform(tf_idf_test_feat)

tf_idf_cross_val_feat = tf_idf_vect.transform(cross_val_data['Text'])
tf_idf_cross_val_feat = StandardScaler(with_mean = False).fit_transform(tf_idf_cross_val_feat)
```

In [70]:

```
print("Size of TFIDF coded text feature of Train Data = "+str(tf_idf_train_feat.shape))
print("Size of TFIDF coded text feature of Test Data = "+str(tf_idf_test_feat.shape))
print("Size of TFIDF coded text feature of CV Data = "+str(tf_idf_cross_val_feat.shape))
```

```
Size of TFIDF coded text feature of Train Data = (2121, 100000)
Size of TFIDF coded text feature of Test Data = (664, 100000)
Size of TFIDF coded text feature of CV Data = (531, 100000)
```

In [71]:

```
count_vec = CountVectorizer(min_df = 3)
word_occ_train = count_vec.fit_transform(final_train['Text'])
TrainFeatures= count_vec.get_feature_names()
final_dict_occrword = dict(zip(TrainFeatures,word_occ_train.sum(axis=0).A1))
```

In [72]:

```
new_dictsort_word = dict(sorted(final_dict_occrword.items(), key=lambda x: x[1] , reverse=True))
```

In [73]:

```
new_dictsort_word_list = np.array(list(new_dictsort_word.values()))
```

In [74]:

```
NoOfWord_Occurrence = Counter(new_dictsort_word_list)
```

In [75]:

```
print(NoOfWord_Occurrence)
```

```
Counter({3: 4768, 4: 3476, 5: 2804, 6: 2549, 7: 2038, 8: 1908, 10: 1754, 12: 1541, 9: 1499, 14: 130
5, 16: 973, 11: 900, 13: 794, 15: 767, 18: 642, 20: 640, 17: 575, 21: 563, 19: 525, 22: 497, 24: 4
84, 30: 414, 23: 399, 25: 398, 28: 366, 37: 348, 27: 340, 26: 310, 33: 308, 32: 303, 36: 299, 31:
```

272, 47: 269, 29: 264, 35: 262, 40: 240, 42: 237, 34: 223, 48: 206, 38: 204, 43: 200, 45: 185, 39:
185, 44: 182, 41: 170, 46: 164, 50: 159, 56: 155, 51: 153, 60: 150, 57: 140, 53: 139, 49: 137, 55:
132, 54: 130, 52: 124, 64: 122, 63: 117, 58: 116, 62: 113, 72: 107, 59: 107, 67: 104, 70: 101, 68:
101, 75: 100, 61: 96, 66: 92, 65: 92, 86: 90, 73: 89, 84: 88, 77: 88, 74: 87, 71: 87, 69: 84, 81:
83, 76: 80, 80: 78, 88: 75, 94: 73, 85: 73, 92: 69, 103: 67, 83: 67, 79: 65, 100: 63, 82: 63, 87:
62, 91: 61, 78: 61, 89: 60, 98: 59, 97: 58, 90: 57, 93: 55, 101: 54, 96: 54, 108: 53, 105: 53, 95
: 52, 99: 51, 116: 50, 126: 49, 120: 49, 114: 49, 141: 48, 112: 48, 111: 48, 109: 48, 118: 47, 140
: 45, 119: 45, 147: 43, 110: 43, 107: 43, 121: 42, 133: 41, 130: 41, 113: 41, 104: 41, 102: 41, 13
8: 40, 156: 39, 129: 39, 122: 38, 132: 37, 160: 36, 154: 36, 142: 36, 136: 36, 128: 36, 124: 36, 1
17: 36, 115: 36, 137: 35, 135: 35, 134: 35, 131: 35, 123: 34, 158: 33, 153: 33, 151: 33, 139: 33,
106: 33, 149: 32, 167: 31, 150: 31, 143: 31, 144: 30, 216: 29, 168: 29, 163: 29, 148: 29, 190: 28,
181: 28, 125: 28, 212: 27, 146: 27, 166: 26, 155: 26, 215: 25, 203: 25, 176: 25, 175: 25, 170: 25,
213: 24, 210: 24, 194: 24, 182: 24, 172: 24, 164: 24, 157: 24, 207: 23, 202: 23, 201: 23, 186: 23,
178: 23, 161: 23, 258: 22, 211: 22, 204: 22, 171: 22, 165: 22, 159: 22, 145: 22, 269: 21, 248: 21,
196: 21, 195: 21, 177: 21, 231: 20, 208: 20, 180: 20, 127: 20, 240: 19, 237: 19, 229: 19, 224: 19,
205: 19, 198: 19, 191: 19, 189: 19, 184: 19, 183: 19, 179: 19, 152: 19, 243: 18, 241: 18, 226: 18,
218: 18, 200: 18, 199: 18, 197: 18, 174: 18, 173: 18, 282: 17, 253: 17, 247: 17, 230: 17, 223: 17,
221: 17, 214: 17, 192: 17, 188: 17, 185: 17, 162: 17, 333: 16, 325: 16, 308: 16, 306: 16, 264: 16,
260: 16, 256: 16, 239: 16, 220: 16, 320: 15, 296: 15, 289: 15, 252: 15, 251: 15, 236: 15, 225: 15,
312: 14, 300: 14, 273: 14, 272: 14, 266: 14, 250: 14, 245: 14, 222: 14, 219: 14, 209: 14, 187: 14,
379: 13, 354: 13, 314: 13, 305: 13, 290: 13, 274: 13, 271: 13, 235: 13, 233: 13, 228: 13, 217: 13,
193: 13, 169: 13, 442: 12, 346: 12, 345: 12, 332: 12, 331: 12, 315: 12, 287: 12, 284: 12, 283: 12,
280: 12, 278: 12, 276: 12, 270: 12, 259: 12, 257: 12, 249: 12, 238: 12, 234: 12, 227: 12, 206: 12,
617: 11, 433: 11, 409: 11, 388: 11, 387: 11, 372: 11, 349: 11, 343: 11, 342: 11, 341: 11, 318: 11,
317: 11, 316: 11, 310: 11, 299: 11, 297: 11, 288: 11, 267: 11, 255: 11, 244: 11, 232: 11, 511: 10,
460: 10, 447: 10, 444: 10, 439: 10, 432: 10, 395: 10, 370: 10, 363: 10, 362: 10, 340: 10, 338: 10,
336: 10, 330: 10, 309: 10, 302: 10, 301: 10, 294: 10, 291: 10, 286: 10, 285: 10, 265: 10, 261: 10,
254: 10, 246: 10, 491: 9, 469: 9, 456: 9, 443: 9, 437: 9, 425: 9, 424: 9, 404: 9, 399: 9, 376: 9,
365: 9, 356: 9, 347: 9, 339: 9, 329: 9, 328: 9, 324: 9, 298: 9, 295: 9, 293: 9, 292: 9, 268: 9,
263: 9, 262: 9, 242: 9, 802: 8, 701: 8, 647: 8, 574: 8, 492: 8, 483: 8, 478: 8, 474: 8, 448: 8,
426: 8, 420: 8, 380: 8, 377: 8, 371: 8, 361: 8, 360: 8, 350: 8, 344: 8, 337: 8, 327: 8, 321: 8,
311: 8, 307: 8, 304: 8, 303: 8, 279: 8, 277: 8, 275: 8, 820: 7, 808: 7, 714: 7, 677: 7, 659: 7,
611: 7, 583: 7, 575: 7, 562: 7, 541: 7, 505: 7, 497: 7, 489: 7, 468: 7, 467: 7, 457: 7, 430: 7,
428: 7, 423: 7, 421: 7, 412: 7, 411: 7, 410: 7, 407: 7, 401: 7, 393: 7, 383: 7, 375: 7, 367: 7,
355: 7, 352: 7, 351: 7, 348: 7, 334: 7, 319: 7, 313: 7, 1049: 6, 864: 6, 814: 6, 800: 6, 724: 6,
713: 6, 710: 6, 705: 6, 673: 6, 649: 6, 643: 6, 633: 6, 616: 6, 610: 6, 603: 6, 584: 6, 579: 6,
564: 6, 558: 6, 552: 6, 547: 6, 544: 6, 532: 6, 526: 6, 524: 6, 523: 6, 518: 6, 516: 6, 509: 6,
507: 6, 504: 6, 463: 6, 462: 6, 458: 6, 453: 6, 441: 6, 438: 6, 436: 6, 427: 6, 405: 6, 402: 6,
400: 6, 398: 6, 390: 6, 386: 6, 382: 6, 378: 6, 368: 6, 364: 6, 359: 6, 358: 6, 357: 6, 353: 6,
335: 6, 1362: 5, 1228: 5, 1001: 5, 959: 5, 942: 5, 940: 5, 897: 5, 892: 5, 774: 5, 761: 5, 758: 5,
755: 5, 754: 5, 745: 5, 735: 5, 722: 5, 718: 5, 711: 5, 709: 5, 702: 5, 686: 5, 671: 5, 667: 5,
660: 5, 653: 5, 646: 5, 645: 5, 640: 5, 632: 5, 626: 5, 619: 5, 588: 5, 582: 5, 576: 5, 567: 5,
548: 5, 528: 5, 525: 5, 522: 5, 520: 5, 517: 5, 510: 5, 496: 5, 493: 5, 488: 5, 486: 5, 485: 5,
477: 5, 452: 5, 449: 5, 415: 5, 406: 5, 403: 5, 396: 5, 392: 5, 391: 5, 384: 5, 381: 5, 374: 5,
366: 5, 326: 5, 281: 5, 2503: 4, 1507: 4, 1456: 4, 1312: 4, 1283: 4, 1264: 4, 1226: 4, 1154: 4, 104
7: 4, 1041: 4, 981: 4, 974: 4, 954: 4, 919: 4, 917: 4, 912: 4, 899: 4, 889: 4, 868: 4, 866: 4,
863: 4, 860: 4, 843: 4, 840: 4, 835: 4, 822: 4, 819: 4, 798: 4, 783: 4, 781: 4, 775: 4, 770: 4,
764: 4, 692: 4, 688: 4, 685: 4, 682: 4, 679: 4, 676: 4, 675: 4, 674: 4, 668: 4, 663: 4, 650: 4,
648: 4, 644: 4, 642: 4, 638: 4, 637: 4, 636: 4, 635: 4, 631: 4, 627: 4, 624: 4, 612: 4, 601: 4,
590: 4, 586: 4, 585: 4, 581: 4, 570: 4, 569: 4, 561: 4, 553: 4, 550: 4, 549: 4, 545: 4, 540: 4,
538: 4, 536: 4, 530: 4, 514: 4, 503: 4, 499: 4, 495: 4, 494: 4, 481: 4, 480: 4, 479: 4, 472: 4,
466: 4, 465: 4, 461: 4, 459: 4, 455: 4, 454: 4, 446: 4, 434: 4, 422: 4, 416: 4, 413: 4, 394: 4,
369: 4, 4026: 3, 3400: 3, 3233: 3, 2611: 3, 2549: 3, 2470: 3, 2432: 3, 2244: 3, 2154: 3, 2144: 3, 2
124: 3, 2098: 3, 2056: 3, 2051: 3, 2044: 3, 2040: 3, 2024: 3, 2003: 3, 1947: 3, 1945: 3, 1926: 3, 1
908: 3, 1831: 3, 1824: 3, 1818: 3, 1779: 3, 1741: 3, 1717: 3, 1653: 3, 1623: 3, 1613: 3, 1600: 3, 1
595: 3, 1579: 3, 1553: 3, 1515: 3, 1487: 3, 1482: 3, 1458: 3, 1453: 3, 1430: 3, 1427: 3, 1414: 3, 1
406: 3, 1392: 3, 1380: 3, 1373: 3, 1370: 3, 1363: 3, 1359: 3, 1356: 3, 1324: 3, 1316: 3, 1301: 3, 1
295: 3, 1273: 3, 1259: 3, 1258: 3, 1237: 3, 1234: 3, 1232: 3, 1231: 3, 1230: 3, 1216: 3, 1211: 3, 1
205: 3, 1202: 3, 1198: 3, 1192: 3, 1185: 3, 1168: 3, 1162: 3, 1145: 3, 1142: 3, 1139: 3, 1132: 3, 1
129: 3, 1127: 3, 1125: 3, 1124: 3, 1123: 3, 1118: 3, 1116: 3, 1113: 3, 1111: 3, 1100: 3, 1083: 3, 1
080: 3, 1077: 3, 1076: 3, 1075: 3, 1058: 3, 1028: 3, 1026: 3, 1024: 3, 1014: 3, 1013: 3, 1009: 3, 1
000: 3, 996: 3, 972: 3, 965: 3, 961: 3, 960: 3, 956: 3, 953: 3, 941: 3, 931: 3, 929: 3, 927: 3,
926: 3, 920: 3, 913: 3, 910: 3, 907: 3, 900: 3, 896: 3, 894: 3, 891: 3, 885: 3, 881: 3, 879: 3,
871: 3, 867: 3, 859: 3, 857: 3, 842: 3, 833: 3, 832: 3, 812: 3, 811: 3, 810: 3, 807: 3, 804: 3,
794: 3, 788: 3, 786: 3, 784: 3, 773: 3, 769: 3, 768: 3, 766: 3, 765: 3, 753: 3, 752: 3, 742: 3,
739: 3, 734: 3, 729: 3, 727: 3, 726: 3, 723: 3, 721: 3, 719: 3, 717: 3, 699: 3, 694: 3, 689: 3,
687: 3, 683: 3, 670: 3, 669: 3, 664: 3, 661: 3, 641: 3, 639: 3, 629: 3, 621: 3, 613: 3, 602: 3,
597: 3, 595: 3, 593: 3, 592: 3, 591: 3, 589: 3, 587: 3, 571: 3, 568: 3, 566: 3, 559: 3, 557: 3,
556: 3, 554: 3, 546: 3, 543: 3, 539: 3, 535: 3, 533: 3, 519: 3, 515: 3, 512: 3, 508: 3, 502: 3,
501: 3, 487: 3, 484: 3, 476: 3, 475: 3, 473: 3, 470: 3, 464: 3, 451: 3, 450: 3, 445: 3, 440: 3,
435: 3, 418: 3, 417: 3, 414: 3, 408: 3, 397: 3, 322: 3, 33437: 2, 12808: 2, 10079: 2, 7776: 2,
7107: 2, 6821: 2, 6006: 2, 5659: 2, 5167: 2, 5076: 2, 5049: 2, 4864: 2, 4804: 2, 4739: 2, 4397: 2,
4292: 2, 4161: 2, 4129: 2, 3990: 2, 3967: 2, 3927: 2, 3891: 2, 3876: 2, 3666: 2, 3609: 2, 3569: 2,
3565: 2, 3517: 2, 3450: 2, 3422: 2, 3403: 2, 3389: 2, 3374: 2, 3340: 2, 3328: 2, 3305: 2, 3278: 2,
3251: 2, 3206: 2, 3189: 2, 3145: 2, 3114: 2, 3031: 2, 2930: 2, 2927: 2, 2918: 2, 2915: 2, 2908: 2,
2897: 2, 2882: 2, 2803: 2, 2642: 2, 2637: 2, 2636: 2, 2610: 2, 2607: 2, 2587: 2, 2564: 2, 2537: 2,
2531: 2, 2493: 2, 2469: 2, 2459: 2, 2411: 2, 2408: 2, 2392: 2, 2389: 2, 2377: 2, 2372: 2, 2368: 2,

2357: 2, 2356: 2, 2355: 2, 2320: 2, 2315: 2, 2293: 2, 2281: 2, 2278: 2, 2251: 2, 2249: 2, 2229: 2,
2218: 2, 2212: 2, 2208: 2, 2195: 2, 2159: 2, 2113: 2, 2088: 2, 2072: 2, 2053: 2, 2050: 2, 2047: 2,
2016: 2, 1995: 2, 1984: 2, 1982: 2, 1969: 2, 1959: 2, 1939: 2, 1924: 2, 1920: 2, 1883: 2, 1876: 2,
1875: 2, 1874: 2, 1865: 2, 1863: 2, 1859: 2, 1854: 2, 1847: 2, 1841: 2, 1840: 2, 1839: 2, 1833: 2,
1822: 2, 1814: 2, 1811: 2, 1809: 2, 1805: 2, 1803: 2, 1798: 2, 1794: 2, 1785: 2, 1769: 2, 1766: 2,
1757: 2, 1755: 2, 1753: 2, 1747: 2, 1743: 2, 1731: 2, 1709: 2, 1708: 2, 1706: 2, 1703: 2, 1702: 2,
1699: 2, 1678: 2, 1671: 2, 1670: 2, 1665: 2, 1661: 2, 1646: 2, 1636: 2, 1635: 2, 1620: 2, 1616: 2,
1615: 2, 1597: 2, 1596: 2, 1594: 2, 1591: 2, 1590: 2, 1588: 2, 1582: 2, 1573: 2, 1569: 2, 1568: 2,
1545: 2, 1542: 2, 1536: 2, 1510: 2, 1498: 2, 1484: 2, 1479: 2, 1477: 2, 1471: 2, 1465: 2, 1461: 2,
1449: 2, 1446: 2, 1445: 2, 1443: 2, 1439: 2, 1435: 2, 1428: 2, 1411: 2, 1409: 2, 1408: 2, 1399: 2,
1398: 2, 1389: 2, 1384: 2, 1375: 2, 1372: 2, 1364: 2, 1354: 2, 1352: 2, 1350: 2, 1347: 2, 1345: 2,
1336: 2, 1335: 2, 1334: 2, 1332: 2, 1331: 2, 1317: 2, 1310: 2, 1306: 2, 1303: 2, 1302: 2, 1300: 2,
1297: 2, 1292: 2, 1291: 2, 1289: 2, 1287: 2, 1285: 2, 1284: 2, 1281: 2, 1280: 2, 1274: 2, 1268: 2,
1265: 2, 1255: 2, 1253: 2, 1252: 2, 1249: 2, 1248: 2, 1245: 2, 1239: 2, 1238: 2, 1225: 2, 1224: 2,
1207: 2, 1204: 2, 1203: 2, 1196: 2, 1194: 2, 1190: 2, 1180: 2, 1178: 2, 1166: 2, 1165: 2, 1164: 2,
1159: 2, 1157: 2, 1156: 2, 1153: 2, 1152: 2, 1151: 2, 1147: 2, 1146: 2, 1144: 2, 1141: 2, 1136: 2,
1135: 2, 1134: 2, 1130: 2, 1122: 2, 1119: 2, 1117: 2, 1112: 2, 1109: 2, 1107: 2, 1094: 2, 1089: 2,
1086: 2, 1082: 2, 1079: 2, 1078: 2, 1073: 2, 1072: 2, 1068: 2, 1067: 2, 1063: 2, 1061: 2, 1048: 2,
1044: 2, 1043: 2, 1040: 2, 1035: 2, 1033: 2, 1031: 2, 1017: 2, 1016: 2, 1010: 2, 1006: 2, 1004: 2,
997: 2, 991: 2, 984: 2, 982: 2, 980: 2, 977: 2, 976: 2, 969: 2, 962: 2, 958: 2, 957: 2, 952: 2,
950: 2, 948: 2, 945: 2, 932: 2, 930: 2, 924: 2, 922: 2, 915: 2, 903: 2, 902: 2, 893: 2, 883: 2,
882: 2, 876: 2, 872: 2, 870: 2, 869: 2, 856: 2, 855: 2, 851: 2, 850: 2, 849: 2, 847: 2, 839: 2,
838: 2, 827: 2, 826: 2, 825: 2, 823: 2, 821: 2, 817: 2, 806: 2, 803: 2, 801: 2, 799: 2, 796: 2,
795: 2, 791: 2, 785: 2, 780: 2, 778: 2, 777: 2, 772: 2, 771: 2, 767: 2, 763: 2, 762: 2, 756: 2,
747: 2, 746: 2, 744: 2, 741: 2, 738: 2, 737: 2, 733: 2, 725: 2, 716: 2, 708: 2, 706: 2, 703: 2,
696: 2, 695: 2, 693: 2, 684: 2, 681: 2, 680: 2, 678: 2, 672: 2, 662: 2, 658: 2, 656: 2, 655: 2,
654: 2, 652: 2, 634: 2, 630: 2, 628: 2, 625: 2, 623: 2, 618: 2, 615: 2, 614: 2, 609: 2, 607: 2,
605: 2, 604: 2, 600: 2, 598: 2, 596: 2, 594: 2, 580: 2, 578: 2, 572: 2, 563: 2, 560: 2, 551: 2,
542: 2, 537: 2, 534: 2, 513: 2, 506: 2, 498: 2, 490: 2, 482: 2, 471: 2, 431: 2, 429: 2, 419: 2,
389: 2, 373: 2, 323: 2, 967866: 1, 797531: 1, 594967: 1, 586947: 1, 362677: 1, 249100: 1, 185953:
1, 175104: 1, 165876: 1, 151630: 1, 137442: 1, 135667: 1, 135143: 1, 125452: 1, 117727: 1, 106225:
1, 93722: 1, 90232: 1, 84478: 1, 82397: 1, 79687: 1, 75857: 1, 73314: 1, 72147: 1, 67517: 1, 66289
: 1, 66278: 1, 65288: 1, 65266: 1, 64969: 1, 63412: 1, 62402: 1, 58060: 1, 54237: 1, 53819: 1, 518
60: 1, 50631: 1, 49731: 1, 48755: 1, 47222: 1, 46691: 1, 44675: 1, 42263: 1, 42180: 1, 41395: 1, 4
1390: 1, 41250: 1, 41104: 1, 40313: 1, 40031: 1, 39889: 1, 38336: 1, 37754: 1, 37739: 1, 36465: 1,
36158: 1, 35499: 1, 35266: 1, 34210: 1, 34125: 1, 32743: 1, 32109: 1, 31396: 1, 31019: 1, 30058: 1
, 30033: 1, 29465: 1, 29208: 1, 27967: 1, 27827: 1, 27632: 1, 27210: 1, 27175: 1, 26726: 1, 26073:
1, 26009: 1, 25970: 1, 25764: 1, 24998: 1, 24679: 1, 24632: 1, 24537: 1, 24022: 1, 23966: 1, 23740
: 1, 23404: 1, 23212: 1, 22611: 1, 22521: 1, 22170: 1, 22153: 1, 22110: 1, 22050: 1, 21722: 1, 216
41: 1, 21614: 1, 21371: 1, 21363: 1, 21131: 1, 20980: 1, 20513: 1, 20305: 1, 20228: 1, 20224: 1, 1
9947: 1, 19718: 1, 19708: 1, 19388: 1, 19342: 1, 19276: 1, 19201: 1, 19042: 1, 18816: 1, 18678: 1,
18640: 1, 18637: 1, 18596: 1, 18574: 1, 18513: 1, 18202: 1, 18130: 1, 18114: 1, 18047: 1, 17996: 1
, 17854: 1, 17709: 1, 17667: 1, 17601: 1, 17524: 1, 17499: 1, 17291: 1, 17250: 1, 17212: 1, 17171:
1, 17033: 1, 16971: 1, 16917: 1, 16819: 1, 16796: 1, 16750: 1, 16430: 1, 16399: 1, 16385: 1, 15905
: 1, 15830: 1, 15755: 1, 15715: 1, 15661: 1, 15572: 1, 15509: 1, 15296: 1, 15280: 1, 15262: 1, 152
21: 1, 15176: 1, 14988: 1, 14968: 1, 14839: 1, 14706: 1, 14594: 1, 14484: 1, 14375: 1, 14371: 1, 1
4298: 1, 14265: 1, 14148: 1, 13820: 1, 13810: 1, 13684: 1, 13673: 1, 13666: 1, 13580: 1, 13541: 1,
13361: 1, 13356: 1, 13132: 1, 13113: 1, 13082: 1, 13065: 1, 12963: 1, 12928: 1, 12853: 1, 12744: 1
, 12734: 1, 12720: 1, 12699: 1, 12622: 1, 12616: 1, 12579: 1, 12556: 1, 12515: 1, 12478: 1, 12414:
1, 12406: 1, 12370: 1, 12326: 1, 12260: 1, 12257: 1, 12201: 1, 12193: 1, 12142: 1, 12119: 1, 12118
: 1, 12043: 1, 12026: 1, 12019: 1, 12014: 1, 11976: 1, 11967: 1, 11939: 1, 11920: 1, 11869: 1, 118
66: 1, 11853: 1, 11759: 1, 11723: 1, 11715: 1, 11663: 1, 11651: 1, 11626: 1, 11554: 1, 11501: 1, 1
1443: 1, 11418: 1, 11356: 1, 11340: 1, 11284: 1, 11283: 1, 11199: 1, 11125: 1, 11104: 1, 11021: 1,
10975: 1, 10974: 1, 10901: 1, 10825: 1, 10786: 1, 10765: 1, 10695: 1, 10588: 1, 10581: 1, 10552: 1
, 10506: 1, 10461: 1, 10452: 1, 10451: 1, 10425: 1, 10385: 1, 10298: 1, 10254: 1, 10217: 1, 10206:
1, 10100: 1, 10093: 1, 10086: 1, 10081: 1, 10037: 1, 9955: 1, 9933: 1, 9894: 1, 9859: 1, 9836: 1,
9770: 1, 9753: 1, 9748: 1, 9671: 1, 9649: 1, 9609: 1, 9594: 1, 9525: 1, 9510: 1, 9429: 1, 9411: 1,
9364: 1, 9344: 1, 9342: 1, 9318: 1, 9317: 1, 9298: 1, 9258: 1, 9197: 1, 9184: 1, 9167: 1, 9137: 1,
9104: 1, 9091: 1, 9082: 1, 9045: 1, 8995: 1, 8957: 1, 8948: 1, 8944: 1, 8867: 1, 8865: 1, 8847: 1,
8833: 1, 8678: 1, 8644: 1, 8635: 1, 8623: 1, 8549: 1, 8456: 1, 8438: 1, 8407: 1, 8392: 1, 8391: 1,
8344: 1, 8317: 1, 8316: 1, 8298: 1, 8213: 1, 8207: 1, 8191: 1, 8189: 1, 8178: 1, 8158: 1, 8151: 1,
8148: 1, 8142: 1, 8140: 1, 8138: 1, 8137: 1, 8118: 1, 8097: 1, 8047: 1, 8027: 1, 7938: 1, 7935: 1,
7902: 1, 7883: 1, 7878: 1, 7855: 1, 7837: 1, 7820: 1, 7814: 1, 7766: 1, 7752: 1, 7726: 1, 7709: 1,
7698: 1, 7695: 1, 7689: 1, 7644: 1, 7629: 1, 7612: 1, 7584: 1, 7568: 1, 7525: 1, 7520: 1, 7481: 1,
7479: 1, 7463: 1, 7446: 1, 7433: 1, 7416: 1, 7409: 1, 7377: 1, 7371: 1, 7343: 1, 7341: 1, 7315: 1,
7305: 1, 7299: 1, 7293: 1, 7265: 1, 7242: 1, 7229: 1, 7216: 1, 7210: 1, 7181: 1, 7162: 1, 7138: 1,
7129: 1, 7110: 1, 7076: 1, 7064: 1, 7058: 1, 7057: 1, 7005: 1, 6995: 1, 6984: 1, 6962: 1, 6953: 1,
6945: 1, 6944: 1, 6935: 1, 6916: 1, 6907: 1, 6902: 1, 6900: 1, 6871: 1, 6824: 1, 6804: 1, 6793: 1,
6788: 1, 6684: 1, 6679: 1, 6657: 1, 6655: 1, 6654: 1, 6650: 1, 6644: 1, 6627: 1, 6621: 1, 6620: 1,
6613: 1, 6601: 1, 6576: 1, 6564: 1, 6557: 1, 6550: 1, 6506: 1, 6497: 1, 6481: 1, 6474: 1, 6463: 1,
6460: 1, 6439: 1, 6435: 1, 6433: 1, 6416: 1, 6412: 1, 6379: 1, 6358: 1, 6343: 1, 6338: 1, 6321: 1,
6312: 1, 6300: 1, 6291: 1, 6286: 1, 6283: 1, 6281: 1, 6277: 1, 6252: 1, 6247: 1, 6246: 1, 6237: 1,
6186: 1, 6164: 1, 6153: 1, 6150: 1, 6137: 1, 6130: 1, 6112: 1, 6107: 1, 6100: 1, 6078: 1, 6075: 1,
6057: 1, 6037: 1, 6031: 1, 6028: 1, 6022: 1, 6017: 1, 5995: 1, 5982: 1, 5977: 1, 5976: 1, 5972: 1,
5969: 1, 5964: 1, 5934: 1, 5921: 1, 5911: 1, 5909: 1, 5904: 1, 5866: 1, 5855: 1, 5850: 1, 5849: 1,
5826: 1, 5808: 1, 5795: 1, 5750: 1, 5744: 1, 5717: 1, 5711: 1, 5709: 1, 5683: 1, 5668: 1, 5652: 1,
5639: 1, 5627: 1, 5563: 1, 5562: 1, 5537: 1, 5534: 1, 5533: 1, 5513: 1, 5503: 1, 5501: 1, 5495: 1,

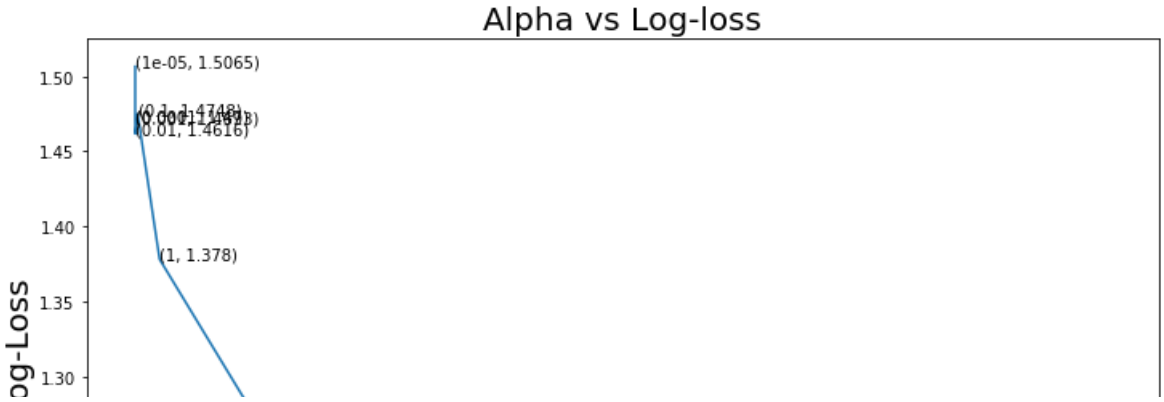
5484: 1, 5480: 1, 5467: 1, 5417: 1, 5405: 1, 5397: 1, 5393: 1, 5370: 1, 5365: 1, 5360: 1, 5355: 1,
5347: 1, 5333: 1, 5328: 1, 5319: 1, 5317: 1, 5306: 1, 5293: 1, 5277: 1, 5276: 1, 5268: 1, 5267: 1,
5246: 1, 5231: 1, 5210: 1, 5178: 1, 5170: 1, 5162: 1, 5156: 1, 5155: 1, 5142: 1, 5139: 1, 5137: 1,
5113: 1, 5105: 1, 5092: 1, 5087: 1, 5083: 1, 5054: 1, 5038: 1, 5029: 1, 4998: 1, 4997: 1, 4987: 1,
4982: 1, 4977: 1, 4971: 1, 4970: 1, 4957: 1, 4954: 1, 4946: 1, 4930: 1, 4912: 1, 4910: 1, 4902: 1,
4891: 1, 4890: 1, 4868: 1, 4857: 1, 4847: 1, 4846: 1, 4830: 1, 4828: 1, 4825: 1, 4819: 1, 4818: 1,
4816: 1, 4812: 1, 4797: 1, 4790: 1, 4775: 1, 4772: 1, 4768: 1, 4766: 1, 4765: 1, 4763: 1, 4734: 1,
4711: 1, 4710: 1, 4687: 1, 4685: 1, 4684: 1, 4678: 1, 4675: 1, 4659: 1, 4640: 1, 4639: 1, 4633: 1,
4630: 1, 4629: 1, 4626: 1, 4605: 1, 4603: 1, 4586: 1, 4582: 1, 4577: 1, 4554: 1, 4553: 1, 4552: 1,
4544: 1, 4542: 1, 4540: 1, 4533: 1, 4528: 1, 4511: 1, 4510: 1, 4508: 1, 4500: 1, 4498: 1, 4487: 1,
4486: 1, 4483: 1, 4414: 1, 4406: 1, 4396: 1, 4392: 1, 4378: 1, 4373: 1, 4355: 1, 4354: 1, 4351: 1,
4335: 1, 4329: 1, 4321: 1, 4318: 1, 4291: 1, 4287: 1, 4282: 1, 4277: 1, 4268: 1, 4263: 1, 4262: 1,
4254: 1, 4250: 1, 4244: 1, 4242: 1, 4230: 1, 4223: 1, 4221: 1, 4216: 1, 4209: 1, 4207: 1, 4202: 1,
4196: 1, 4190: 1, 4188: 1, 4187: 1, 4182: 1, 4169: 1, 4158: 1, 4150: 1, 4137: 1, 4136: 1, 4133: 1,
4131: 1, 4130: 1, 4116: 1, 4112: 1, 4111: 1, 4108: 1, 4101: 1, 4100: 1, 4092: 1, 4085: 1, 4083: 1,
4080: 1, 4077: 1, 4075: 1, 4074: 1, 4066: 1, 4062: 1, 4054: 1, 4043: 1, 4042: 1, 4038: 1, 4037: 1,
4023: 1, 4015: 1, 4006: 1, 4000: 1, 3997: 1, 3996: 1, 3995: 1, 3987: 1, 3970: 1, 3957: 1, 3955: 1,
3947: 1, 3942: 1, 3941: 1, 3931: 1, 3924: 1, 3902: 1, 3893: 1, 3890: 1, 3885: 1, 3880: 1, 3874: 1,
3872: 1, 3870: 1, 3869: 1, 3852: 1, 3842: 1, 3841: 1, 3817: 1, 3810: 1, 3809: 1, 3808: 1, 3801: 1,
3800: 1, 3783: 1, 3782: 1, 3779: 1, 3771: 1, 3769: 1, 3751: 1, 3737: 1, 3736: 1, 3733: 1, 3728: 1,
3723: 1, 3722: 1, 3714: 1, 3706: 1, 3700: 1, 3699: 1, 3694: 1, 3691: 1, 3683: 1, 3681: 1, 3679: 1,
3673: 1, 3672: 1, 3664: 1, 3661: 1, 3652: 1, 3651: 1, 3650: 1, 3642: 1, 3639: 1, 3638: 1, 3637: 1,
3633: 1, 3631: 1, 3619: 1, 3613: 1, 3598: 1, 3597: 1, 3587: 1, 3582: 1, 3579: 1, 3575: 1, 3572: 1,
3568: 1, 3564: 1, 3556: 1, 3555: 1, 3537: 1, 3529: 1, 3527: 1, 3525: 1, 3524: 1, 3521: 1, 3515: 1,
3509: 1, 3506: 1, 3501: 1, 3483: 1, 3470: 1, 3464: 1, 3460: 1, 3459: 1, 3457: 1, 3455: 1, 3454: 1,
3449: 1, 3447: 1, 3446: 1, 3445: 1, 3440: 1, 3439: 1, 3435: 1, 3431: 1, 3424: 1, 3416: 1, 3413: 1,
3409: 1, 3391: 1, 3375: 1, 3371: 1, 3368: 1, 3366: 1, 3363: 1, 3361: 1, 3350: 1, 3349: 1, 3344: 1,
3331: 1, 3318: 1, 3316: 1, 3315: 1, 3313: 1, 3309: 1, 3307: 1, 3306: 1, 3300: 1, 3299: 1, 3295: 1,
3290: 1, 3286: 1, 3279: 1, 3271: 1, 3270: 1, 3267: 1, 3264: 1, 3258: 1, 3250: 1, 3247: 1, 3240: 1,
3239: 1, 3226: 1, 3220: 1, 3212: 1, 3204: 1, 3199: 1, 3195: 1, 3194: 1, 3191: 1, 3184: 1, 3177: 1,
3172: 1, 3169: 1, 3165: 1, 3163: 1, 3161: 1, 3158: 1, 3153: 1, 3149: 1, 3144: 1, 3139: 1, 3138: 1,
3133: 1, 3129: 1, 3128: 1, 3121: 1, 3120: 1, 3115: 1, 3110: 1, 3107: 1, 3102: 1, 3100: 1, 3094: 1,
3093: 1, 3077: 1, 3075: 1, 3072: 1, 3067: 1, 3064: 1, 3060: 1, 3055: 1, 3054: 1, 3051: 1, 3030: 1,
3026: 1, 3021: 1, 3020: 1, 3018: 1, 3017: 1, 3010: 1, 3004: 1, 3001: 1, 2995: 1, 2991: 1, 2987: 1,
2981: 1, 2979: 1, 2976: 1, 2971: 1, 2963: 1, 2962: 1, 2961: 1, 2947: 1, 2944: 1, 2936: 1, 2913: 1,
2909: 1, 2907: 1, 2904: 1, 2903: 1, 2902: 1, 2901: 1, 2884: 1, 2868: 1, 2866: 1, 2859: 1, 2855: 1,
2853: 1, 2851: 1, 2848: 1, 2844: 1, 2843: 1, 2842: 1, 2834: 1, 2833: 1, 2827: 1, 2825: 1, 2817: 1,
2815: 1, 2811: 1, 2806: 1, 2795: 1, 2794: 1, 2788: 1, 2784: 1, 2760: 1, 2755: 1, 2750: 1, 2749: 1,
2746: 1, 2743: 1, 2724: 1, 2721: 1, 2718: 1, 2715: 1, 2707: 1, 2701: 1, 2698: 1, 2693: 1, 2678: 1,
2677: 1, 2676: 1, 2672: 1, 2671: 1, 2670: 1, 2663: 1, 2660: 1, 2657: 1, 2656: 1, 2653: 1, 2651: 1,
2644: 1, 2643: 1, 2639: 1, 2633: 1, 2630: 1, 2624: 1, 2623: 1, 2620: 1, 2612: 1, 2609: 1, 2605: 1,
2601: 1, 2600: 1, 2597: 1, 2596: 1, 2592: 1, 2591: 1, 2589: 1, 2588: 1, 2586: 1, 2569: 1, 2567: 1,
2563: 1, 2560: 1, 2559: 1, 2548: 1, 2543: 1, 2542: 1, 2541: 1, 2540: 1, 2533: 1, 2529: 1, 2527: 1,
2525: 1, 2524: 1, 2523: 1, 2520: 1, 2516: 1, 2515: 1, 2514: 1, 2513: 1, 2511: 1, 2508: 1, 2505: 1,
2504: 1, 2498: 1, 2489: 1, 2487: 1, 2479: 1, 2472: 1, 2466: 1, 2464: 1, 2458: 1, 2451: 1, 2449: 1,
2440: 1, 2436: 1, 2435: 1, 2431: 1, 2430: 1, 2428: 1, 2424: 1, 2422: 1, 2421: 1, 2420: 1, 2417: 1,
2415: 1, 2414: 1, 2406: 1, 2402: 1, 2394: 1, 2393: 1, 2383: 1, 2382: 1, 2381: 1, 2379: 1, 2370: 1,
2366: 1, 2365: 1, 2354: 1, 2353: 1, 2350: 1, 2349: 1, 2343: 1, 2341: 1, 2340: 1, 2337: 1, 2334: 1,
2323: 1, 2318: 1, 2314: 1, 2313: 1, 2307: 1, 2303: 1, 2294: 1, 2290: 1, 2288: 1, 2280: 1, 2276: 1,
2274: 1, 2273: 1, 2265: 1, 2263: 1, 2259: 1, 2257: 1, 2255: 1, 2254: 1, 2253: 1, 2250: 1, 2245: 1,
2242: 1, 2232: 1, 2224: 1, 2221: 1, 2217: 1, 2216: 1, 2209: 1, 2207: 1, 2205: 1, 2203: 1, 2201: 1,
2200: 1, 2199: 1, 2198: 1, 2191: 1, 2186: 1, 2183: 1, 2182: 1, 2181: 1, 2176: 1, 2175: 1, 2173: 1,
2167: 1, 2166: 1, 2165: 1, 2164: 1, 2157: 1, 2155: 1, 2150: 1, 2143: 1, 2141: 1, 2137: 1, 2136: 1,
2134: 1, 2131: 1, 2129: 1, 2121: 1, 2119: 1, 2117: 1, 2116: 1, 2114: 1, 2110: 1, 2109: 1, 2108: 1,
2104: 1, 2092: 1, 2090: 1, 2089: 1, 2082: 1, 2080: 1, 2079: 1, 2077: 1, 2070: 1, 2069: 1, 2066: 1,
2063: 1, 2061: 1, 2058: 1, 2057: 1, 2052: 1, 2043: 1, 2042: 1, 2039: 1, 2037: 1, 2035: 1, 2033: 1,
2032: 1, 2029: 1, 2027: 1, 2023: 1, 2021: 1, 2018: 1, 2010: 1, 2000: 1, 1999: 1, 1998: 1, 1993: 1,
1990: 1, 1989: 1, 1986: 1, 1983: 1, 1980: 1, 1979: 1, 1977: 1, 1976: 1, 1975: 1, 1974: 1, 1972: 1,
1963: 1, 1960: 1, 1956: 1, 1953: 1, 1946: 1, 1942: 1, 1938: 1, 1936: 1, 1935: 1, 1929: 1, 1923: 1,
1922: 1, 1919: 1, 1918: 1, 1912: 1, 1911: 1, 1910: 1, 1907: 1, 1906: 1, 1903: 1, 1902: 1, 1899: 1,
1894: 1, 1891: 1, 1890: 1, 1886: 1, 1882: 1, 1880: 1, 1861: 1, 1858: 1, 1857: 1, 1856: 1, 1852: 1,
1848: 1, 1844: 1, 1842: 1, 1834: 1, 1832: 1, 1828: 1, 1826: 1, 1819: 1, 1816: 1, 1815: 1, 1812: 1,
1808: 1, 1807: 1, 1802: 1, 1800: 1, 1796: 1, 1791: 1, 1790: 1, 1788: 1, 1786: 1, 1784: 1, 1783: 1,
1777: 1, 1774: 1, 1773: 1, 1771: 1, 1770: 1, 1767: 1, 1765: 1, 1762: 1, 1760: 1, 1759: 1, 1758: 1,
1752: 1, 1745: 1, 1739: 1, 1735: 1, 1733: 1, 1732: 1, 1728: 1, 1726: 1, 1725: 1, 1723: 1, 1721: 1,
1719: 1, 1716: 1, 1715: 1, 1714: 1, 1713: 1, 1710: 1, 1707: 1, 1705: 1, 1698: 1, 1697: 1, 1692: 1,
1690: 1, 1689: 1, 1687: 1, 1686: 1, 1685: 1, 1679: 1, 1677: 1, 1675: 1, 1672: 1, 1663: 1, 1662: 1,
1651: 1, 1650: 1, 1648: 1, 1645: 1, 1641: 1, 1633: 1, 1632: 1, 1631: 1, 1630: 1, 1628: 1, 1626: 1,
1625: 1, 1624: 1, 1622: 1, 1619: 1, 1614: 1, 1611: 1, 1609: 1, 1607: 1, 1606: 1, 1605: 1, 1601: 1,
1593: 1, 1587: 1, 1584: 1, 1583: 1, 1581: 1, 1580: 1, 1577: 1, 1574: 1, 1572: 1, 1570: 1, 1566: 1,
1565: 1, 1564: 1, 1561: 1, 1556: 1, 1555: 1, 1552: 1, 1551: 1, 1548: 1, 1547: 1, 1546: 1, 1544: 1,
1543: 1, 1539: 1, 1537: 1, 1531: 1, 1528: 1, 1527: 1, 1526: 1, 1522: 1, 1520: 1, 1518: 1, 1516: 1,
1513: 1, 1512: 1, 1509: 1, 1506: 1, 1504: 1, 1503: 1, 1502: 1, 1500: 1, 1497: 1, 1495: 1, 1491: 1,
1489: 1, 1485: 1, 1481: 1, 1478: 1, 1476: 1, 1473: 1, 1472: 1, 1470: 1, 1467: 1, 1466: 1, 1462: 1,
1459: 1, 1457: 1, 1454: 1, 1450: 1, 1448: 1, 1447: 1, 1444: 1, 1441: 1, 1433: 1, 1432: 1, 1429: 1,
1425: 1, 1422: 1, 1421: 1, 1418: 1, 1417: 1, 1416: 1, 1413: 1, 1412: 1, 1410: 1, 1404: 1, 1403: 1,
1401: 1, 1397: 1, 1394: 1, 1388: 1, 1386: 1, 1385: 1, 1383: 1, 1381: 1, 1379: 1, 1374: 1, 1368: 1,

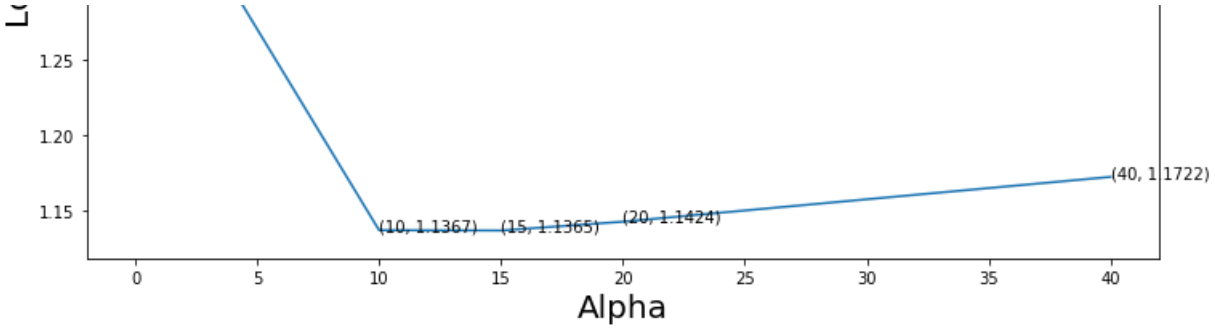
```
1366: 1, 1361: 1, 1358: 1, 1357: 1, 1355: 1, 1353: 1, 1348: 1, 1346: 1, 1344: 1, 1342: 1, 1338: 1,
1337: 1, 1333: 1, 1329: 1, 1327: 1, 1319: 1, 1318: 1, 1314: 1, 1313: 1, 1311: 1, 1308: 1, 1299: 1,
1294: 1, 1286: 1, 1282: 1, 1277: 1, 1276: 1, 1275: 1, 1272: 1, 1270: 1, 1269: 1, 1267: 1, 1266: 1,
1256: 1, 1251: 1, 1250: 1, 1247: 1, 1246: 1, 1244: 1, 1241: 1, 1240: 1, 1235: 1, 1233: 1, 1229: 1,
1227: 1, 1223: 1, 1222: 1, 1221: 1, 1220: 1, 1219: 1, 1218: 1, 1215: 1, 1214: 1, 1213: 1, 1209: 1,
1208: 1, 1199: 1, 1197: 1, 1195: 1, 1193: 1, 1188: 1, 1181: 1, 1179: 1, 1177: 1, 1175: 1, 1172: 1,
1170: 1, 1167: 1, 1163: 1, 1161: 1, 1155: 1, 1149: 1, 1143: 1, 1138: 1, 1137: 1, 1133: 1, 1131: 1,
1128: 1, 1126: 1, 1115: 1, 1114: 1, 1110: 1, 1108: 1, 1105: 1, 1103: 1, 1101: 1, 1097: 1, 1096: 1,
1095: 1, 1093: 1, 1092: 1, 1090: 1, 1088: 1, 1084: 1, 1081: 1, 1074: 1, 1071: 1, 1069: 1, 1064: 1,
1062: 1, 1057: 1, 1056: 1, 1055: 1, 1054: 1, 1053: 1, 1052: 1, 1051: 1, 1050: 1, 1045: 1, 1037: 1,
1036: 1, 1034: 1, 1032: 1, 1030: 1, 1027: 1, 1022: 1, 1021: 1, 1015: 1, 1012: 1, 1005: 1, 1003: 1,
1002: 1, 999: 1, 995: 1, 994: 1, 993: 1, 992: 1, 989: 1, 988: 1, 985: 1, 983: 1, 979: 1, 978: 1,
975: 1, 971: 1, 970: 1, 968: 1, 939: 1, 935: 1, 934: 1, 933: 1, 928: 1, 925: 1, 921: 1, 918: 1,
916: 1, 914: 1, 911: 1, 906: 1, 905: 1, 901: 1, 895: 1, 888: 1, 887: 1, 880: 1, 878: 1, 877: 1,
874: 1, 865: 1, 862: 1, 861: 1, 858: 1, 854: 1, 853: 1, 852: 1, 848: 1, 846: 1, 844: 1, 841: 1,
837: 1, 831: 1, 830: 1, 829: 1, 828: 1, 818: 1, 815: 1, 813: 1, 809: 1, 805: 1, 793: 1, 792: 1,
790: 1, 787: 1, 782: 1, 779: 1, 776: 1, 760: 1, 759: 1, 751: 1, 750: 1, 748: 1, 743: 1, 740: 1,
736: 1, 728: 1, 720: 1, 715: 1, 712: 1, 707: 1, 704: 1, 700: 1, 698: 1, 697: 1, 691: 1, 690: 1,
666: 1, 665: 1, 651: 1, 622: 1, 620: 1, 608: 1, 606: 1, 599: 1, 577: 1, 573: 1, 565: 1, 555: 1,
531: 1, 529: 1, 527: 1, 500: 1, 385: 1})
```

In [76]:

```
alpha = [10 ** x for x in range(-5, 2)]
alpha.append(15)
alpha.append(20)
alpha.append(40)

cross_val_lgloss = []
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(tf_idf_train_feat, cross_val_y)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    classifier_generate_claibrated.fit(tf_idf_train_feat, cross_val_y)
    predicted_y = classifier_generate_claibrated.predict_proba(tf_idf_cross_val_feat)
    cross_val_lgloss.append(log_loss(label_cross_val_data, predicted_y, labels=clf.classes_))
    print("For alpha value of "+str(i)+" CV log loss = "+str(log_loss(label_cross_val_data, predict
ed_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```

```
For alpha value of 1e-05 CV log loss = 1.5064758775968379
For alpha value of 0.0001 CV log loss = 1.4700399302424787
For alpha value of 0.001 CV log loss = 1.4692512602174146
For alpha value of 0.01 CV log loss = 1.461566819403949
For alpha value of 0.1 CV log loss = 1.4747612683435707
For alpha value of 1 CV log loss = 1.3779810481900128
For alpha value of 10 CV log loss = 1.1367292578836257
For alpha value of 15 CV log loss = 1.1365034460152175
For alpha value of 20 CV log loss = 1.1423904297122562
For alpha value of 40 CV log loss = 1.1722490341971343
```





In [77]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]
clf = SGDClassifier(alpha=best_alpha, penalty='l2', loss='log', random_state=42)
clf.fit(tf_idf_train_feat, cross_val_y)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(tf_idf_train_feat, cross_val_y)

train_final_pred = classifier_generate_claibrated.predict_proba(tf_idf_train_feat)
print('For values of best alpha = ', best_alpha, "the train log loss =:",log_loss(cross_val_y, tra
in_final_pred, labels=clf.classes_))

cross_val_final_pred = classifier_generate_claibrated.predict_proba(tf_idf_cross_val_feat)
print('For values of best alpha = ', best_alpha, "the CV log loss
=:",log_loss(label_cross_val_data, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(tf_idf_test_feat)
print('For values of best alpha = ', best_alpha, "the test log loss =:",log_loss(Y_Test,
test_final_pred, labels=clf.classes_))
```

```
For values of best alpha =  15 the train log loss =: 0.6816928497360148
For values of best alpha =  15 the CV log loss =: 1.1365034460152175
For values of best alpha =  15 the test log loss =: 1.0888379907457424
```

In [78]:

```
def feat_comon(df):
    vect_count = CountVectorizer(min_df = 3)
    bag_words = vect_count.fit_transform(df["Text"])
    bag_words_feat = vect_count.get_feature_names()
    bag_words_uniqfeat = len(set(bag_words_feat))
    feat_com = len(set(TrainFeatures) & set(bag_words_feat))
    return bag_words_uniqfeat, feat_com
```

In [79]:

```
vect_count = CountVectorizer(min_df = 3)
bag_words = vect_count.fit_transform(X_Test["Text"])
bag_words_feat = vect_count.get_feature_names()
```

In [80]:

```
len1, len2 = feat_comon(X_Test)
print("Percentage of common features in train and test data = "+str(np.round((len2/len1)*100, 4))+
"%")
#it prints: Out of total features in test data, how many features are also present in train data
len3, len4 = feat_comon(cross_val_data)
print("Percentage of common features in train and CV data = "+str(np.round((len4/len3)*100, 4))+"%
")
#it prints: Out of total features in CV data, how many features are also present in train data
```

```
Percentage of common features in train and test data = 95.6438%
Percentage of common features in train and CV data = 97.5012%
```

Machine Learning Models

In [ ]:

```
def find_feat_importance(indices, gene, variation, text, noOfFeatures):
    new_gen_vect = CountVectorizer()
    new_varvect = CountVectorizer()
    new_textvect = TfidfVectorizer(ngram_range = (1,2), stop_words = "english", min_df = 3,
max_features = 100000)

    new_gen_vect_final = new_gen_vect.fit(final_train['Gene'])
    new_varvect_final  = new_varvect.fit(final_train['Variation'])
    new_textvect_final = new_textvect.fit(final_train['Text'])

    final_genfeat = new_gen_vect.get_feature_names()
    final_varfeat = varVect.get_feature_names()
    final_textfeat = new_textvect.get_feature_names()

    final_genfeatlen = len(final_genfeat)
    final_feat_varlen = len(final_varfeat)

    First = [x1 for x1 in range(0, 491, 10)]
    del First[1]
    Second = [x2 for x2 in range(1, 492, 10)]
    del Second[1]
    Third = [x3 for x3 in range(2, 493, 10)]
    del Third[1]

    word_present = 0
    for i, v in enumerate(indices):
        if v < final_genfeatlen:
            word = final_genfeat[v]

            if word == gene:
                word_present += 1

                if i in First:
                    print("{}st Gene feature [{}] is present in query point".format(i+1, word))

                elif i in Second:
                    print("{}nd Gene feature [{}] is present in query point".format(i+1, word))

                elif i in Third:
                    print("{}rd Gene feature [{}] is present in query point".format(i+1, word))

                else:
                    print("{}th Gen feature [{}] is present in query point".format(i+1, word))

        elif v < gene_feat_len + final_feat_varlen:
            word = variation_features[v - gene_feat_len]

            if word == variation:
                word_present += 1
                if i in First:
                    print("{}st Variation feature [{}] is present in query point".format(i+1, word)
)

                elif i in Second:
                    print("{}nd Variation feature [{}] is present in query point".format(i+1, word)
)

                elif i in Third:
                    print("{}rd Variation feature [{}] is present in query point".format(i+1, word)
)

                else:
                    print("{}th Variation feature [{}] is present in query point".format(i+1, word)
)
        else:
            word = final_textfeat[v - (gene_feat_len + final_feat_varlen)]

            if word in text.split():
                word_present += 1

                if i in First:
                    print("{}st Text feature [{}] is present in query point".format(i+1, word))

                elif i in Second:
                    print("{}nd Text feature [{}] is present in query point".format(i+1, word))
```

```
                elif i in Third:
                    print("{}rd Text feature [{}] is present in query point".format(i+1, word))

                else:
                    print("{}th Text feature [{}] is present in query point".format(i+1, word))

    print("-"*63)
    print("Out of the top "+str(noOfFeatures)+" features "+str(word_present)+" are present in
query point")
```

In [40]:

```
table = pd.DataFrame(columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis-Cla
ssified CV", "Mis-Classified Test", "Remarks"])
```

Stacking three types of classifier

In [84]:

```
gene_train_encoding_oneehot = hstack((train_gene_feature_onehotCoding,
train_variation_feature_onehotCoding))
gene_crossval_encoding_oneehot = hstack((cv_gene_feature_onehotCoding,
cv_variation_feature_onehotCoding))
gene_test_encoding_oneehot = hstack((test_gene_feature_onehotCoding,
test_variation_feature_onehotCoding))

gene_final_X_TRAIN = hstack((gene_train_encoding_oneehot, tf_idf_train_feat))
gene_final_X_TRAIN = gene_final_X_TRAIN.tocsr()
gene_final_Y_TRAIN = np.array(list(cross_val_y))

gene_final_crossval_X = hstack((gene_crossval_encoding_oneehot, tf_idf_cross_val_feat))
gene_final_crossval_X = gene_final_crossval_X.tocsr()
gene_final_crossval_Y = np.array(list(label_cross_val_data))

gene_final_X_TEST = hstack((gene_test_encoding_oneehot, tf_idf_test_feat))
gene_final_X_TEST = gene_final_X_TEST.tocsr()
gene_final_Y_TEST = np.array(list(Y_Test))
```

In [85]:

```
print("Shape of One hot encoded Gene and Variation, TFIDF text stacked vector for Train Data = "+s
tr(gene_final_X_TRAIN.shape))
print("Shape of One hot encoded Gene and Variation, TFIDF text stacked vector for CV Data = "+str(
gene_final_crossval_X.shape))
print("Shape of One hot encoded Gene and Variation, TFIDF text stacked vector for Test Data = "+st
r(gene_final_X_TEST.shape))
```

```
Shape of One hot encoded Gene and Variation, TFIDF text stacked vector for Train Data = (2121, 102
191)
Shape of One hot encoded Gene and Variation, TFIDF text stacked vector for CV Data = (531, 102191)
Shape of One hot encoded Gene and Variation, TFIDF text stacked vector for Test Data = (664, 10219
1)
```

In [86]:

```
train_gene_var_responseCoded = np.hstack((gene_train_encoding,
train_variation_feature_responseCoding))
cv_gene_var_responseCoded = np.hstack((gene_crossval_encoding, cv_variation_feature_responseCoding
))
test_gene_var_responseCoded = np.hstack((gene_test_encoding, test_variation_feature_responseCoding
))

gene_final_X_TRAIN_ResponseCoded = np.hstack((train_gene_var_responseCoded,
train_text_feature_responseCoding))
gene_final_crossval_X_ResponseCoded = np.hstack((cv_gene_var_responseCoded,
cv_text_feature_responseCoding))
gene_final_X_TEST_ResponseCoded = np.hstack((test_gene_var_responseCoded,
test_text_feature_responseCoding))
```

In [87]:

```
print("Shape of Response Coded Gene and Variation, TFIDF text stacked vector for Train Data = "+st
r(gene_final_X_TRAIN_ResponseCoded.shape))
print("Shape of Response Coded Gene and Variation, TFIDF text stacked vector for CV Data = "+str(g
ene_final_crossval_X_ResponseCoded.shape))
print("Shape of Response Coded Gene and Variation, TFIDF text stacked vector for Test Data = "+str
(gene_final_X_TEST_ResponseCoded.shape))
```

```
Shape of Response Coded Gene and Variation, TFIDF text stacked vector for Train Data = (2121, 27)
Shape of Response Coded Gene and Variation, TFIDF text stacked vector for CV Data = (531, 27)
Shape of Response Coded Gene and Variation, TFIDF text stacked vector for Test Data = (664, 27)
```

Base Models

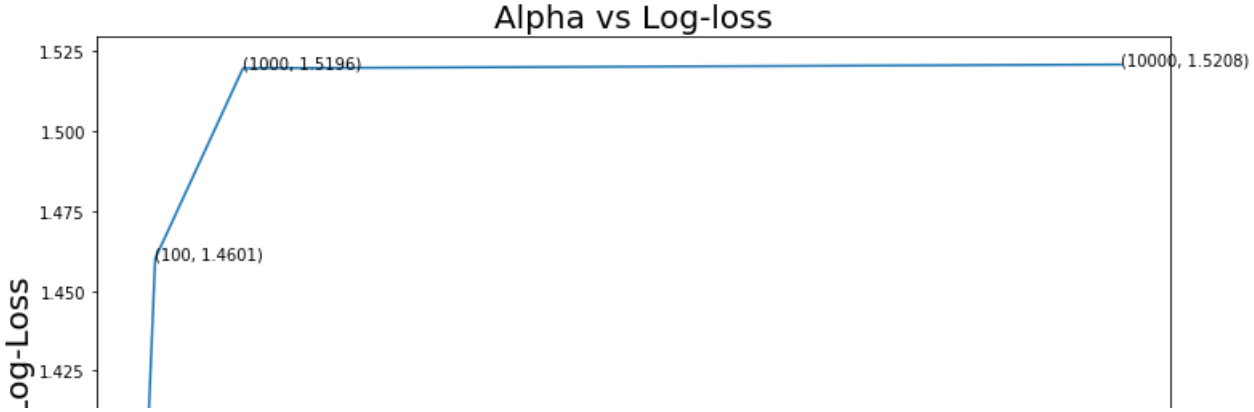
Naive Bayes

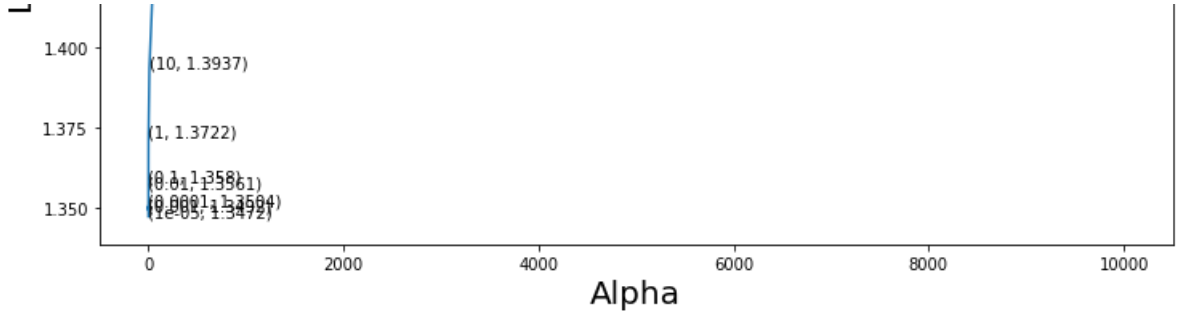
Hyper-Parameter Tuning

In [117]:

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

cross_val_lgloss = []
for i in alpha:
    clf = MultinomialNB(alpha=i)
    clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    predicted_y = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
    cross_val_lgloss.append(log_loss(gene_final_crossval_Y, predicted_y, labels=clf.classes_))
    print("For alpha value of "+str(i)+" CV log loss = "+str(log_loss(gene_final_crossval_Y, predic
ted_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```

```
For alpha value of 1e-05 CV log loss = 1.3471712144052246
For alpha value of 0.0001 CV log loss = 1.3504495385459239
For alpha value of 0.001 CV log loss = 1.3491828952854628
For alpha value of 0.01 CV log loss = 1.3560756526469306
For alpha value of 0.1 CV log loss = 1.357974432285551
For alpha value of 1 CV log loss = 1.372217919431397
For alpha value of 10 CV log loss = 1.3936886399206103
For alpha value of 100 CV log loss = 1.4600569331000073
For alpha value of 1000 CV log loss = 1.519593771555386
For alpha value of 10000 CV log loss = 1.5207830444039192
```





Testing with best hyper-parameter

In [118]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]
clf = MultinomialNB(alpha=best_alpha)
clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)

train_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TRAIN)
print('For values of best alpha = ', best_alpha, "the train log loss
=:",log_loss(gene_final_Y_TRAIN, train_final_pred, labels=clf.classes_))

cross_val_final_pred = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
print('For values of best alpha = ', best_alpha, "the CV log loss
=:",log_loss(gene_final_crossval_Y, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TEST)
print('For values of best alpha = ', best_alpha, "the test log loss =:",log_loss(gene_final_Y_TEST
, test_final_pred, labels=clf.classes_))
```

```
For values of best alpha =  1e-05 the train log loss =: 0.903102740053625
For values of best alpha =  1e-05 the CV log loss =: 1.3471712144052246
For values of best alpha =  1e-05 the test log loss =: 1.2976410460598533
```

In [119]:

```
print("Percentage of mis-classified for CV points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_crossval_X) -
gene_final_crossval_Y)/gene_final_crossval_X.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_X_TEST) -
gene_final_Y_TEST)/gene_final_X_TEST.shape[0]*100), 2))+"%")
```

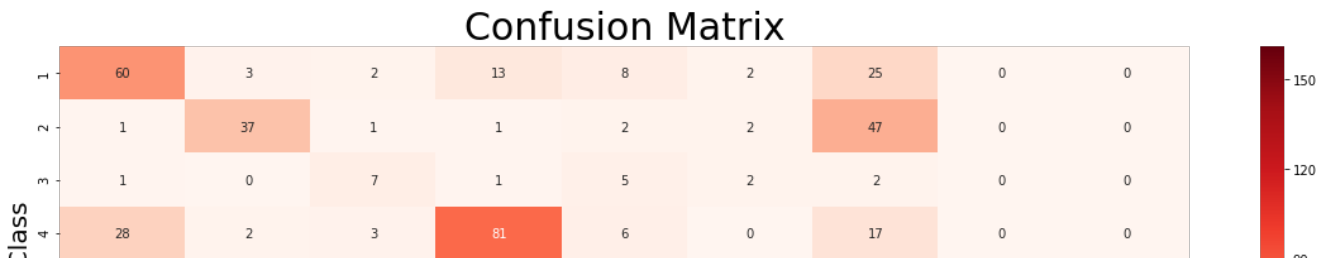
```
Percentage of mis-classified for CV points = 42.75%
Percentage of mis-classified for Test points = 39.46%
```

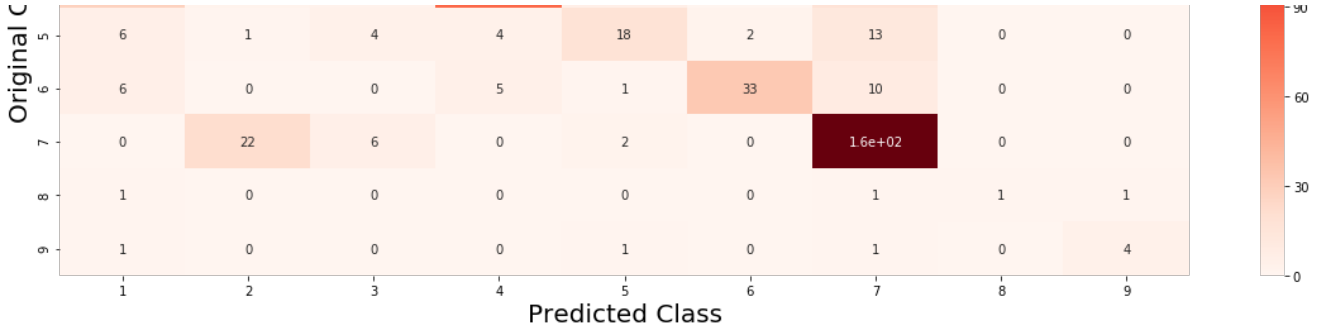
In [41]:

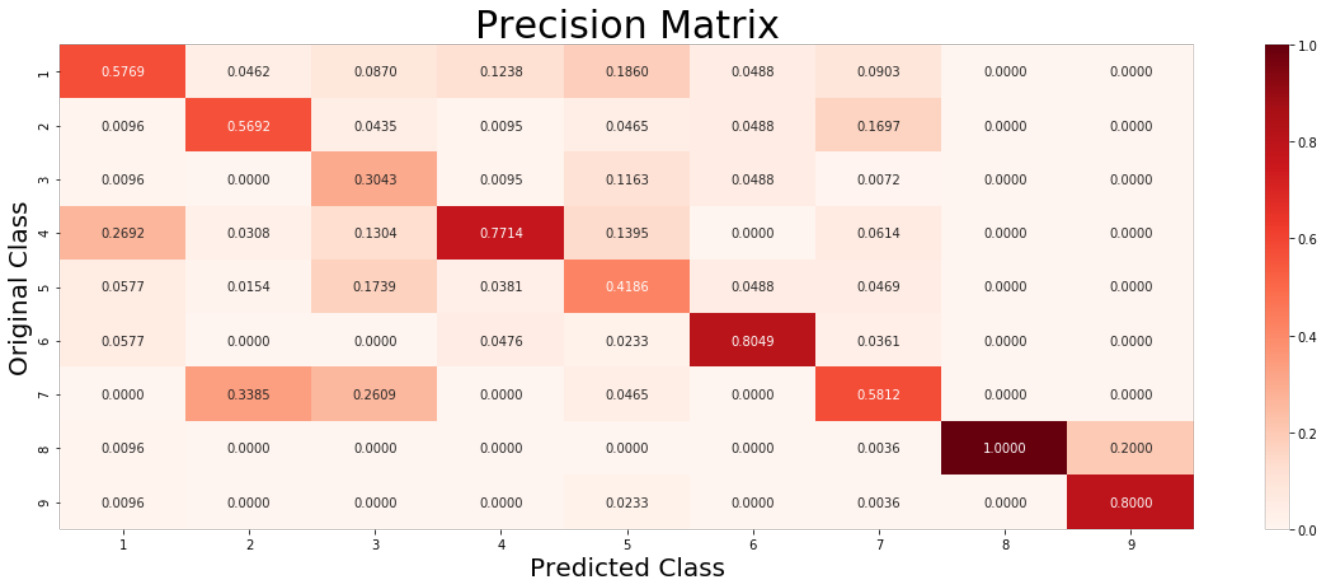
```
table = table.append(pd.DataFrame([["Naive Bayes", 0.9031, 1.3471, 1.2976, "42.75%", "39.46%", "Goo
dFit"]], columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis-Classified CV",
"Mis-Classified Test", "Remarks"]))
```

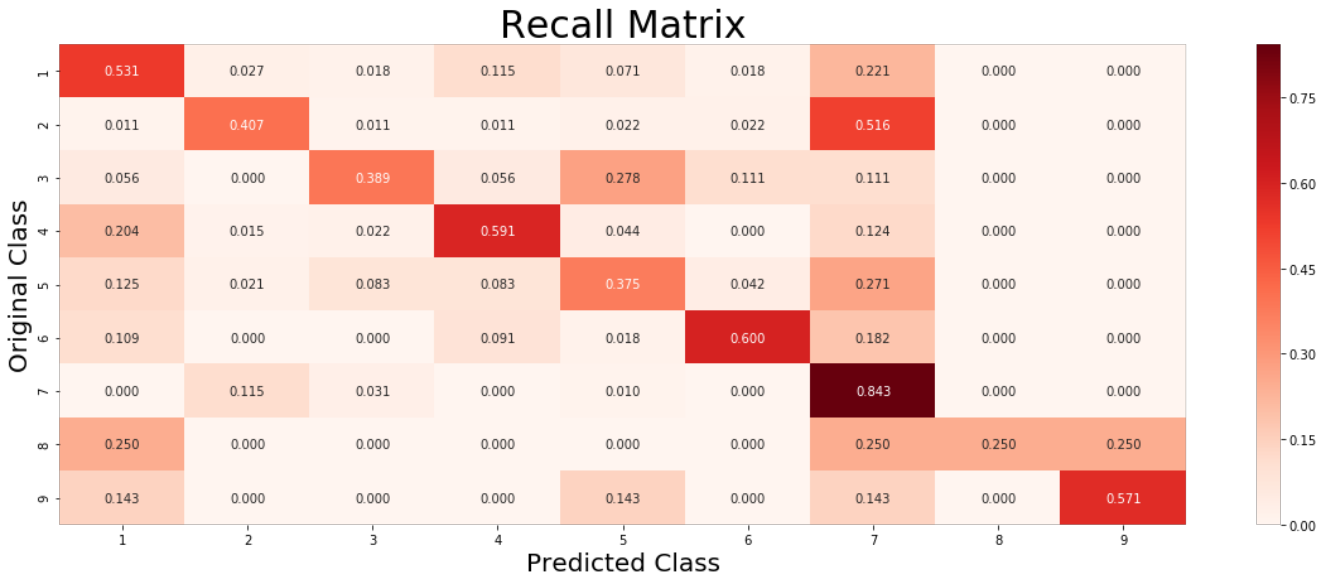
In [120]:

```
generate_conf_mat(gene_final_Y_TEST, classifier_generate_claibrated.predict(gene_final_X_TEST))
```









first 100 features of correctly classified point

In [122]:

```
dataset_test_check = 5
no_feature = 100
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
```

```
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 4)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 4
Predicted Probabilities for test point = [[0.0818 0.0835 0.0078 0.6667 0.038  0.0381 0.0758 0.0047
0.0035]]
True class label for test point = 4
---------------------------------------------------------------
1st Text feature [shown] is present in query point
2nd Text feature [results] is present in query point
3rd Text feature [protein] is present in query point
4th Text feature [described] is present in query point
5th Text feature [activity] is present in query point
6th Text feature [using] is present in query point
7th Text feature [proteins] is present in query point
8th Text feature [determined] is present in query point
9th Text feature [type] is present in query point
10th Text feature [analysis] is present in query point
11th Text feature [mutations] is present in query point
12th Text feature [important] is present in query point
13th Text feature [previously] is present in query point
14th Text feature [addition] is present in query point
15th Text feature [indicated] is present in query point
16th Text feature [experiments] is present in query point
17th Text feature [10] is present in query point
18th Text feature [indicate] is present in query point
19th Text feature [cells] is present in query point
20th Text feature [suggest] is present in query point
21st Text feature [similar] is present in query point
22nd Text feature [expressed] is present in query point
23rd Text feature [acid] is present in query point
24th Text feature [function] is present in query point
25th Text feature [loss] is present in query point
27th Text feature [wild] is present in query point
28th Text feature [levels] is present in query point
29th Text feature [expression] is present in query point
30th Text feature [30] is present in query point
31st Text feature [discussion] is present in query point
32nd Text feature [analyzed] is present in query point
33rd Text feature [amino] is present in query point
34th Text feature [performed] is present in query point
36th Text feature [containing] is present in query point
37th Text feature [determine] is present in query point
38th Text feature [vitro] is present in query point
39th Text feature [lower] is present in query point
41st Text feature [ability] is present in query point
42nd Text feature [15] is present in query point
43rd Text feature [result] is present in query point
44th Text feature [possible] is present in query point
45th Text feature [functions] is present in query point
46th Text feature [reduced] is present in query point
47th Text feature [effects] is present in query point
48th Text feature [associated] is present in query point
49th Text feature [compared] is present in query point
50th Text feature [role] is present in query point
51st Text feature [reported] is present in query point
52nd Text feature [used] is present in query point
53rd Text feature [respectively] is present in query point
54th Text feature [previous] is present in query point
55th Text feature [control] is present in query point
56th Text feature [mutation] is present in query point
57th Text feature [different] is present in query point
58th Text feature [introduction] is present in query point
59th Text feature [contribute] is present in query point
60th Text feature [suggesting] is present in query point
61st Text feature [contrast] is present in query point
62nd Text feature [table] is present in query point
63rd Text feature [did] is present in query point
64th Text feature [involved] is present in query point
65th Text feature [critical] is present in query point
```

```
66th Text feature [including] is present in query point
67th Text feature [high] is present in query point
68th Text feature [cell] is present in query point
69th Text feature [incubated] is present in query point
70th Text feature [indicating] is present in query point
71st Text feature [missense] is present in query point
72nd Text feature [purified] is present in query point
73rd Text feature [figure] is present in query point
74th Text feature [directly] is present in query point
75th Text feature [transfected] is present in query point
76th Text feature [human] is present in query point
79th Text feature [generated] is present in query point
80th Text feature [resulting] is present in query point
81st Text feature [data] is present in query point
83rd Text feature [according] is present in query point
84th Text feature [observed] is present in query point
85th Text feature [standard] is present in query point
86th Text feature [highly] is present in query point
87th Text feature [fig] is present in query point
88th Text feature [vivo] is present in query point
89th Text feature [transfection] is present in query point
90th Text feature [mutant] is present in query point
91st Text feature [functional] is present in query point
92nd Text feature [terminal] is present in query point
93rd Text feature [tested] is present in query point
94th Text feature [specific] is present in query point
95th Text feature [required] is present in query point
96th Text feature [fact] is present in query point
97th Text feature [vector] is present in query point
98th Text feature [bind] is present in query point
99th Text feature [dependent] is present in query point
100th Text feature [study] is present in query point
----------------------------------------------------------------------
Out of the top 100 features 94 are present in query point
```

Incorrectly classifying points

In [124]:

```
dataset_test_check = 10
no_feature = 100
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 4)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point ", end='')
print(final_pred_new_probs[0])
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-clf.coef_)[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 5
Predicted Probabilities for test point [0.105  0.107  0.0101 0.1295 0.4922 0.049  0.0967 0.006
0.0044]
True class label for test point = 3
----------------------------------------------------------------------
1st Text feature [results] is present in query point
2nd Text feature [based] is present in query point
3rd Text feature [introduction] is present in query point
4th Text feature [used] is present in query point
5th Text feature [data] is present in query point
6th Text feature [analysis] is present in query point
7th Text feature [discussion] is present in query point
8th Text feature [functional] is present in query point
9th Text feature [addition] is present in query point
10th Text feature [using] is present in query point
12th Text feature [assays] is present in query point
13th Text feature [table] is present in query point
14th Text feature [effect] is present in query point
16th Text feature [assay] is present in query point
17th Text feature [variants] is present in query point
```

```
17th Text feature [variants] is present in query point
18th Text feature [previously] is present in query point
19th Text feature [shown] is present in query point
20th Text feature [large] is present in query point
21st Text feature [control] is present in query point
22nd Text feature [protein] is present in query point
23rd Text feature [likely] is present in query point
24th Text feature [possible] is present in query point
25th Text feature [type] is present in query point
26th Text feature [controls] is present in query point
27th Text feature [assess] is present in query point
29th Text feature [include] is present in query point
32nd Text feature [tested] is present in query point
34th Text feature [nhgri] is present in query point
36th Text feature [10] is present in query point
38th Text feature [comparison] is present in query point
39th Text feature [highly] is present in query point
41st Text feature [research] is present in query point
43rd Text feature [genetic] is present in query point
45th Text feature [methods] is present in query point
46th Text feature [cancer] is present in query point
47th Text feature [different] is present in query point
49th Text feature [neutral] is present in query point
51st Text feature [provide] is present in query point
52nd Text feature [wild] is present in query point
54th Text feature [sequence] is present in query point
55th Text feature [analyzed] is present in query point
57th Text feature [known] is present in query point
58th Text feature [similar] is present in query point
59th Text feature [allow] is present in query point
60th Text feature [15] is present in query point
62nd Text feature [additional] is present in query point
63rd Text feature [published] is present in query point
64th Text feature [available] is present in query point
66th Text feature [variant] is present in query point
67th Text feature [compared] is present in query point
68th Text feature [sensitivity] is present in query point
70th Text feature [align] is present in query point
72nd Text feature [way] is present in query point
73rd Text feature [observed] is present in query point
77th Text feature [indicate] is present in query point
78th Text feature [number] is present in query point
79th Text feature [clear] is present in query point
81st Text feature [intermediate] is present in query point
82nd Text feature [important] is present in query point
83rd Text feature [article] is present in query point
84th Text feature [v1736a] is present in query point
86th Text feature [vitro] is present in query point
87th Text feature [studies] is present in query point
89th Text feature [functionally] is present in query point
90th Text feature [previous] is present in query point
92nd Text feature [uncertain] is present in query point
93rd Text feature [effects] is present in query point
96th Text feature [corresponding] is present in query point
99th Text feature [predicted] is present in query point
100th Text feature [function] is present in query point
-----------------------------------------------------------------
Out of the top 100 features 70 are present in query point
```

K Nearest Classification

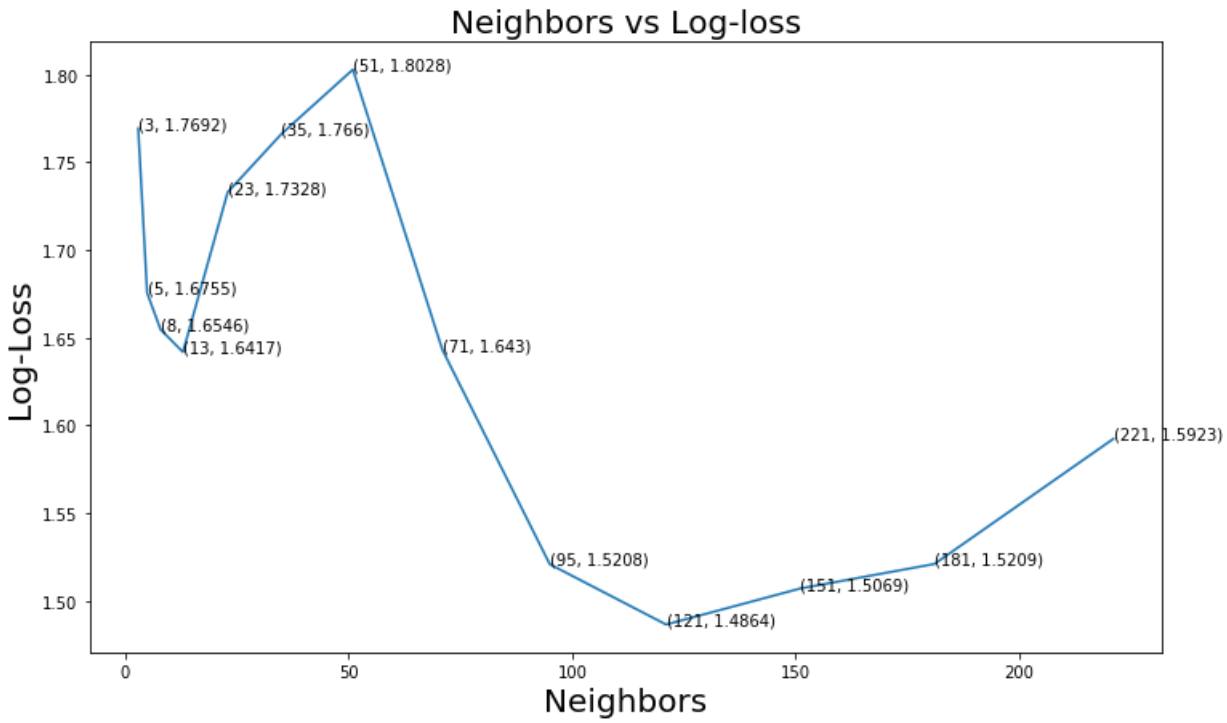
Hyper-Parameter Tuning

In [126]:

```
neighbors = [3, 5, 8, 13, 23, 35, 51, 71, 95, 121, 151, 181, 221]

cross_val_lgloss = []
for i in neighbors:
    clf = KNeighborsClassifier(n_neighbors = i, n_jobs = -1)
    clf.fit(gene_final_X_TRAIN_ResponseCoded, gene_final_Y_TRAIN)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    classifier_generate_claibrated.fit(gene_final_X_TRAIN_ResponseCoded, gene_final_Y_TRAIN)
    predicted_y = classifier_generate_claibrated.predict_proba(gene_final_crossval_X_ResponseCoded
```

```
)
    cross_val_lgloss.append(log_loss(gene_final_crossval_Y, predicted_y, labels=clf.classes_))
    print("For Neighbor value of "+str(i)+" CV log loss = "+str(log_loss(gene_final_crossval_Y, pre
dicted_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(neighbors, cross_val_lgloss)
for xy in zip(neighbors, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Neighbors vs Log-loss", fontsize = 20)
plt.xlabel("Neighbors", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```

```
For Neighbor value of 3 CV log loss = 1.7692214013523657
For Neighbor value of 5 CV log loss = 1.6754501759178058
For Neighbor value of 8 CV log loss = 1.6545776360047968
For Neighbor value of 13 CV log loss = 1.641749372341594
For Neighbor value of 23 CV log loss = 1.7328293508503723
For Neighbor value of 35 CV log loss = 1.766035963843943
For Neighbor value of 51 CV log loss = 1.8028255611665078
For Neighbor value of 71 CV log loss = 1.6429965277783
For Neighbor value of 95 CV log loss = 1.5207840245105297
For Neighbor value of 121 CV log loss = 1.486401438577594
For Neighbor value of 151 CV log loss = 1.5069418199773952
For Neighbor value of 181 CV log loss = 1.5209270496094205
For Neighbor value of 221 CV log loss = 1.5922847082821472
```



Testing with best hyper-parameter

In [127]:

```
best_neighbors = neighbors[np.argmin(cross_val_lgloss)]
clf = KNeighborsClassifier(n_neighbors = best_neighbors, n_jobs = -1)
clf.fit(gene_final_X_TRAIN_ResponseCoded, gene_final_Y_TRAIN)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(gene_final_X_TRAIN_ResponseCoded, gene_final_Y_TRAIN)

train_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TRAIN_ResponseCoded)
print('For values of best neighbors = ', best_neighbors, "the train log loss
=:",log_loss(gene_final_Y_TRAIN, train_final_pred, labels=clf.classes_))

cross_val_final_pred =
classifier_generate_claibrated.predict_proba(gene_final_crossval_X_ResponseCoded)
print('For values of best neighbors = ', best_neighbors, "the CV log loss
```

```
print('For values of best neighbors = ', best_neighbors, "the CV log loss
=:",log_loss(gene_final_crossval_Y, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TEST_ResponseCoded)
print('For values of best neighbors = ', best_neighbors, "the test log loss
=:",log_loss(gene_final_Y_TEST, test_final_pred, labels=clf.classes_))
```

```
For values of best neighbors =  121 the train log loss =: 0.09521210390493469
For values of best neighbors =  121 the CV log loss =: 1.486401438577594
For values of best neighbors =  121 the test log loss =: 1.5292688269258228
```

In [141]:

```
print("Percentage of mis-classified for CV points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_crossval_X_Respc
seCoded) - gene_final_crossval_Y)/gene_final_crossval_X_ResponseCoded.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_X_TEST_ResponseC
ded) - gene_final_Y_TEST)/gene_final_X_TEST_ResponseCoded.shape[0]*100), 2))+"%")
```

```
Percentage of mis-classified for CV points = 45.39%
Percentage of mis-classified for Test points = 44.88%
```

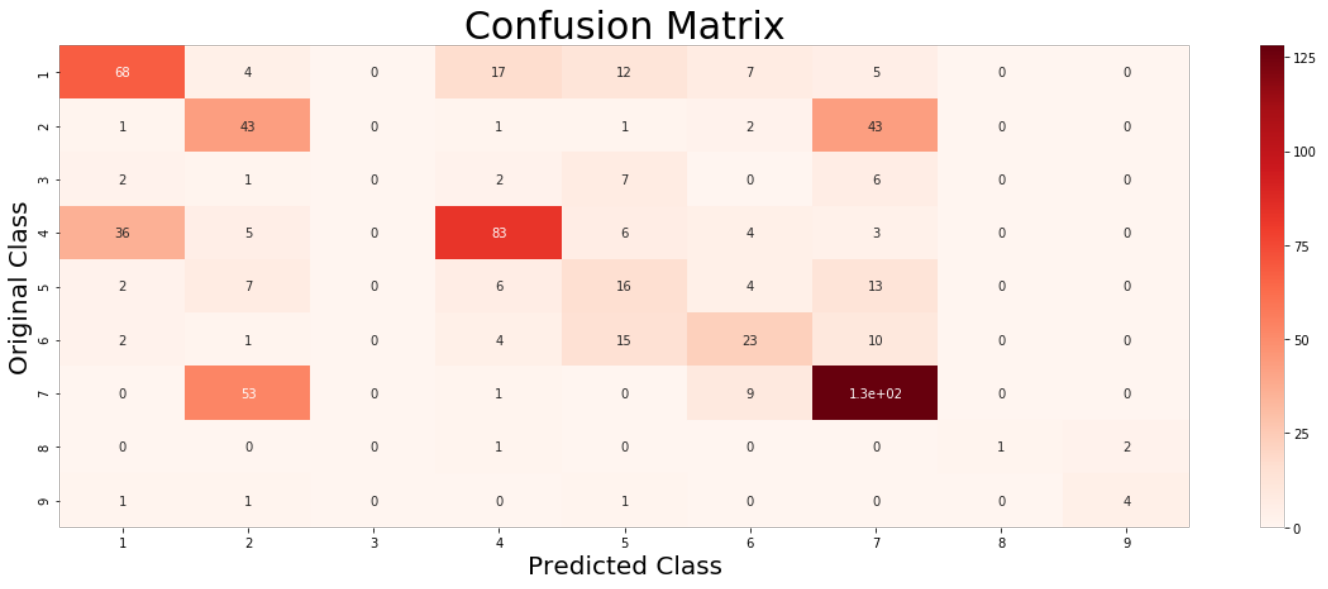
In [42]:

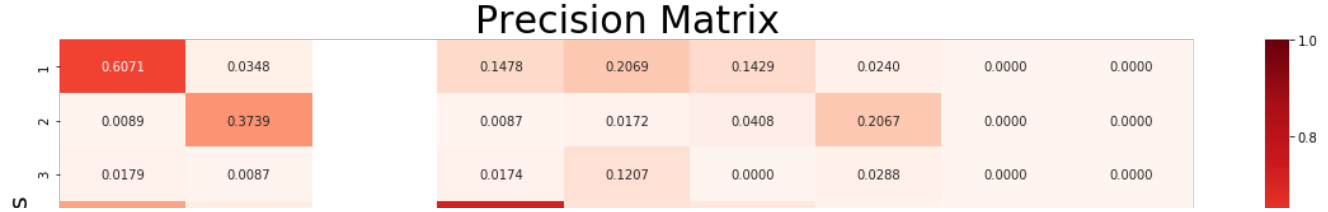
```
table = table.append(pd.DataFrame([["KNN", 0.0952, 1.4864, 1.529, "45.39%", "44.88%", "OverFit"]],
columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis-Classified CV", "Mis-Cla
ssified Test", "Remarks"]))
```

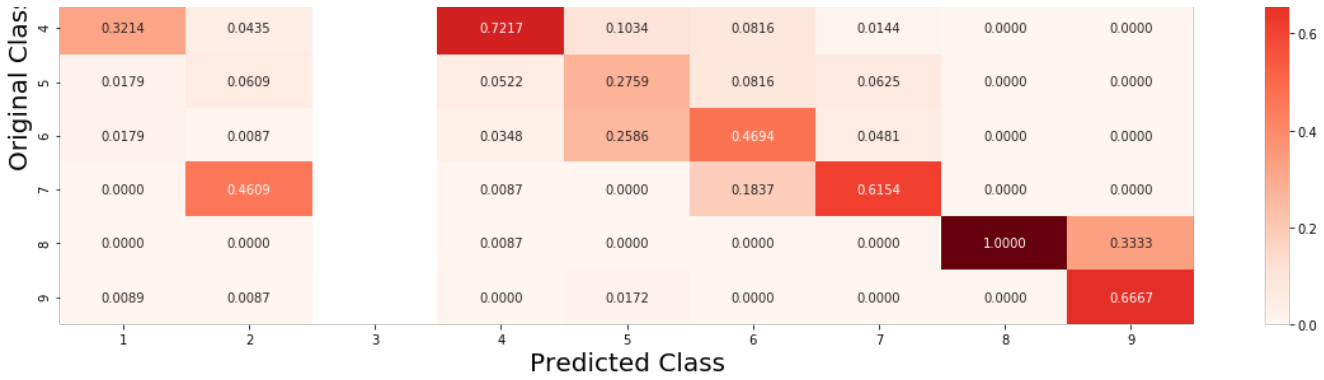
In [130]:

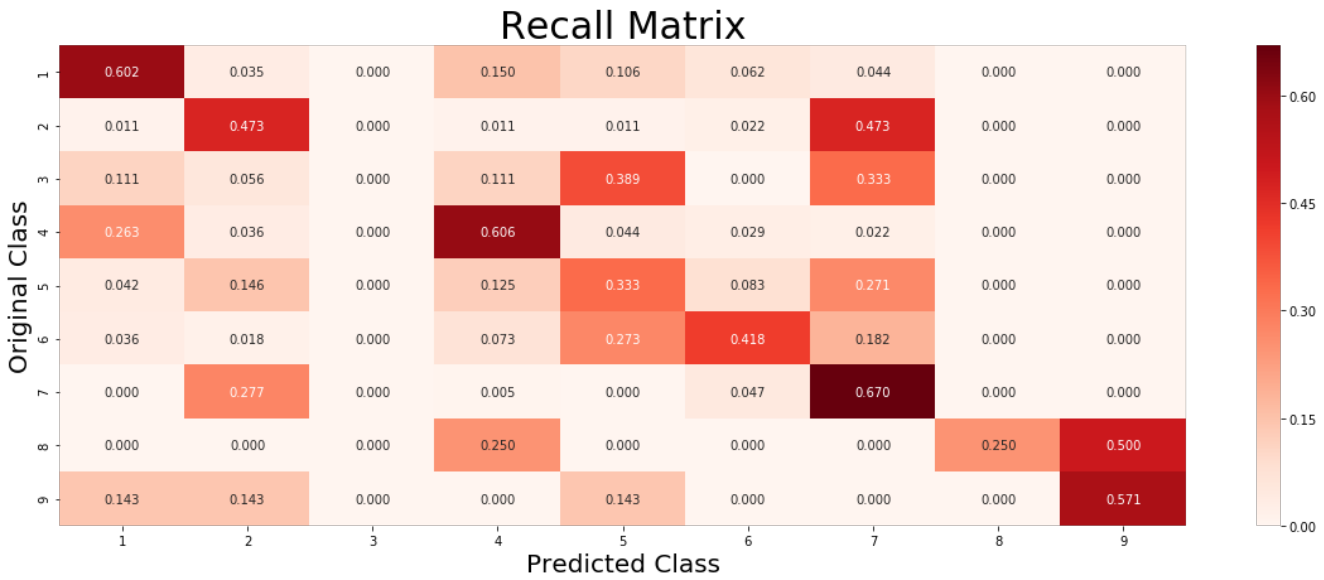
```
generate_conf_mat(gene_final_Y_TEST,
classifier_generate_claibrated.predict(gene_final_X_TEST_ResponseCoded))
```

```
C:\Users\GauravP\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: RuntimeWarning: invalid
value encountered in true_divide
  after removing the cwd from sys.path.
```









Checking Nearest Neighbors for correctly classified point

In [131]:

```
dataset_test_check = 35
predicted_cls =
classifier_generate_claibrated.predict(gene_final_X_TEST_ResponseCoded[dataset_test_check].reshape
(1, -1)) #it will convert vector into 2D.
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs =
np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST_ResponseCoded[dataset_test_
check].reshape(1, -1)), 2)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point ", end='')
print(final_pred_new_probs[0])
print("True class label for test point = "+str(correct_lable_new)+"\n")
nearest_neighbor_points =
clf.kneighbors(gene_final_X_TEST_ResponseCoded[dataset_test_check].reshape(1, -1), n_neighbors = be
st_neighbors, return_distance = False)
print("Labels of nearest neighbors to test points =
"+str(gene_final_Y_TRAIN[nearest_neighbor_points][0]))
print("Class Label: Number of neighboring points =
"+str(Counter(gene_final_Y_TRAIN[nearest_neighbor_points][0])))
```

```
Predicted Class label for test point = 4
Predicted Probabilities for test point [0.   0.01 0.   0.98 0.   0.01 0.   0.   0.  ]
True class label for test point = 4

Labels of nearest neighbors to test points = [4 4 4 4 4 4 4 4 4 4 1 2 4 4 7 6 2 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 1 1
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 3 3 3 3 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4]
```

```
 1 1 1 1 1 1 1 1 1 1]
Class Label: Number of neighboring points = Counter({4: 107, 1: 5, 3: 4, 2: 2, 7: 1, 6: 1, 5: 1})
```

Checking Nearest Neighbors for incorrectly classified point

In [142]:

```
dataset_test_check = 100
predicted_cls =
classifier_generate_claibrated.predict(gene_final_X_TEST_ResponseCoded[dataset_test_check].reshape
(1, -1)) #it will convert vector into 2D.
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs =
np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST_ResponseCoded[dataset_test_
check].reshape(1, -1)), 2)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point ", end='')
print(final_pred_new_probs[0])
print("True class label for test point = "+str(correct_lable_new)+"\n")
nearest_neighbor_points =
clf.kneighbors(gene_final_X_TEST_ResponseCoded[dataset_test_check].reshape(1, -1), n_neighbors = be
st_neighbors, return_distance = False)
print("The best value of nearest neighbors is "+str(best_neighbors)+" and class labels of those
nearest neighbors to test points = "+str(gene_final_Y_TRAIN[nearest_neighbor_points][0])+"\n")
print("Class Label: Number of neighboring points =
"+str(Counter(gene_final_Y_TRAIN[nearest_neighbor_points][0])))
```

```
Predicted Class label for test point = 1
Predicted Probabilities for test point [0.75 0.03 0.01 0.15 0.01 0.05 0.   0.01 0.  ]
True class label for test point = 4

The best value of nearest neighbors is 121 and class labels of those nearest neighbors to test
points = [2 1 6 4 4 4 1 4 1 1 1 1 4 7 1 1 1 1 1 1 1 5 6 5 6 6 1 1 1 1 5 6 6 1 1 1 1
 1 1 1 1 1 1 5 5 6 6 1 4 6 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 6 7 1 1 1 1 1 1
 6 7 1 5 6 7 1 1 1 5 1 1 1 1 1 4 4 4 4 4 5 5 4 4 5 5 6 6 6 6 6 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1]

Class Label: Number of neighboring points = Counter({1: 74, 6: 17, 4: 13, 5: 11, 7: 5, 2: 1})
```

Logistic Regression with class balancing

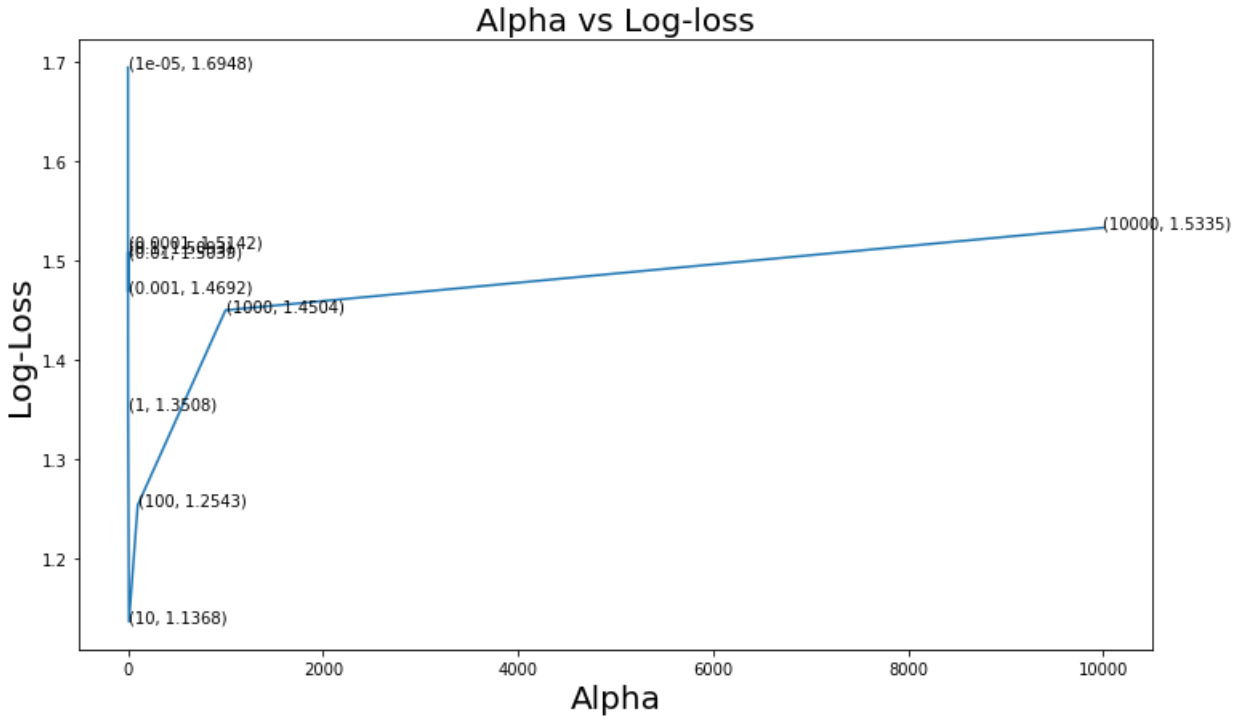
Hyper-Parameter Tuning

In [144]:

```
warnings.simplefilter('ignore')
alpha = [10**x for x in range(-5, 5)]

cross_val_lgloss = []
for i in alpha:
    clf = SGDClassifier(loss = "log", alpha = i, class_weight = "balanced")
    clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    predicted_y = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
    cross_val_lgloss.append(log_loss(gene_final_crossval_Y, predicted_y, labels=clf.classes_))
    print("For alpha value of "+str(i)+" CV log loss = "+str(log_loss(gene_final_crossval_Y, predic
ted_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```

```
For alpha value of 1e-05 CV log loss = 1.694849342601553
```

```
For alpha value of 0.0001 CV log loss = 1.5142490054486
For alpha value of 0.001 CV log loss = 1.4692365455902614
For alpha value of 0.01 CV log loss = 1.5039307671421531
For alpha value of 0.1 CV log loss = 1.5092500171355756
For alpha value of 1 CV log loss = 1.3507614962625538
For alpha value of 10 CV log loss = 1.1368489626548015
For alpha value of 100 CV log loss = 1.2543070732891224
For alpha value of 1000 CV log loss = 1.4504392646114053
For alpha value of 10000 CV log loss = 1.533506015557066
```



Testing with best hyper-parameter

In [145]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]
clf = SGDClassifier(loss = "log", alpha = best_alpha, class_weight = "balanced")
clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)

train_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TRAIN)
print('For values of best alpha = ', best_alpha, "the train log loss
=:",log_loss(gene_final_Y_TRAIN, train_final_pred, labels=clf.classes_))

cross_val_final_pred = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
print('For values of best alpha = ', best_alpha, "the CV log loss
=:",log_loss(gene_final_crossval_Y, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TEST)
print('For values of best alpha = ', best_alpha, "the test log loss =:",log_loss(gene_final_Y_TEST
, test_final_pred, labels=clf.classes_))
```

```
For values of best alpha =  10 the train log loss =: 0.6738349447177868
For values of best alpha =  10 the CV log loss =: 1.1314883600603884
For values of best alpha =  10 the test log loss =: 1.1014944605034171
```

In [146]:

```
print("Percentage of mis-classified for CV points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_crossval_X) -
gene_final_crossval_Y)/gene_final_crossval_X.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_X_TEST) -
gene_final_Y_TEST)/gene_final_X_TEST.shape[0]*100), 2))+"%")
```

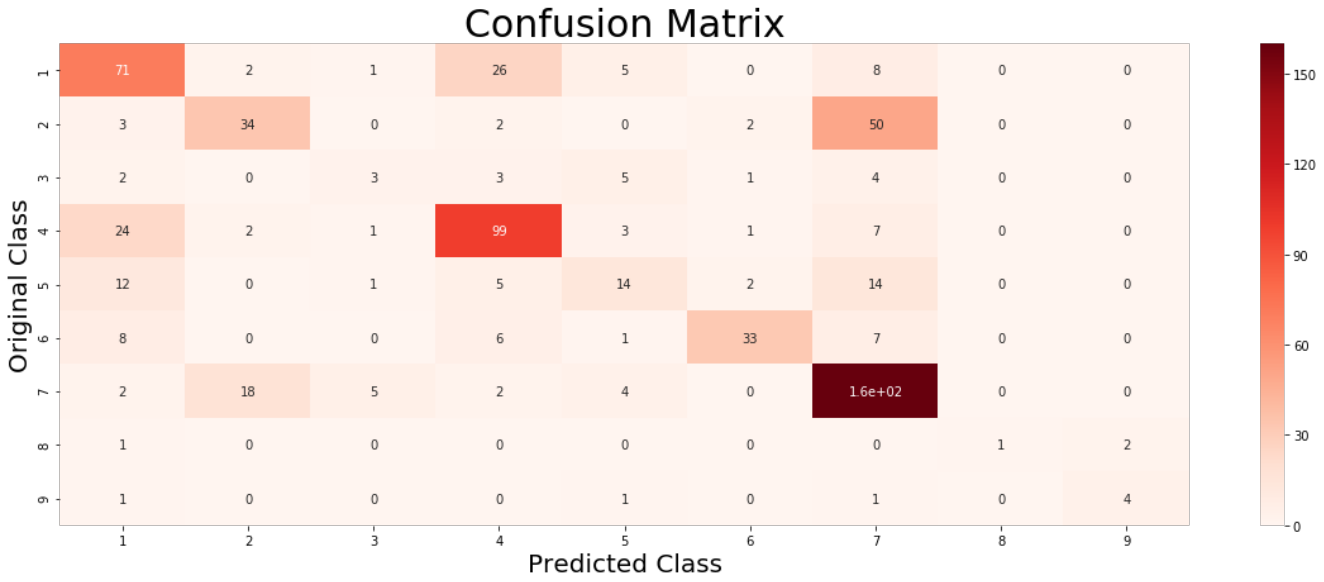
```
Percentage of mis-classified for CV points = 40.49%
Percentage of mis-classified for Test points = 36.9%
```

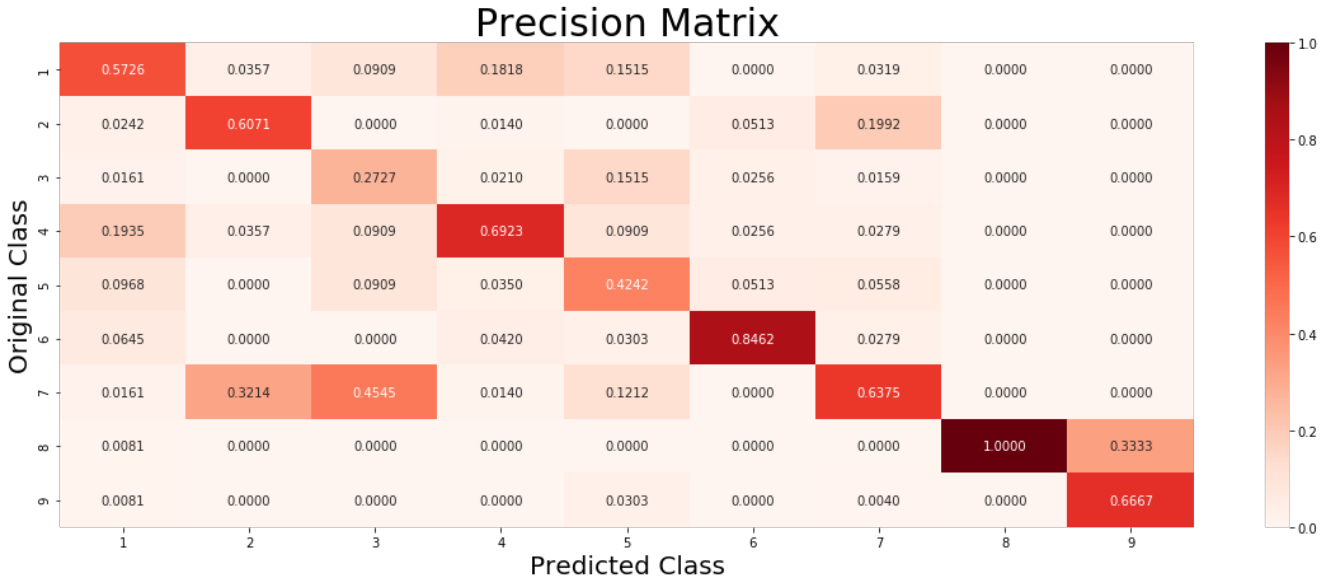
In [48]:

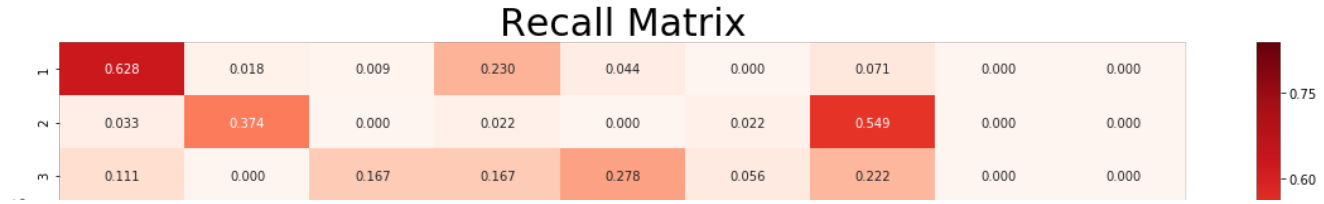
```
table = table.append(pd.DataFrame([["Logistic Regresion(Balanced)", 0.6738, 1.1314, 1.101, "40.49%"
, "36.9%", "Good Fit"]], columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis
-Classified CV", "Mis-Classified Test", "Remarks"]))
```

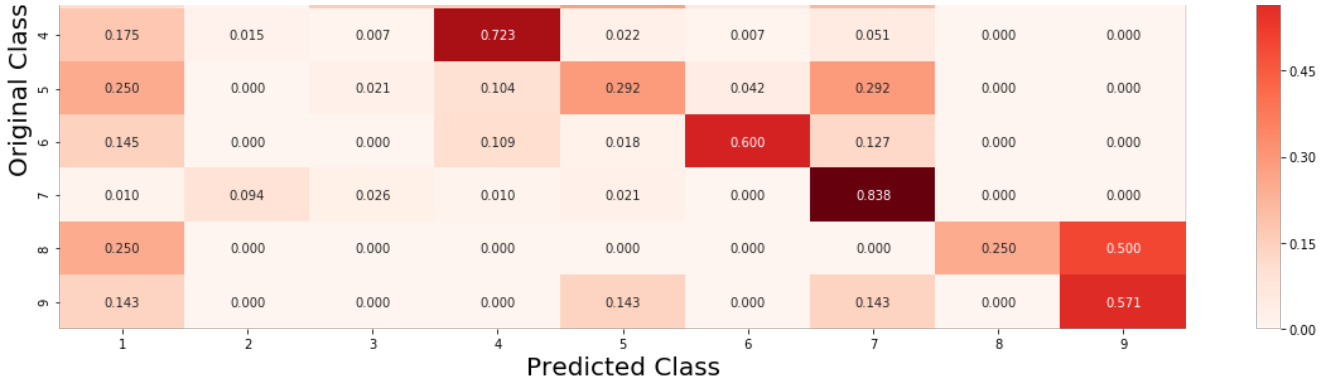
In [148]:

```
generate_conf_mat(gene_final_Y_TEST, classifier_generate_claibrated.predict(gene_final_X_TEST))
```









Checking first 500 important features for correctly classified test point

In [151]:

```
dataset_test_check = 5
no_feature = 500
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 3)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 4
Predicted Probabilities for test point = [[0.025 0.014 0.003 0.922 0.015 0.007 0.013 0.001 0.001]]
True class label for test point = 4
---------------------------------------------------------------
1st Text feature [recently] is present in query point
2nd Text feature [potential] is present in query point
3rd Text feature [addition] is present in query point
4th Text feature [independent] is present in query point
5th Text feature [compared] is present in query point
6th Text feature [activation] is present in query point
7th Text feature [using] is present in query point
8th Text feature [suggests] is present in query point
9th Text feature [kinase] is present in query point
10th Text feature [presence] is present in query point
11th Text feature [shown] is present in query point
12th Text feature [10] is present in query point
13th Text feature [reported] is present in query point
14th Text feature [including] is present in query point
15th Text feature [concentrations] is present in query point
17th Text feature [identification] is present in query point
18th Text feature [currently] is present in query point
19th Text feature [factor] is present in query point
20th Text feature [confirmed] is present in query point
21st Text feature [did] is present in query point
22nd Text feature [cells] is present in query point
23rd Text feature [results] is present in query point
24th Text feature [previously] is present in query point
27th Text feature [study] is present in query point
28th Text feature [additional] is present in query point
29th Text feature [mechanism] is present in query point
30th Text feature [different] is present in query point
31st Text feature [domain] is present in query point
32nd Text feature [respectively] is present in query point
33rd Text feature [activating] is present in query point
34th Text feature [screening] is present in query point
35th Text feature [identified] is present in query point
38th Text feature [small] is present in query point
39th Text feature [22] is present in query point
40th Text feature [report] is present in query point
41st Text feature [novel] is present in query point
43rd Text feature [showed] is present in query point
```

44th Text feature [similar] is present in query point
45th Text feature [therapeutic] is present in query point
46th Text feature [inhibitor] is present in query point
49th Text feature [17] is present in query point
50th Text feature [analysis] is present in query point
52nd Text feature [obtained] is present in query point
55th Text feature [studies] is present in query point
56th Text feature [discussion] is present in query point
58th Text feature [phosphorylation] is present in query point
59th Text feature [increased] is present in query point
60th Text feature [mutation] is present in query point
61st Text feature [observed] is present in query point
62nd Text feature [13] is present in query point
65th Text feature [fusion] is present in query point
66th Text feature [12] is present in query point
67th Text feature [15] is present in query point
70th Text feature [does] is present in query point
72nd Text feature [fish] is present in query point
76th Text feature [confirm] is present in query point
77th Text feature [suggest] is present in query point
78th Text feature [interestingly] is present in query point
82nd Text feature [gene] is present in query point
83rd Text feature [1a] is present in query point
84th Text feature [previous] is present in query point
85th Text feature [figure] is present in query point
87th Text feature [located] is present in query point
88th Text feature [mutations] is present in query point
89th Text feature [highest] is present in query point
90th Text feature [increase] is present in query point
93rd Text feature [clinical] is present in query point
94th Text feature [cell] is present in query point
97th Text feature [important] is present in query point
98th Text feature [performed] is present in query point
99th Text feature [consistent] is present in query point
100th Text feature [approved] is present in query point
103rd Text feature [common] is present in query point
104th Text feature [identify] is present in query point
106th Text feature [molecular] is present in query point
107th Text feature [highly] is present in query point
108th Text feature [result] is present in query point
112nd Text feature [demonstrated] is present in query point
113rd Text feature [total] is present in query point
114th Text feature [treated] is present in query point
115th Text feature [growth] is present in query point
117th Text feature [conformation] is present in query point
118th Text feature [16] is present in query point
119th Text feature [14] is present in query point
122nd Text feature [taken] is present in query point
124th Text feature [expressing] is present in query point
125th Text feature [fig] is present in query point
127th Text feature [transcript] is present in query point
129th Text feature [positive] is present in query point
130th Text feature [mainly] is present in query point
133rd Text feature [indicated] is present in query point
137th Text feature [tumors] is present in query point
138th Text feature [24] is present in query point
147th Text feature [proliferation] is present in query point
155th Text feature [sequences] is present in query point
157th Text feature [include] is present in query point
160th Text feature [model] is present in query point
166th Text feature [needed] is present in query point
167th Text feature [intracellular] is present in query point
170th Text feature [absence] is present in query point
171st Text feature [11] is present in query point
173rd Text feature [derived] is present in query point
174th Text feature [evaluate] is present in query point
175th Text feature [patients] is present in query point
176th Text feature [various] is present in query point
177th Text feature [inhibitors] is present in query point
184th Text feature [table] is present in query point
185th Text feature [18] is present in query point
186th Text feature [time] is present in query point
188th Text feature [inhibited] is present in query point
192nd Text feature [fused] is present in query point
194th Text feature [partners] is present in query point
196th Text feature [findings] is present in query point
202nd Text feature [detected] is present in query point

203rd Text feature [required] is present in query point
205th Text feature [furthermore] is present in query point
206th Text feature [overexpression] is present in query point
207th Text feature [modest] is present in query point
208th Text feature [expression] is present in query point
209th Text feature [human] is present in query point
214th Text feature [data] is present in query point
219th Text feature [non] is present in query point
222nd Text feature [single] is present in query point
224th Text feature [selected] is present in query point
229th Text feature [selective] is present in query point
231st Text feature [confer] is present in query point
233rd Text feature [approximately] is present in query point
235th Text feature [stable] is present in query point
239th Text feature [present] is present in query point
243rd Text feature [active] is present in query point
245th Text feature [primers] is present in query point
247th Text feature [contrast] is present in query point
248th Text feature [pcr] is present in query point
249th Text feature [1c] is present in query point
250th Text feature [form] is present in query point
251st Text feature [factors] is present in query point
255th Text feature [noted] is present in query point
262nd Text feature [sequencing] is present in query point
263rd Text feature [commonly] is present in query point
268th Text feature [distinct] is present in query point
273rd Text feature [event] is present in query point
275th Text feature [inhibition] is present in query point
276th Text feature [32] is present in query point
278th Text feature [www] is present in query point
282nd Text feature [cancer] is present in query point
285th Text feature [activated] is present in query point
286th Text feature [targets] is present in query point
287th Text feature [conditions] is present in query point
291st Text feature [followed] is present in query point
292nd Text feature [review] is present in query point
295th Text feature [stimulation] is present in query point
301st Text feature [analyzed] is present in query point
306th Text feature [occur] is present in query point
307th Text feature [requires] is present in query point
308th Text feature [1b] is present in query point
309th Text feature [known] is present in query point
310th Text feature [developed] is present in query point
319th Text feature [induce] is present in query point
321st Text feature [appear] is present in query point
324th Text feature [described] is present in query point
325th Text feature [samples] is present in query point
326th Text feature [significant] is present in query point
332nd Text feature [example] is present in query point
333rd Text feature [targeted] is present in query point
338th Text feature [remains] is present in query point
342nd Text feature [fold] is present in query point
346th Text feature [coding] is present in query point
351st Text feature [identical] is present in query point
355th Text feature [2b] is present in query point
359th Text feature [established] is present in query point
360th Text feature [21] is present in query point
362nd Text feature [greater] is present in query point
363rd Text feature [27] is present in query point
372nd Text feature [3c] is present in query point
375th Text feature [position] is present in query point
380th Text feature [patient] is present in query point
386th Text feature [sequence] is present in query point
387th Text feature [frame] is present in query point
399th Text feature [used] is present in query point
402nd Text feature [25] is present in query point
406th Text feature [determined] is present in query point
410th Text feature [viable] is present in query point
416th Text feature [point] is present in query point
421st Text feature [region] is present in query point
426th Text feature [clones] is present in query point
429th Text feature [directed] is present in query point
432nd Text feature [block] is present in query point
437th Text feature [directly] is present in query point
440th Text feature [contained] is present in query point
443rd Text feature [fragment] is present in query point
445th Text feature [hum] is present in query point

```
447th Text feature [sensitive] is present in query point
454th Text feature [exon] is present in query point
456th Text feature [harbored] is present in query point
459th Text feature [genetic] is present in query point
463rd Text feature [sequenced] is present in query point
467th Text feature [expressed] is present in query point
468th Text feature [underlying] is present in query point
470th Text feature [having] is present in query point
478th Text feature [20] is present in query point
480th Text feature [chromosome] is present in query point
481st Text feature [based] is present in query point
483rd Text feature [100] is present in query point
486th Text feature [specific] is present in query point
487th Text feature [remain] is present in query point
491st Text feature [line] is present in query point
493rd Text feature [serum] is present in query point
497th Text feature [concentration] is present in query point
498th Text feature [vitro] is present in query point
-----------------------------------------------------------
Out of the top 500 features 209 are present in query point
```

Checking first 500 important features for incorrectly classified test point

In [153]:

```
dataset_test_check = 10
no_feature = 500
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 3)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 5
Predicted Probabilities for test point = [[0.074 0.007 0.005 0.117 0.719 0.073 0.002 0.002 0.   ]]
True class label for test point = 3
---------------------------------------------------------------
2nd Text feature [cells] is present in query point
4th Text feature [expression] is present in query point
7th Text feature [indicating] is present in query point
10th Text feature [presence] is present in query point
12th Text feature [10] is present in query point
14th Text feature [identified] is present in query point
15th Text feature [shown] is present in query point
18th Text feature [potential] is present in query point
21st Text feature [role] is present in query point
22nd Text feature [results] is present in query point
23rd Text feature [figure] is present in query point
24th Text feature [increased] is present in query point
27th Text feature [revealed] is present in query point
32nd Text feature [specific] is present in query point
33rd Text feature [resulting] is present in query point
35th Text feature [dependent] is present in query point
36th Text feature [respectively] is present in query point
38th Text feature [mediated] is present in query point
41st Text feature [compared] is present in query point
42nd Text feature [previously] is present in query point
43rd Text feature [similar] is present in query point
49th Text feature [mutant] is present in query point
51st Text feature [suggest] is present in query point
52nd Text feature [recently] is present in query point
57th Text feature [described] is present in query point
60th Text feature [mutation] is present in query point
63rd Text feature [determined] is present in query point
64th Text feature [encoding] is present in query point
65th Text feature [important] is present in query point
66th Text feature [addition] is present in query point
68th Text feature [decreased] is present in query point
```

66th Text feature [decreased] is present in query point
72nd Text feature [indicate] is present in query point
74th Text feature [proteins] is present in query point
77th Text feature [reduced] is present in query point
78th Text feature [using] is present in query point
79th Text feature [indicated] is present in query point
80th Text feature [direct] is present in query point
86th Text feature [determine] is present in query point
88th Text feature [analysis] is present in query point
89th Text feature [development] is present in query point
90th Text feature [protein] is present in query point
91st Text feature [detected] is present in query point
95th Text feature [fig] is present in query point
96th Text feature [major] is present in query point
99th Text feature [interestingly] is present in query point
100th Text feature [result] is present in query point
107th Text feature [domain] is present in query point
108th Text feature [remains] is present in query point
109th Text feature [cell] is present in query point
121st Text feature [directly] is present in query point
124th Text feature [associated] is present in query point
126th Text feature [observed] is present in query point
128th Text feature [encodes] is present in query point
130th Text feature [negative] is present in query point
136th Text feature [level] is present in query point
139th Text feature [mutations] is present in query point
140th Text feature [target] is present in query point
141st Text feature [regulation] is present in query point
142nd Text feature [involved] is present in query point
146th Text feature [showed] is present in query point
149th Text feature [molecular] is present in query point
150th Text feature [phosphorylation] is present in query point
153rd Text feature [kinase] is present in query point
154th Text feature [functions] is present in query point
157th Text feature [including] is present in query point
158th Text feature [responsible] is present in query point
159th Text feature [expressed] is present in query point
162nd Text feature [suggests] is present in query point
167th Text feature [caused] is present in query point
168th Text feature [positive] is present in query point
183rd Text feature [taken] is present in query point
184th Text feature [amino] is present in query point
186th Text feature [expressing] is present in query point
187th Text feature [derived] is present in query point
191st Text feature [acids] is present in query point
192nd Text feature [single] is present in query point
193rd Text feature [chromosome] is present in query point
196th Text feature [significantly] is present in query point
197th Text feature [30] is present in query point
201st Text feature [strongly] is present in query point
205th Text feature [required] is present in query point
212nd Text feature [green] is present in query point
213rd Text feature [independent] is present in query point
214th Text feature [essential] is present in query point
216th Text feature [type] is present in query point
225th Text feature [containing] is present in query point
232nd Text feature [interacts] is present in query point
234th Text feature [activity] is present in query point
236th Text feature [higher] is present in query point
239th Text feature [contribute] is present in query point
240th Text feature [terminal] is present in query point
241st Text feature [resulted] is present in query point
244th Text feature [ii] is present in query point
245th Text feature [half] is present in query point
249th Text feature [regulate] is present in query point
252nd Text feature [complete] is present in query point
254th Text feature [distinct] is present in query point
255th Text feature [generate] is present in query point
256th Text feature [recent] is present in query point
259th Text feature [generated] is present in query point
261st Text feature [inhibition] is present in query point
262nd Text feature [tissue] is present in query point
264th Text feature [mutants] is present in query point
268th Text feature [site] is present in query point
271st Text feature [western] is present in query point
276th Text feature [13] is present in query point
277th Text feature [patient] is present in query point
280th Text feature [occurs] is present in query point

```
280th Text feature [occurs] is present in query point
282nd Text feature [high] is present in query point
284th Text feature [prepared] is present in query point
286th Text feature [cellular] is present in query point
290th Text feature [near] is present in query point
292nd Text feature [signaling] is present in query point
293rd Text feature [suggesting] is present in query point
294th Text feature [characterized] is present in query point
295th Text feature [thought] is present in query point
296th Text feature [disease] is present in query point
302nd Text feature [normal] is present in query point
303rd Text feature [complex] is present in query point
310th Text feature [led] is present in query point
313rd Text feature [limited] is present in query point
325th Text feature [inhibitor] is present in query point
329th Text feature [deletion] is present in query point
332nd Text feature [tumor] is present in query point
334th Text feature [poor] is present in query point
335th Text feature [15] is present in query point
337th Text feature [discussion] is present in query point
338th Text feature [blot] is present in query point
341st Text feature [multiple] is present in query point
343rd Text feature [cancers] is present in query point
346th Text feature [critical] is present in query point
348th Text feature [end] is present in query point
351st Text feature [inhibit] is present in query point
352nd Text feature [12] is present in query point
357th Text feature [absence] is present in query point
358th Text feature [early] is present in query point
360th Text feature [additional] is present in query point
361st Text feature [university] is present in query point
363rd Text feature [et] is present in query point
366th Text feature [concentrations] is present in query point
371st Text feature [iii] is present in query point
374th Text feature [plays] is present in query point
375th Text feature [common] is present in query point
382nd Text feature [al] is present in query point
384th Text feature [interact] is present in query point
385th Text feature [loss] is present in query point
386th Text feature [phenotype] is present in query point
387th Text feature [cases] is present in query point
389th Text feature [majority] is present in query point
391st Text feature [levels] is present in query point
394th Text feature [antibody] is present in query point
403rd Text feature [40] is present in query point
406th Text feature [contrast] is present in query point
409th Text feature [suggested] is present in query point
416th Text feature [expected] is present in query point
417th Text feature [analyzed] is present in query point
419th Text feature [mouse] is present in query point
422nd Text feature [induced] is present in query point
424th Text feature [mechanisms] is present in query point
431st Text feature [work] is present in query point
437th Text feature [27] is present in query point
454th Text feature [growth] is present in query point
455th Text feature [introduction] is present in query point
458th Text feature [4b] is present in query point
462nd Text feature [leading] is present in query point
464th Text feature [lead] is present in query point
465th Text feature [reported] is present in query point
467th Text feature [downstream] is present in query point
469th Text feature [showing] is present in query point
470th Text feature [synthesis] is present in query point
472nd Text feature [carried] is present in query point
475th Text feature [18] is present in query point
476th Text feature [highly] is present in query point
478th Text feature [sites] is present in query point
481st Text feature [blood] is present in query point
482nd Text feature [factors] is present in query point
487th Text feature [significant] is present in query point
500th Text feature [like] is present in query point
------------------------------------------------------------
Out of the top 500 features 178 are present in query point
```

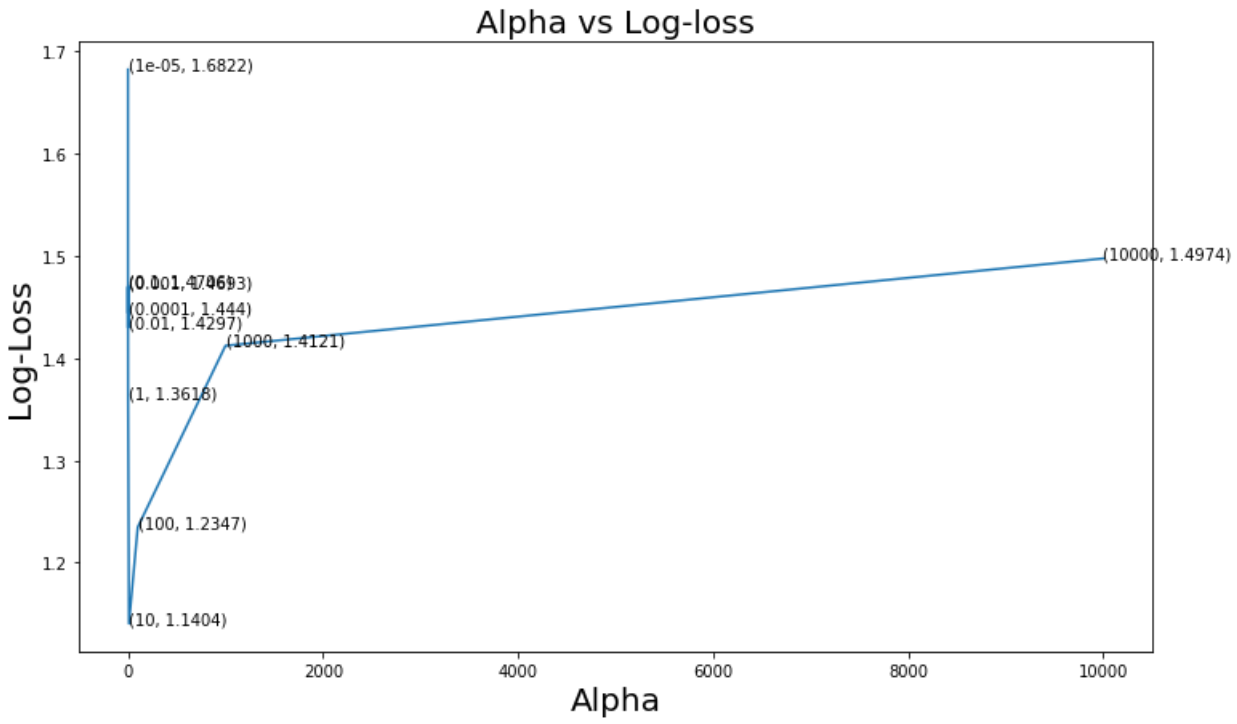
Logistic Regression without class balancing

Hyper-Parameter Tuning

In [154]:

```
warnings.simplefilter('ignore')
alpha = [10**x for x in range(-5, 5)]

cross_val_lgloss = []
for i in alpha:
    clf = SGDClassifier(loss = "log", alpha = i)
    clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    predicted_y = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
    cross_val_lgloss.append(log_loss(gene_final_crossval_Y, predicted_y, labels=clf.classes_))
    print("For alpha value of "+str(i)+" CV log loss = "+str(log_loss(gene_final_crossval_Y, predic
ted_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```

```
For alpha value of 1e-05 CV log loss = 1.6822267421428247
For alpha value of 0.0001 CV log loss = 1.4439792925818673
For alpha value of 0.001 CV log loss = 1.4693254245846028
For alpha value of 0.01 CV log loss = 1.42972062512733
For alpha value of 0.1 CV log loss = 1.4705901498451686
For alpha value of 1 CV log loss = 1.3617882483998252
For alpha value of 10 CV log loss = 1.1403742963873613
For alpha value of 100 CV log loss = 1.2347395370034169
For alpha value of 1000 CV log loss = 1.4121348341789688
For alpha value of 10000 CV log loss = 1.497412269848006
```



Testing with best hyper-parameter

In [155]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]
clf = SGDClassifier(loss = "log", alpha = best_alpha)
clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)

train_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TRAIN)
print('For values of best alpha = ', best_alpha, "the train log loss
=:",log_loss(gene_final_Y_TRAIN, train_final_pred, labels=clf.classes_))

cross_val_final_pred = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
print('For values of best alpha = ', best_alpha, "the CV log loss
=:",log_loss(gene_final_crossval_Y, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TEST)
print('For values of best alpha = ', best_alpha, "the test log loss =:",log_loss(gene_final_Y_TEST
, test_final_pred, labels=clf.classes_))
```

```
For values of best alpha =  10 the train log loss =: 0.6640724039427359
For values of best alpha =  10 the CV log loss =: 1.1335247800136392
For values of best alpha =  10 the test log loss =: 1.0956791338522687
```

In [156]:

```
print("Percentage of mis-classified for CV points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_crossval_X) -
gene_final_crossval_Y)/gene_final_crossval_X.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_X_TEST) -
gene_final_Y_TEST)/gene_final_X_TEST.shape[0]*100), 2))+"%")
```

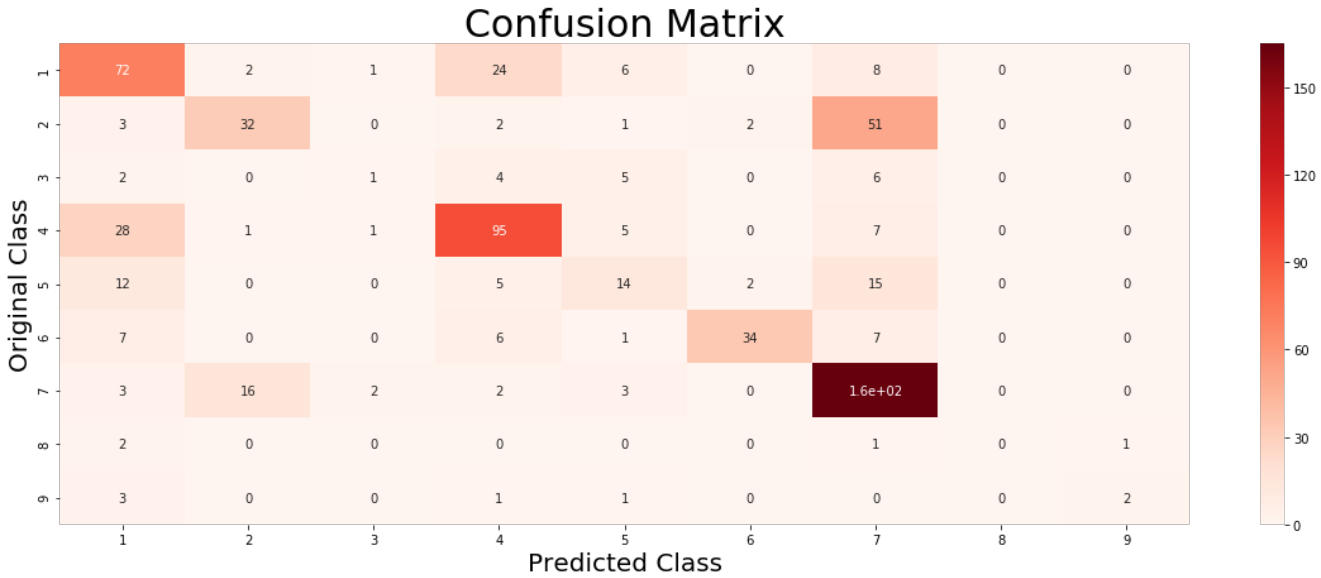
```
Percentage of mis-classified for CV points = 39.36%
Percentage of mis-classified for Test points = 37.5%
```

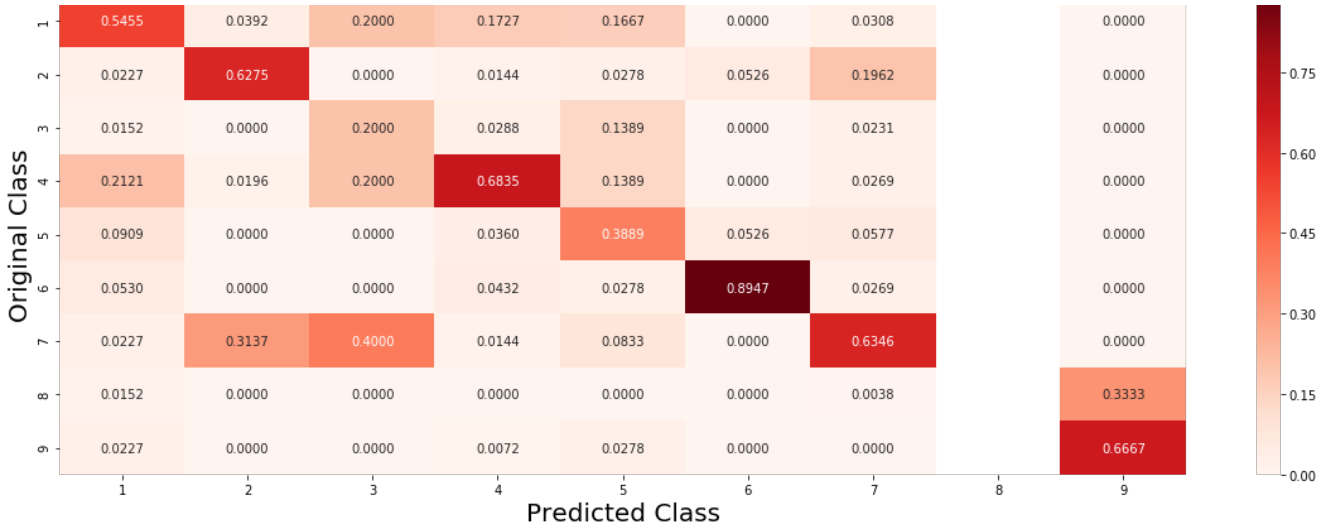
In [49]:

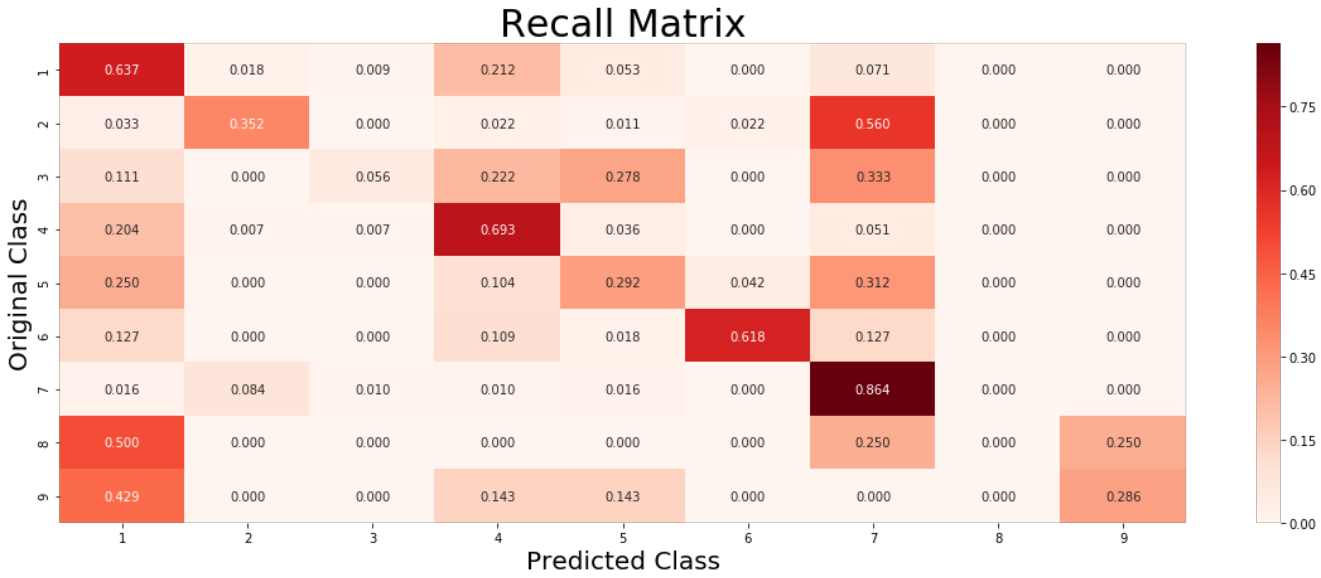
```
table = table.append(pd.DataFrame([["Logistic Regresion(Imbalanced)", 0.6640, 1.1335, 1.0956, "39.3
6%", "37.5%", "Good Fit"]], columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "
Mis-Classified CV", "Mis-Classified Test", "Remarks"]))
```

In [158]:

```
generate_conf_mat(gene_final_Y_TEST, classifier_generate_claibrated.predict(gene_final_X_TEST))
```







Checking first 500 important features for correctly classified test point

In [160]:

```
dataset_test_check = 5
no_feature = 500
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 3)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 4
Predicted Probabilities for test point = [[0.028 0.014 0.002 0.922 0.01  0.007 0.014 0.002 0.   ]]
True class label for test point = 4
---------------------------------------------------------------
1st Text feature [recently] is present in query point
3rd Text feature [potential] is present in query point
7th Text feature [activation] is present in query point
8th Text feature [independent] is present in query point
9th Text feature [addition] is present in query point
```

```
13th Text feature [compared] is present in query point
14th Text feature [currently] is present in query point
15th Text feature [kinase] is present in query point
17th Text feature [suggests] is present in query point
20th Text feature [concentrations] is present in query point
25th Text feature [screening] is present in query point
30th Text feature [activating] is present in query point
36th Text feature [identification] is present in query point
41st Text feature [using] is present in query point
42nd Text feature [including] is present in query point
45th Text feature [factor] is present in query point
55th Text feature [presence] is present in query point
60th Text feature [reported] is present in query point
65th Text feature [therapeutic] is present in query point
66th Text feature [small] is present in query point
67th Text feature [fusion] is present in query point
78th Text feature [novel] is present in query point
83rd Text feature [inhibitor] is present in query point
88th Text feature [additional] is present in query point
89th Text feature [fish] is present in query point
99th Text feature [10] is present in query point
105th Text feature [22] is present in query point
122nd Text feature [mechanism] is present in query point
123rd Text feature [17] is present in query point
128th Text feature [different] is present in query point
132nd Text feature [phosphorylation] is present in query point
133rd Text feature [shown] is present in query point
137th Text feature [transcript] is present in query point
148th Text feature [13] is present in query point
154th Text feature [obtained] is present in query point
166th Text feature [identified] is present in query point
168th Text feature [confirmed] is present in query point
171st Text feature [conformation] is present in query point
179th Text feature [report] is present in query point
192nd Text feature [study] is present in query point
202nd Text feature [sequences] is present in query point
206th Text feature [highest] is present in query point
208th Text feature [respectively] is present in query point
210th Text feature [clinical] is present in query point
213rd Text feature [does] is present in query point
223rd Text feature [12] is present in query point
230th Text feature [domain] is present in query point
233rd Text feature [approved] is present in query point
236th Text feature [cells] is present in query point
256th Text feature [did] is present in query point
262nd Text feature [located] is present in query point
266th Text feature [14] is present in query point
272nd Text feature [partners] is present in query point
276th Text feature [mainly] is present in query point
282nd Text feature [1a] is present in query point
288th Text feature [previously] is present in query point
289th Text feature [studies] is present in query point
291st Text feature [increase] is present in query point
299th Text feature [previous] is present in query point
312nd Text feature [hum] is present in query point
316th Text feature [showed] is present in query point
322nd Text feature [interestingly] is present in query point
325th Text feature [15] is present in query point
333rd Text feature [treated] is present in query point
334th Text feature [identify] is present in query point
373rd Text feature [gene] is present in query point
379th Text feature [fused] is present in query point
388th Text feature [selective] is present in query point
390th Text feature [modest] is present in query point
394th Text feature [discussion] is present in query point
397th Text feature [intracellular] is present in query point
407th Text feature [growth] is present in query point
411st Text feature [mutation] is present in query point
419th Text feature [confirm] is present in query point
422nd Text feature [common] is present in query point
423rd Text feature [taken] is present in query point
424th Text feature [inhibitors] is present in query point
426th Text feature [noted] is present in query point
427th Text feature [include] is present in query point
435th Text feature [non] is present in query point
439th Text feature [confer] is present in query point
452nd Text feature [total] is present in query point
```

```
464th Text feature [tumors] is present in query point
472nd Text feature [factors] is present in query point
475th Text feature [www] is present in query point
476th Text feature [observed] is present in query point
480th Text feature [time] is present in query point
485th Text feature [review] is present in query point
489th Text feature [inhibited] is present in query point
491st Text feature [results] is present in query point
498th Text feature [requires] is present in query point
-----------------------------------------------------------
Out of the top 500 features 91 are present in query point
```

Checking first 500 important features for incorrectly classified test point

In [161]:

```
dataset_test_check = 25
no_feature = 500
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 4)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 1
Predicted Probabilities for test point = [[0.5426 0.0067 0.0012 0.1688 0.2183 0.0566 0.0032 0.0025
0.    ]]
True class label for test point = 5
-----------------------------------------------------------
1st Text feature [kinase] is present in query point
3rd Text feature [contrast] is present in query point
8th Text feature [shown] is present in query point
16th Text feature [previously] is present in query point
20th Text feature [described] is present in query point
26th Text feature [activation] is present in query point
33rd Text feature [activity] is present in query point
39th Text feature [constitutive] is present in query point
43rd Text feature [mutations] is present in query point
45th Text feature [positive] is present in query point
47th Text feature [expressed] is present in query point
51st Text feature [cells] is present in query point
52nd Text feature [downstream] is present in query point
61st Text feature [phosphorylation] is present in query point
70th Text feature [12] is present in query point
73rd Text feature [showed] is present in query point
92nd Text feature [activating] is present in query point
101st Text feature [point] is present in query point
128th Text feature [similar] is present in query point
131st Text feature [10] is present in query point
135th Text feature [signaling] is present in query point
137th Text feature [discussion] is present in query point
146th Text feature [higher] is present in query point
147th Text feature [independent] is present in query point
149th Text feature [increased] is present in query point
160th Text feature [mutants] is present in query point
161st Text feature [presence] is present in query point
167th Text feature [addition] is present in query point
176th Text feature [fusion] is present in query point
177th Text feature [resulting] is present in query point
178th Text feature [domain] is present in query point
185th Text feature [stably] is present in query point
186th Text feature [characterized] is present in query point
192nd Text feature [mutant] is present in query point
211st Text feature [recently] is present in query point
213rd Text feature [potential] is present in query point
235th Text feature [different] is present in query point
239th Text feature [suggest] is present in query point
252nd Text feature [psi] is present in query point
```

```
266th Text feature [respectively] is present in query point
269th Text feature [factor] is present in query point
276th Text feature [differential] is present in query point
301st Text feature [clinical] is present in query point
308th Text feature [15] is present in query point
327th Text feature [standard] is present in query point
328th Text feature [right] is present in query point
333rd Text feature [mutation] is present in query point
337th Text feature [sensitive] is present in query point
343rd Text feature [immobilized] is present in query point
350th Text feature [levels] is present in query point
356th Text feature [using] is present in query point
368th Text feature [concentrations] is present in query point
382nd Text feature [acid] is present in query point
385th Text feature [40] is present in query point
427th Text feature [various] is present in query point
428th Text feature [observed] is present in query point
430th Text feature [use] is present in query point
434th Text feature [represent] is present in query point
450th Text feature [wt] is present in query point
------------------------------------------------------------
Out of the top 500 features 59 are present in query point
```

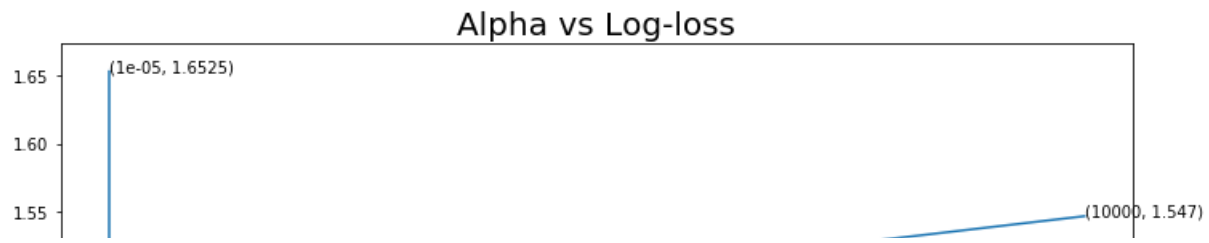
Linear SVM

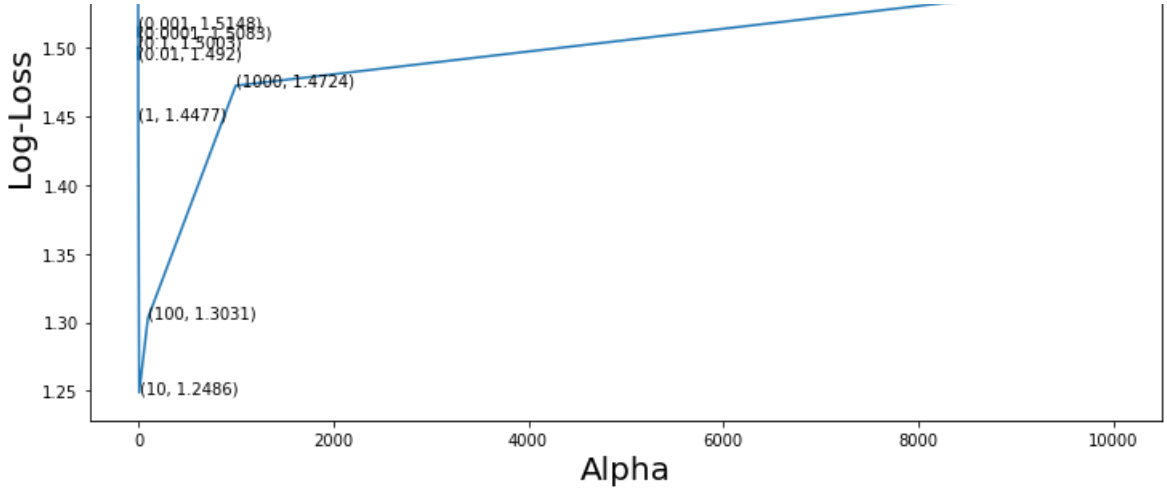
Hyper-Parameter Tuning

In [162]:

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

cross_val_lgloss = []
for i in alpha:
    clf = SGDClassifier(loss = "hinge", alpha = i, class_weight = "balanced")
    clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
    classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    predicted_y = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
    cross_val_lgloss.append(log_loss(gene_final_crossval_Y, predicted_y, labels=clf.classes_))
    print("For alpha value of "+str(i)+" CV log loss = "+str(log_loss(gene_final_crossval_Y, predic
ted_y, labels=clf.classes_)))

plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```

```
For alpha value of 1e-05 CV log loss = 1.6525311487762264
For alpha value of 0.0001 CV log loss = 1.508312724941579
For alpha value of 0.001 CV log loss = 1.5148386727259073
For alpha value of 0.01 CV log loss = 1.492017021066627
For alpha value of 0.1 CV log loss = 1.5003191875232478
For alpha value of 1 CV log loss = 1.4476661721062734
For alpha value of 10 CV log loss = 1.2486288231044749
For alpha value of 100 CV log loss = 1.3031323911685826
For alpha value of 1000 CV log loss = 1.4723578568504232
For alpha value of 10000 CV log loss = 1.547033276044576
```





Testing with best best Hyper-Parameter

In [163]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]
clf = SGDClassifier(loss = "hinge", alpha = best_alpha, class_weight = "balanced")
clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)

train_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TRAIN)
print('For values of best alpha = ', best_alpha, "the train log loss
=:",log_loss(gene_final_Y_TRAIN, train_final_pred, labels=clf.classes_))

cross_val_final_pred = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
print('For values of best alpha = ', best_alpha, "the CV log loss
=:",log_loss(gene_final_crossval_Y, cross_val_final_pred, labels=clf.classes_))

test_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TEST)
print('For values of best alpha = ', best_alpha, "the test log loss =:",log_loss(gene_final_Y_TEST
, test_final_pred, labels=clf.classes_))
```

```
For values of best alpha =  10 the train log loss =: 0.8022730656225745
For values of best alpha =  10 the CV log loss =: 1.2399335397940394
For values of best alpha =  10 the test log loss =: 1.221772926915284
```

In [164]:

```
print("Percentage of mis-classified for CV points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_crossval_X) -
gene_final_crossval_Y)/gene_final_crossval_X.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_X_TEST) -
gene_final_Y_TEST)/gene_final_X_TEST.shape[0]*100), 2))+"%")
```

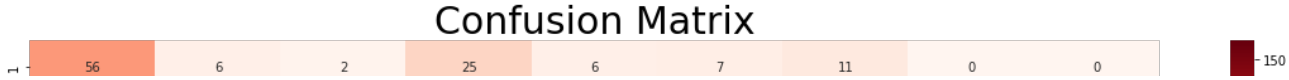
```
Percentage of mis-classified for CV points = 39.36%
Percentage of mis-classified for Test points = 38.7%
```

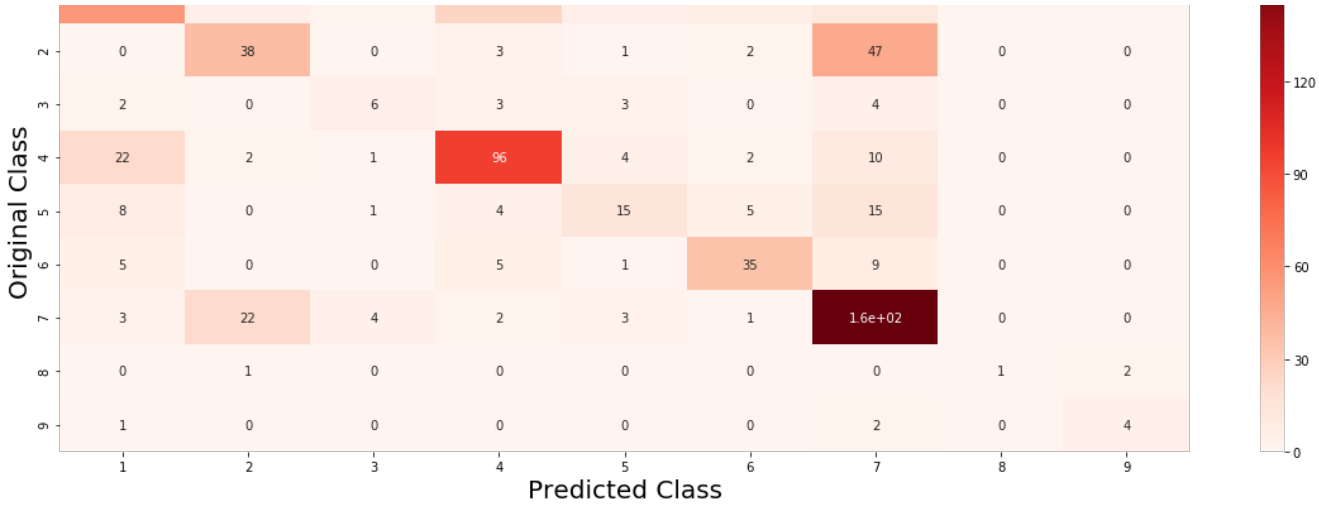
In [50]:

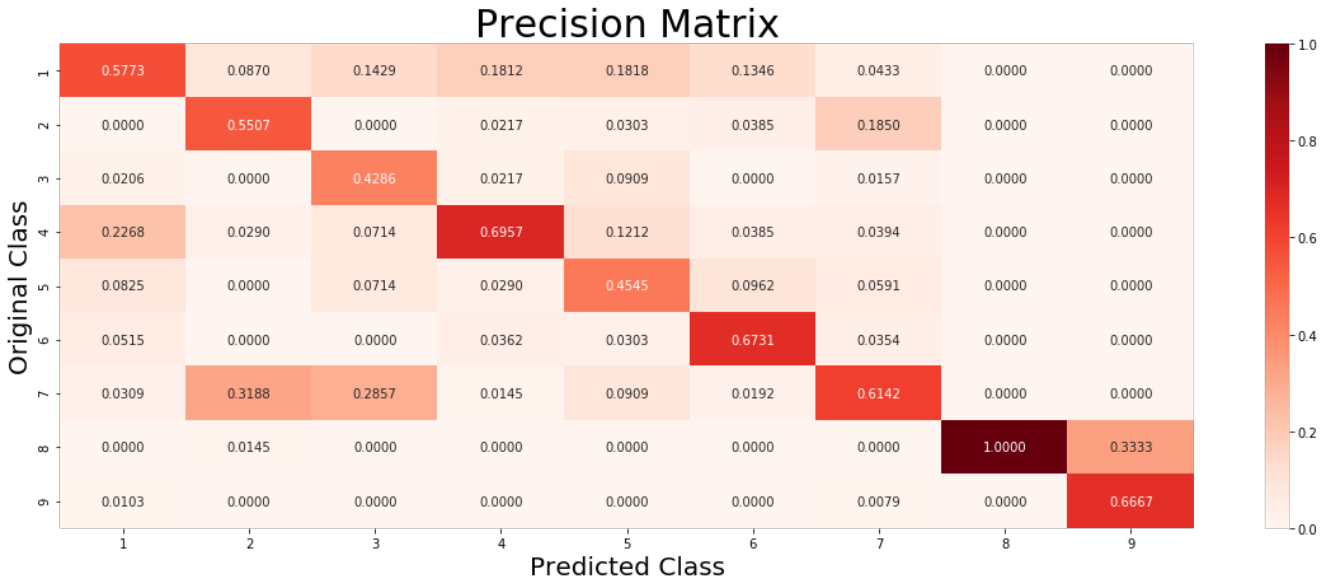
```
table = table.append(pd.DataFrame([["Linear SVM(Balanced)", 0.8022, 1.2399, 1.2217, "39.36%", "38.7
%", "Good Fit"]], columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis-Classi
fied CV", "Mis-Classified Test", "Remarks"]))
```

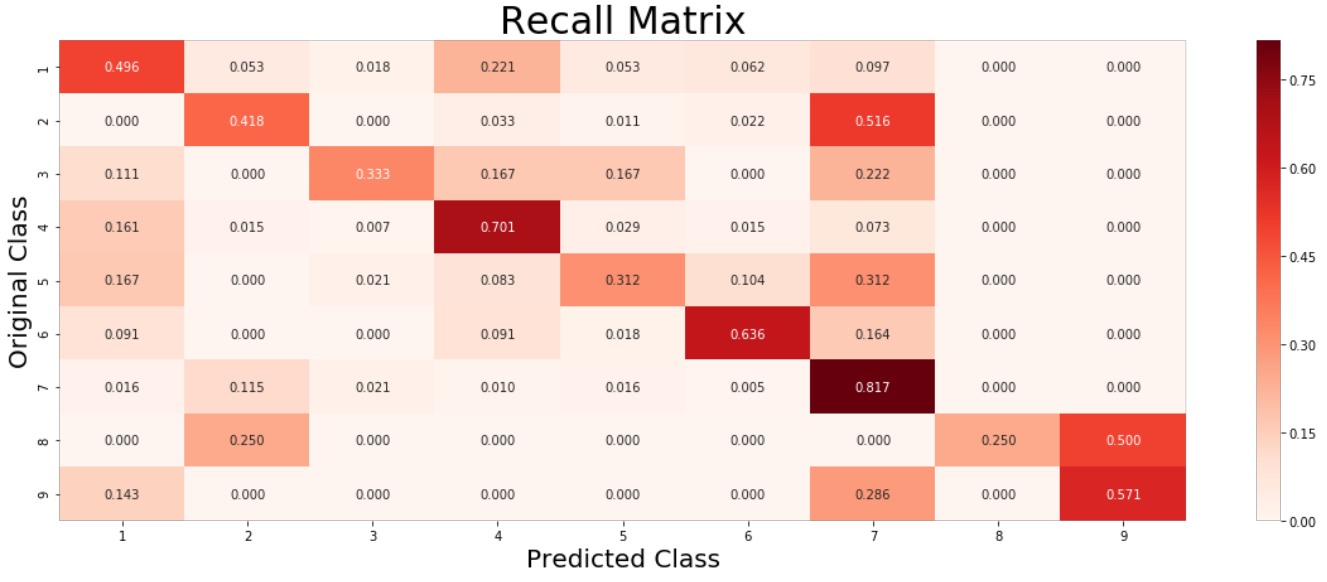
In [166]:

```
generate_conf_mat(gene_final_Y_TEST, classifier_generate_claibrated.predict(gene_final_X_TEST))
```









Checking first 500 important features for correctly classified test point

In [167]:

```
dataset_test_check = 5
no_feature = 500
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 4)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 4
Predicted Probabilities for test point = [[0.0467 0.062  0.0062 0.7801 0.0256 0.0236 0.0494 0.0042
0.0022]]
True class label for test point = 4
---------------------------------------------------------------
10th Text feature [recently] is present in query point
71st Text feature [suggests] is present in query point
78th Text feature [activation] is present in query point
86th Text feature [potential] is present in query point
141st Text feature [screening] is present in query point
180th Text feature [sequences] is present in query point
199th Text feature [concentrations] is present in query point
202nd Text feature [fused] is present in query point
222nd Text feature [addition] is present in query point
243rd Text feature [independent] is present in query point
291st Text feature [identification] is present in query point
297th Text feature [mainly] is present in query point
392nd Text feature [confirmed] is present in query point
453rd Text feature [24] is present in query point
456th Text feature [compared] is present in query point
462nd Text feature [fish] is present in query point
468th Text feature [mgcl] is present in query point
482nd Text feature [neck] is present in query point
497th Text feature [previous] is present in query point
---------------------------------------------------------------
Out of the top 500 features 19 are present in query point
```

Checking first 500 important features for incorrectly classified test point

In [168]:

```
dataset_test_check = 25
no_feature = 500
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 4)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-abs(clf.coef_))[predicted_cls-1][:,:no_feature]
print("-"*63)
find_feat_importance(indices[0], X_Test.iloc[dataset_test_check]["Gene"],
X_Test.iloc[dataset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 1
Predicted Probabilities for test point = [[0.3271 0.0315 0.0042 0.2252 0.3249 0.037  0.0352 0.0099
0.005 ]]
True class label for test point = 5
---------------------------------------------------------------
76th Text feature [contrast] is present in query point
112nd Text feature [shown] is present in query point
119th Text feature [kinase] is present in query point
222nd Text feature [psi] is present in query point
225th Text feature [previously] is present in query point
236th Text feature [explain] is present in query point
248th Text feature [activity] is present in query point
```

```
261st Text feature [iii] is present in query point
269th Text feature [right] is present in query point
275th Text feature [immobilized] is present in query point
312nd Text feature [positive] is present in query point
323rd Text feature [levels] is present in query point
350th Text feature [discussion] is present in query point
376th Text feature [fused] is present in query point
432nd Text feature [patch] is present in query point
434th Text feature [described] is present in query point
460th Text feature [characterized] is present in query point
500th Text feature [respectively] is present in query point
----------------------------------------------------------------------
Out of the top 500 features 18 are present in query point
```

Random Forest

Hyper-Parameter Tuning

In [88]:

```
base_learners = [100, 200, 500, 1000]
max_depth_baseLearners = [7, 12]
cross_val_lgloss = []

for i in base_learners:
    for j in max_depth_baseLearners:
        clf = RandomForestClassifier(n_estimators = i, max_depth = j, n_jobs = -1)
        clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
        classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
        classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
        predicted_y = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
        cross_val_lgloss.append(log_loss(gene_final_crossval_Y, predicted_y, labels=clf.classes_))
        print("For Number of base learners "+str(i)+" and max depth of a tree "+str(j)+" CV log los
s = "+str(log_loss(gene_final_crossval_Y, predicted_y, labels=clf.classes_)))
```

```
For Number of base learners 100 and max depth of a tree 7 CV log loss = 1.154436650893394
For Number of base learners 100 and max depth of a tree 12 CV log loss = 1.1000600923075554
For Number of base learners 200 and max depth of a tree 7 CV log loss = 1.155345537255194
For Number of base learners 200 and max depth of a tree 12 CV log loss = 1.1026486586391353
For Number of base learners 500 and max depth of a tree 7 CV log loss = 1.151956419270043
For Number of base learners 500 and max depth of a tree 12 CV log loss = 1.0912942995856048
For Number of base learners 1000 and max depth of a tree 7 CV log loss = 1.1442461540133317
For Number of base learners 1000 and max depth of a tree 12 CV log loss = 1.089753734225643
```

In [93]:

```
gridLogLoss = []
subLogLoss = []
x = [x for x in range(1, 8, 2)]
for i in range(8):
    subLogLoss.append(np.round(cross_val_lgloss[i], 4))
    if i in x:
        gridLogLoss.append(subLogLoss)
        subLogLoss = []
```

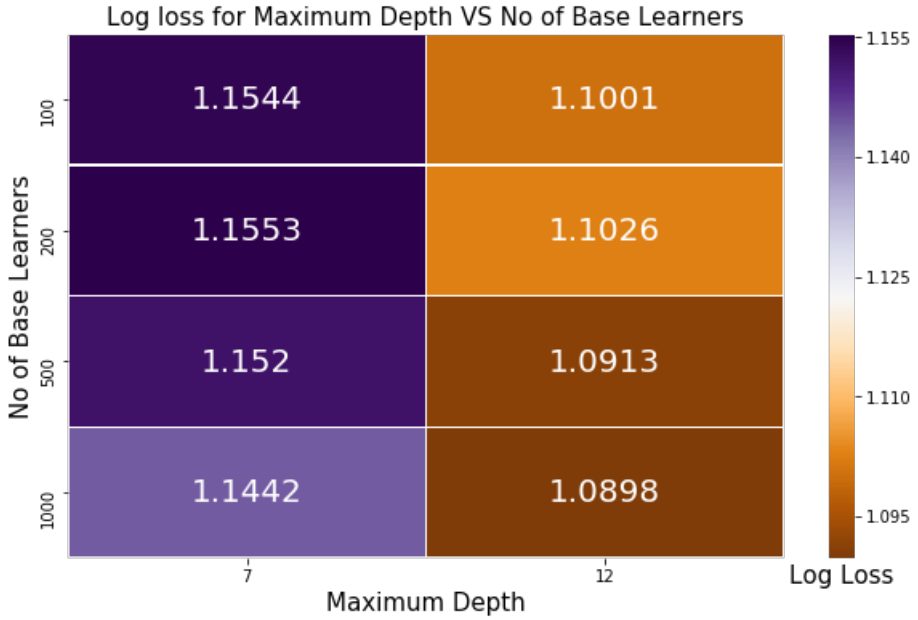
In [94]:

```
gridLogLossFrame = pd.DataFrame(gridLogLoss, columns = max_depth_baseLearners)
gridLogLossFrame["Base_Learners"] = base_learners
gridLogLossFrame.set_index("Base_Learners", append = False, drop = True, inplace = True)
```

In [95]:

```
plt.figure(figsize = (10,6))
plt.title("Log loss for Maximum Depth VS No of Base Learners", size = 15)
ax = sns.heatmap(gridLogLossFrame, annot = True, cmap="PuOr", linewidths = 0.5, fmt = ".5g", annot_
kws={"size": 20})
ax.figure.axes[0].set_xlabel("Maximum Depth", size = 15)
ax.figure.axes[0].set_ylabel("No of Base Learners", size = 15)
ax.figure.axes[-1].set_xlabel("Log Loss", size = 15)
```

```
plt.show()
```



Testing with best Hyper Parameter

In [97]:

```
best = np.argmin(cross_val_lgloss)
best_estimator = base_learners[int(best/2)]
best_depth = max_depth_baseLearners[int(best/4)]
clf = RandomForestClassifier(n_estimators = best_estimator, max_depth = best_depth, n_jobs = -1)
clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated = CalibratedClassifierCV(clf, method = "sigmoid")
classifier_generate_claibrated.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)

train_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TRAIN)
print("For Number of base learners "+str(best_estimator)+" and max depth of a tree
"+str(best_depth)+" Train log loss = "+str(log_loss(gene_final_Y_TRAIN, train_final_pred, labels=cl
f.classes_)))

cross_val_final_pred = classifier_generate_claibrated.predict_proba(gene_final_crossval_X)
print("For Number of base learners "+str(best_estimator)+" and max depth of a tree
"+str(best_depth)+" CV log loss = "+str(log_loss(gene_final_crossval_Y, cross_val_final_pred, label
s=clf.classes_)))

test_final_pred = classifier_generate_claibrated.predict_proba(gene_final_X_TEST)
print("For Number of base learners "+str(best_estimator)+" and max depth of a tree
"+str(best_depth)+" Test log loss = "+str(log_loss(gene_final_Y_TEST, test_final_pred, labels=clf.c
lasses_)))
```

```
For Number of base learners 1000 and max depth of a tree 12 Train log loss = 0.6165995922410343
For Number of base learners 1000 and max depth of a tree 12 CV log loss = 1.0961313160427155
For Number of base learners 1000 and max depth of a tree 12 Test log loss = 1.0994898571114247
```

In [98]:

```
print("Percentage of mis-classified for CV points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_crossval_X) -
gene_final_crossval_Y)/gene_final_crossval_X.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points =
"+str(np.round((np.count_nonzero(classifier_generate_claibrated.predict(gene_final_X_TEST) -
gene_final_Y_TEST)/gene_final_X_TEST.shape[0]*100), 2))+"%")
```

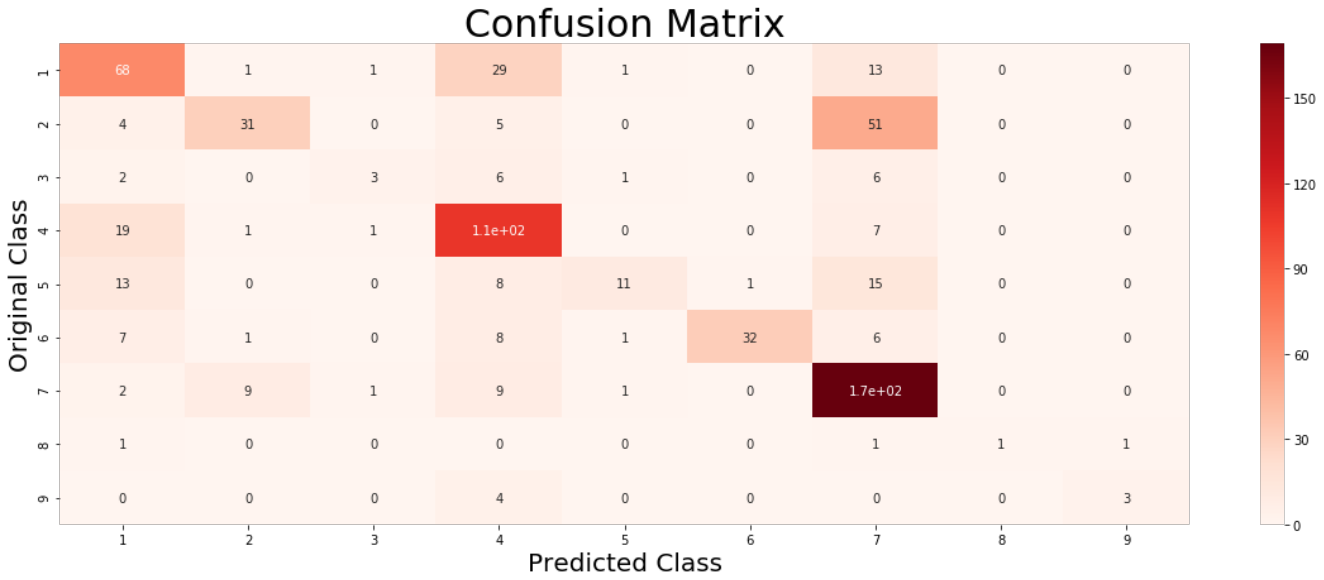
```
Percentage of mis-classified for CV points = 38.23%
Percentage of mis-classified for Test points = 35.69%
```

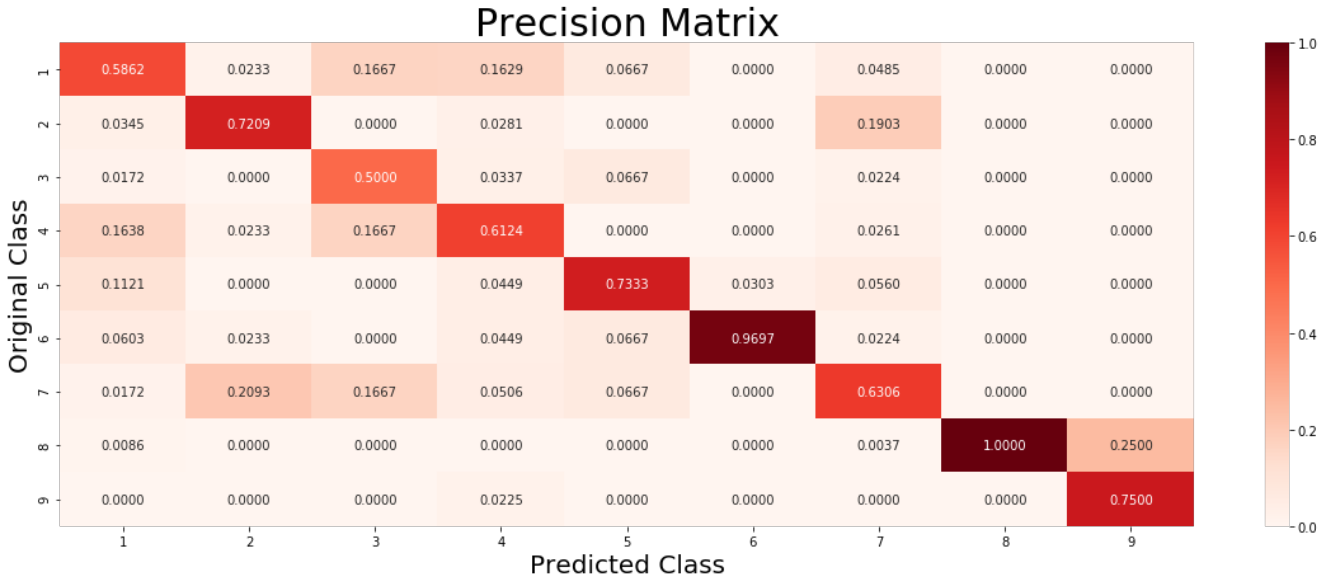
In [51]:

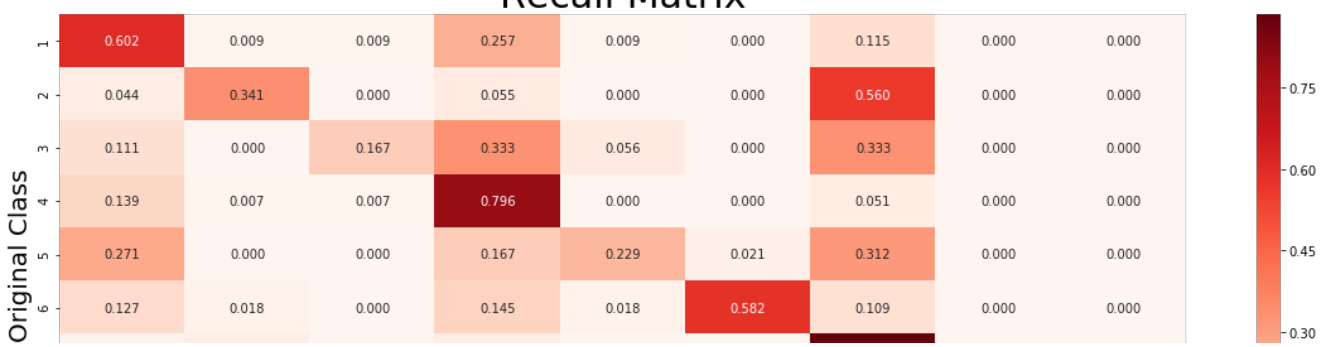
```
table = table.append(pd.DataFrame([["Random Forest", 0.6165, 1.0961, 1.0994, "38.23%", "35.69%", "Best Fit"]], columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis-Classified CV", "Mis-Classified Test", "Remarks"]))
```

In [99]:

```
generate_conf_mat(gene_final_Y_TEST, classifier_generate_claibrated.predict(gene_final_X_TEST))
```







Checking first 100 important features for correctly classified test point

In [116]:

```
dataset_test_check = 5
no_feature = 100
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 4)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-clf.feature_importances_)
print("-"*63)
find_feat_importance(indices[:no_feature], X_Test.iloc[dataset_test_check]["Gene"], X_Test.iloc[da
taset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 4
Predicted Probabilities for test point = [[0.0687 0.0322 0.0129 0.779  0.0342 0.0307 0.0345 0.0037
0.0042]]
True class label for test point = 4
---------------------------------------------------------------
1st Text feature [kinase] is present in query point
3rd Text feature [activated] is present in query point
4th Text feature [activation] is present in query point
6th Text feature [activating] is present in query point
7th Text feature [inhibitors] is present in query point
10th Text feature [phosphorylation] is present in query point
11th Text feature [inhibitor] is present in query point
12th Text feature [suppressor] is present in query point
16th Text feature [function] is present in query point
17th Text feature [nonsense] is present in query point
24th Text feature [growth] is present in query point
25th Text feature [pathogenic] is present in query point
26th Text feature [oncogenic] is present in query point
28th Text feature [therapeutic] is present in query point
29th Text feature [treated] is present in query point
33rd Text feature [cells] is present in query point
37th Text feature [inhibition] is present in query point
39th Text feature [protein] is present in query point
41st Text feature [deleterious] is present in query point
44th Text feature [missense] is present in query point
53rd Text feature [kinases] is present in query point
54th Text feature [variants] is present in query point
56th Text feature [proliferation] is present in query point
58th Text feature [functional] is present in query point
60th Text feature [inhibited] is present in query point
61st Text feature [proteins] is present in query point
64th Text feature [patients] is present in query point
68th Text feature [cell] is present in query point
69th Text feature [loss] is present in query point
71st Text feature [stability] is present in query point
76th Text feature [serum] is present in query point
83rd Text feature [clinical] is present in query point
84th Text feature [carriers] is present in query point
88th Text feature [expressing] is present in query point
98th Text feature [active] is present in query point
---------------------------------------------------------------
Out of the top 100 features 35 are present in query point
```

Checking first 100 important features for incorrectly classified test point

In [115]:

```
dataset_test_check = 25
no_feature = 100
predicted_cls = classifier_generate_claibrated.predict(gene_final_X_TEST[dataset_test_check])
correct_lable_new = gene_final_Y_TEST[dataset_test_check]
final_pred_new_probs = np.round(classifier_generate_claibrated.predict_proba(gene_final_X_TEST[dat
aset_test_check]), 4)
print("Predicted Class label for test point = "+str(predicted_cls[0]))
print("Predicted Probabilities for test point = "+str(final_pred_new_probs))
print("True class label for test point = "+str(correct_lable_new))
indices = np.argsort(-clf.feature_importances_)
print("-"*63)
find_feat_importance(indices[:no_feature], X_Test.iloc[dataset_test_check]["Gene"], X_Test.iloc[da
taset_test_check]["Variation"], X_Test.iloc[dataset_test_check]["Text"], no_feature)
```

```
Predicted Class label for test point = 1
Predicted Probabilities for test point = [[0.6004 0.0173 0.0106 0.045  0.2803 0.0293 0.0116 0.0025
0.0029]]
True class label for test point = 5
---------------------------------------------------------------
1st Text feature [kinase] is present in query point
4th Text feature [activation] is present in query point
6th Text feature [activating] is present in query point
10th Text feature [phosphorylation] is present in query point
12th Text feature [suppressor] is present in query point
16th Text feature [function] is present in query point
19th Text feature [signaling] is present in query point
20th Text feature [constitutive] is present in query point
23rd Text feature [downstream] is present in query point
25th Text feature [pathogenic] is present in query point
27th Text feature [activate] is present in query point
33rd Text feature [cells] is present in query point
39th Text feature [protein] is present in query point
41st Text feature [deleterious] is present in query point
44th Text feature [missense] is present in query point
49th Text feature [ligand] is present in query point
54th Text feature [variants] is present in query point
57th Text feature [response] is present in query point
58th Text feature [functional] is present in query point
61st Text feature [proteins] is present in query point
62nd Text feature [repair] is present in query point
64th Text feature [patients] is present in query point
65th Text feature [yeast] is present in query point
66th Text feature [unstable] is present in query point
68th Text feature [cell] is present in query point
69th Text feature [loss] is present in query point
71st Text feature [stability] is present in query point
82nd Text feature [resistant] is present in query point
83rd Text feature [clinical] is present in query point
---------------------------------------------------------------
Out of the top 100 features 29 are present in query point
```

Stacking the Models

In [175]:

```
clf_NB = MultinomialNB(alpha=10**-5)
clf_NB.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated_NB = CalibratedClassifierCV(clf_NB, method = "sigmoid")
classifier_generate_claibrated_NB.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
print("Naive Bayes CV Log Loss: "+str(np.round(log_loss(gene_final_crossval_Y,
classifier_generate_claibrated_NB.predict_proba(gene_final_crossval_X), labels=clf_NB.classes_), 4
)))

clf_LR = SGDClassifier(loss = "log", alpha = 10, class_weight = "balanced")
clf_LR.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated_LR = CalibratedClassifierCV(clf_LR, method = "sigmoid")
classifier_generate_claibrated_LR.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
print("Logistic Regression CV Log Loss: "+str(np.round(log_loss(gene_final_crossval_Y,
classifier_generate_claibrated_LR.predict_proba(gene_final_crossval_X), labels=clf_LR.classes_), 4
)))

clf_SVM = SGDClassifier(loss = "hinge", alpha = 10, class_weight = "balanced")
```

```
clf_SVM = SGDClassifier(loss = "hinge", alpha = 10, class_weight = "balanced")
clf_SVM.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated_SVM = CalibratedClassifierCV(clf_SVM, method = "sigmoid")
classifier_generate_claibrated_SVM.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
print("SVM CV Log Loss: "+str(np.round(log_loss(gene_final_crossval_Y,
classifier_generate_claibrated_SVM.predict_proba(gene_final_crossval_X), labels=clf_SVM.classes_),
4)))

clf_RF = RandomForestClassifier(n_estimators = 1000, max_depth = 12, n_jobs = -1)
clf_RF.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
classifier_generate_claibrated_RF = CalibratedClassifierCV(clf_RF, method = "sigmoid")
classifier_generate_claibrated_RF.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
print("Random Forest CV Log Loss: "+str(np.round(log_loss(gene_final_crossval_Y,
classifier_generate_claibrated_RF.predict_proba(gene_final_crossval_X), labels=clf_RF.classes_), 4
)))
```

```
Naive Bayes CV Log Loss: 1.3472
Logistic Regression CV Log Loss: 1.1333
SVM CV Log Loss: 1.2505
Random Forest CV Log Loss: 1.093
```

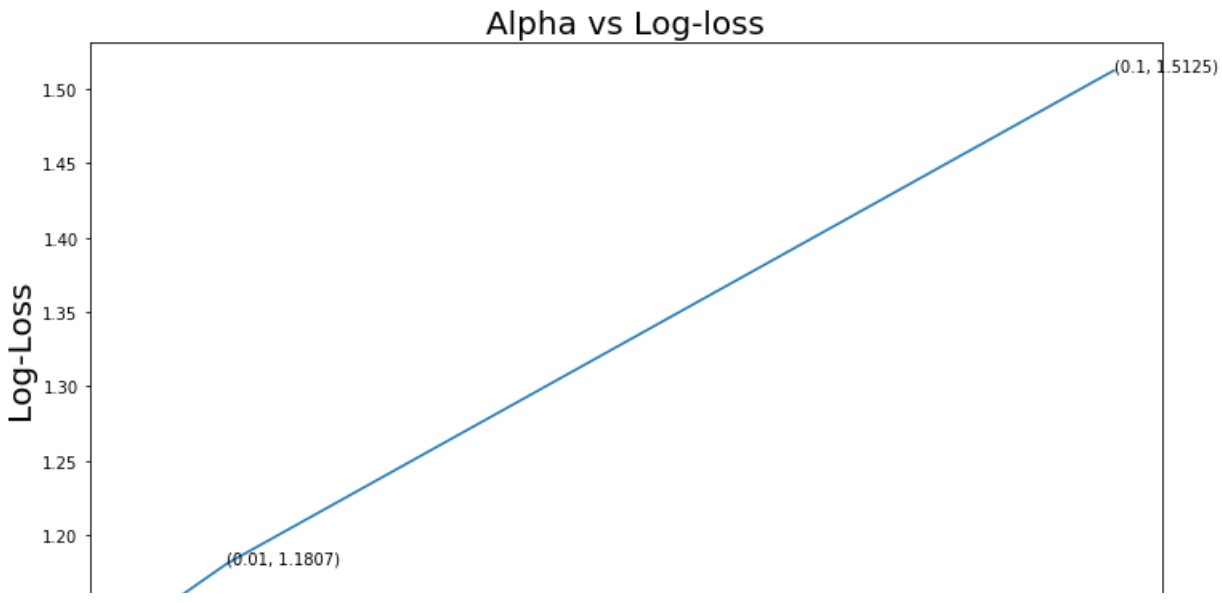
In [181]:

```
cross_val_lgloss = []
alpha = [10**x for x in range(-3, 0)]
for i in alpha:
    lr = SGDClassifier(loss = "log", alpha = i)
    stack_clf = StackingClassifier(classifiers=[classifier_generate_claibrated_NB, classifier_gener
ate_claibrated_LR, classifier_generate_claibrated_SVM, classifier_generate_claibrated_RF],
meta_classifier=lr, use_probas=True)
    stack_clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
    cross_val_lgloss.append(log_loss(gene_final_crossval_Y, stack_clf.predict_proba(gene_final_cros
sval_X)))
    print("Stacking Classifer : For alpha value: "+str(i)+" Log Loss: "+str(np.round(log_loss(gene_
final_crossval_Y, stack_clf.predict_proba(gene_final_crossval_X)), 4)))
```

```
Stacking Classifer : For alpha value: 0.001 Log Loss: 1.139
Stacking Classifer : For alpha value: 0.01 Log Loss: 1.1807
Stacking Classifer : For alpha value: 0.1 Log Loss: 1.5125
```

In [182]:

```
plt.figure(figsize = (12, 7))
plt.plot(alpha, cross_val_lgloss)
for xy in zip(alpha, np.round(cross_val_lgloss, 4)):
    plt.annotate(xy, xy)

plt.title("Alpha vs Log-loss", fontsize = 20)
plt.xlabel("Alpha", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```



In [184]:

```
best_alpha = alpha[np.argmin(cross_val_lgloss)]

lr = SGDClassifier(loss = "log", alpha = best_alpha)
stack_clf = StackingClassifier(classifiers=[classifier_generate_claibrated_NB,
classifier_generate_claibrated_LR, classifier_generate_claibrated_SVM,
classifier_generate_claibrated_RF], meta_classifier=lr, use_probas=True)
stack_clf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)

trainLogLoss = log_loss(gene_final_Y_TRAIN, stack_clf.predict_proba(gene_final_X_TRAIN))
print("Train Log Loss on Stacking Classifier =  "+str(np.round(trainLogLoss, 4)))

cvLogLoss = log_loss(gene_final_crossval_Y, stack_clf.predict_proba(gene_final_crossval_X))
print("Cross Validation Log Loss on Stacking Classifier =  "+str(np.round(cvLogLoss, 4)))

testLogLoss = log_loss(gene_final_Y_TEST, stack_clf.predict_proba(gene_final_X_TEST))
print("Test Log Loss on Stacking Classifier =  "+str(np.round(testLogLoss, 4)))

print("Percentage of mis-classified for CV points = "+str(np.round((np.count_nonzero(stack_clf.pre
dict(gene_final_crossval_X) - gene_final_crossval_Y)/gene_final_crossval_X.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points = "+str(np.round((np.count_nonzero(stack_clf.p
redict(gene_final_X_TEST) - gene_final_Y_TEST)/gene_final_X_TEST.shape[0]*100), 2))+"%")
```

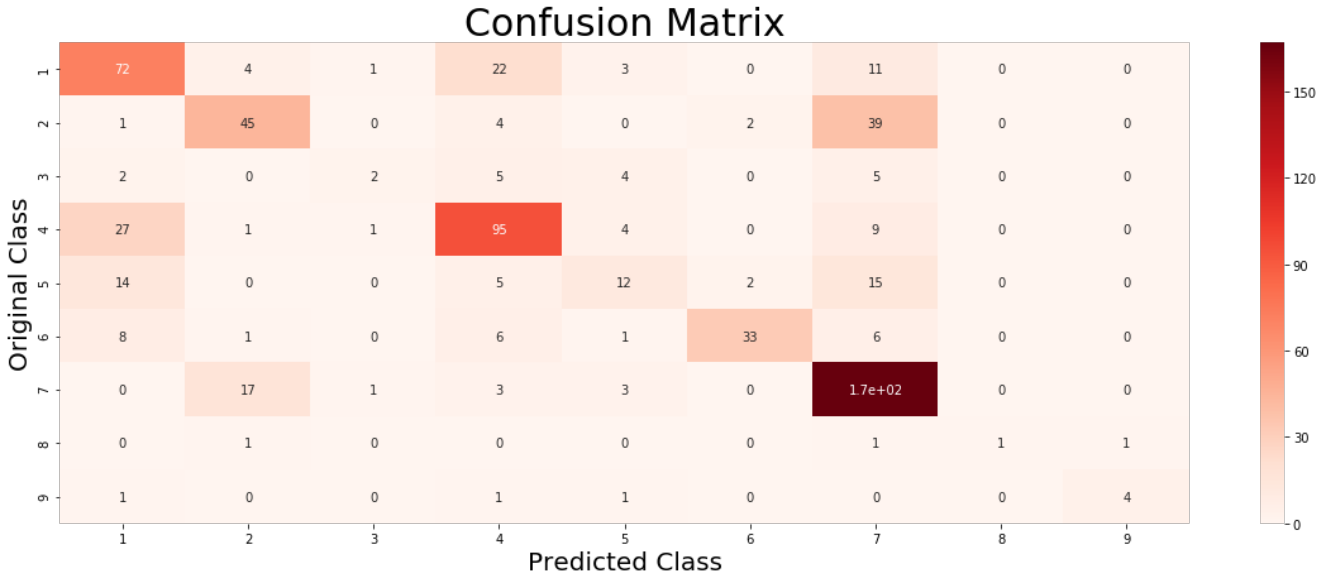
```
Train Log Loss on Stacking Classifier =  0.378
Cross Validation Log Loss on Stacking Classifier =  1.1311
Test Log Loss on Stacking Classifier =  1.0578
Percentage of mis-classified for CV points = 38.98%
Percentage of mis-classified for Test points = 35.09%
```

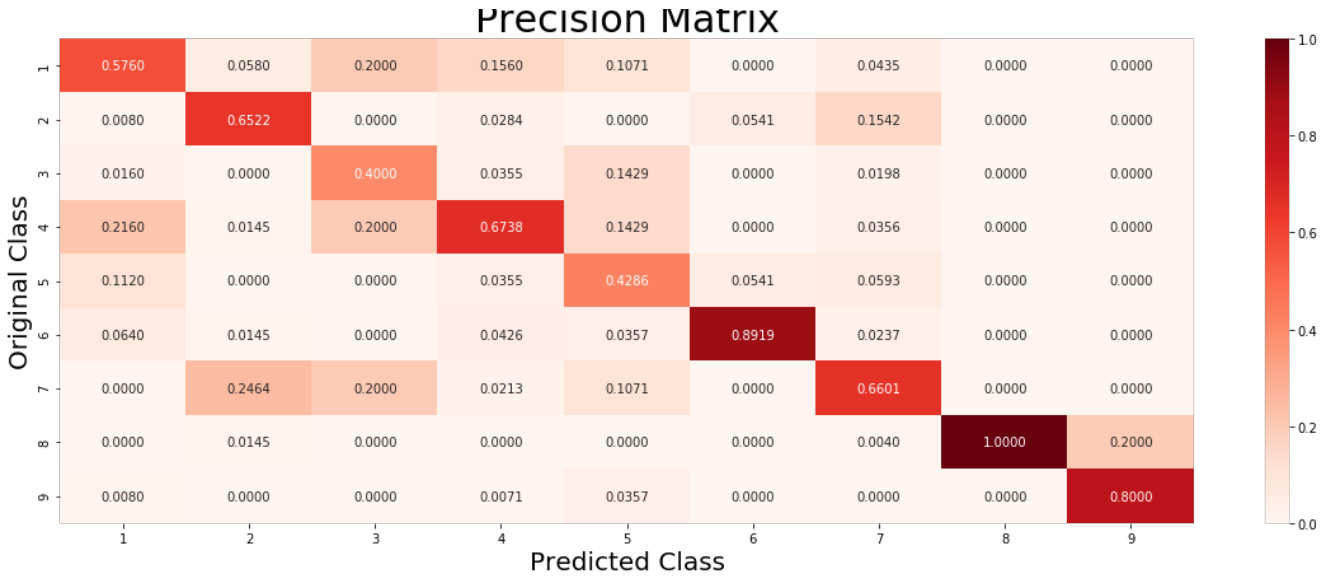
In [52]:

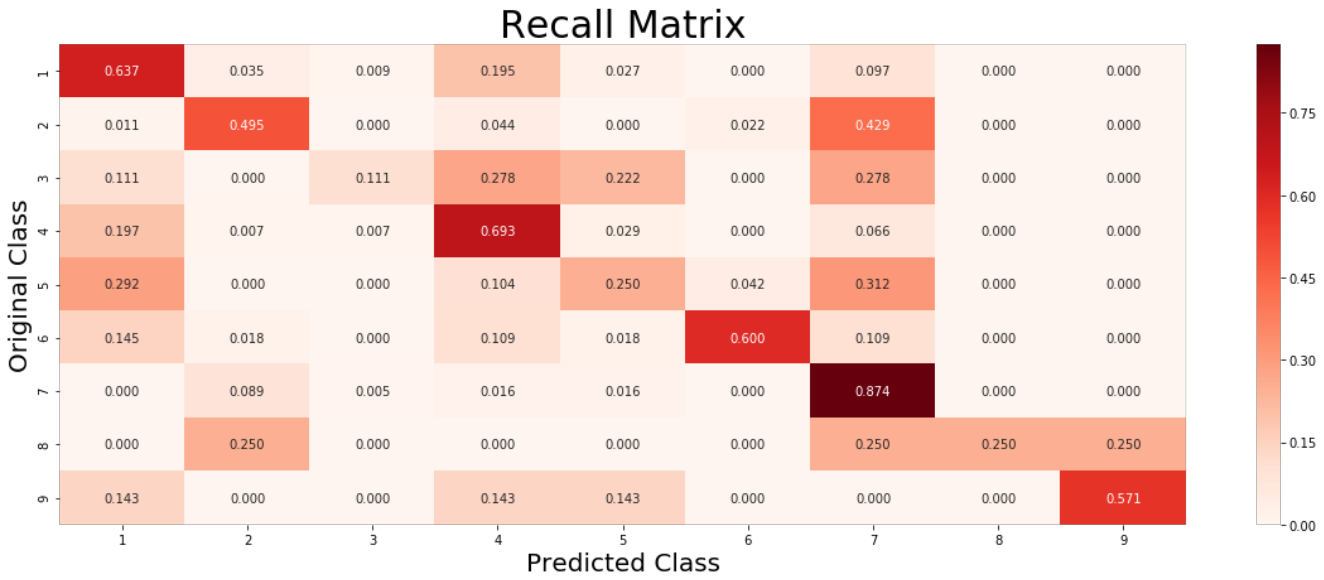
```
table = table.append(pd.DataFrame([["Stacking Classifier", 0.378, 1.1311, 1.0578, "38.98%", "35.09%
", "Best Fit"]], columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis-Classif
ied CV", "Mis-Classified Test", "Remarks"]))
```

In [186]:

```
generate_conf_mat(gene_final_Y_TEST, stack_clf.predict(gene_final_X_TEST))
```







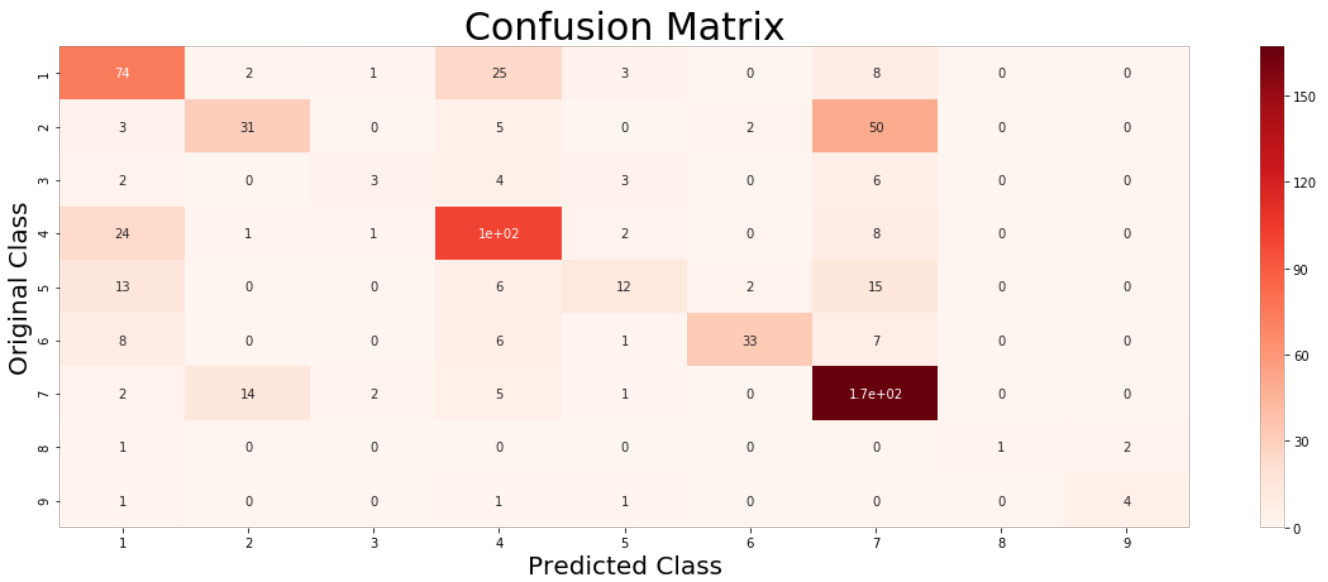
Maximum Voting Classifier: Logistic Regression, Linear SVM, Random Forest

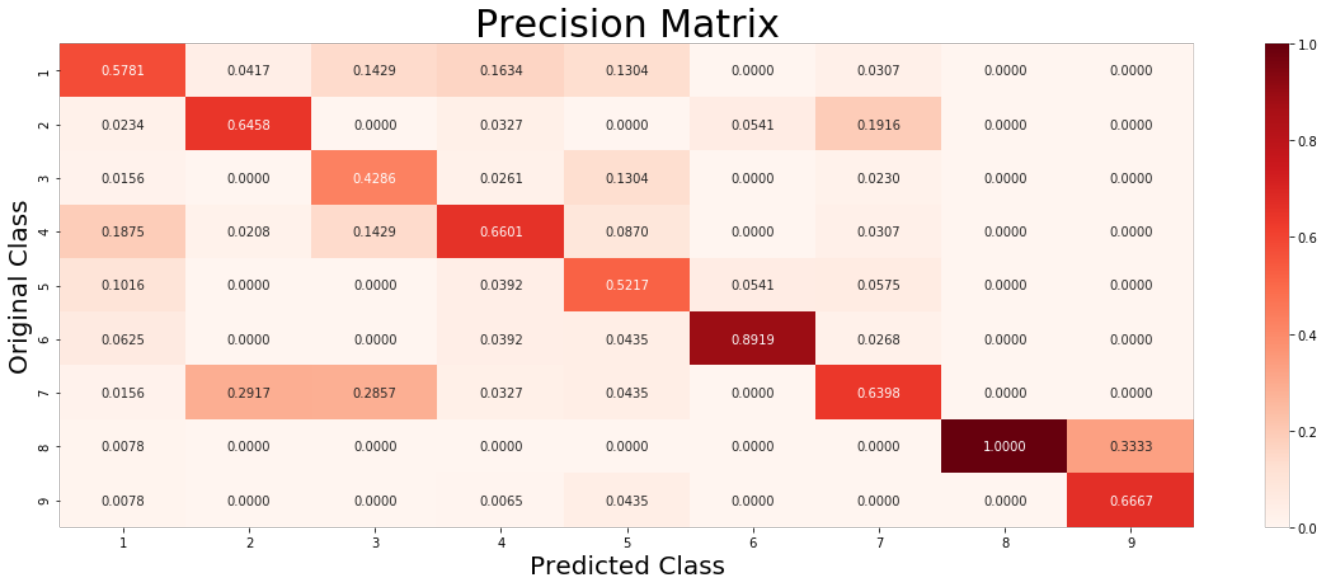
In [191]:

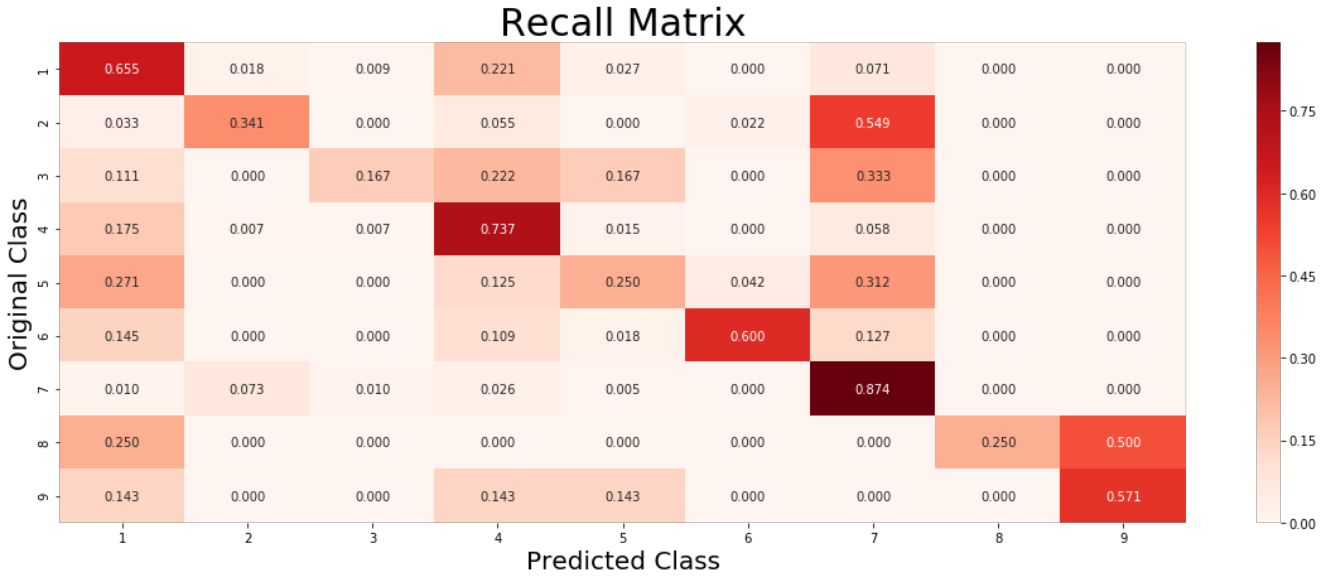
```
vclf = VotingClassifier(estimators=[('lr', classifier_generate_claibrated_LR), ('svc',
classifier_generate_claibrated_SVM), ('rf', classifier_generate_claibrated_RF)], voting='soft',
n_jobs = -1)
vclf.fit(gene_final_X_TRAIN, gene_final_Y_TRAIN)
print("Log loss (train) on the VotingClassifier :"+str(np.round(log_loss(gene_final_Y_TRAIN, vclf.
predict_proba(gene_final_X_TRAIN), labels=vclf.classes_), 2)))
print("Log loss (CV) on the VotingClassifier :"+str(np.round(log_loss(gene_final_crossval_Y, vclf.
predict_proba(gene_final_crossval_X), labels=vclf.classes_), 2)))
print("Log loss (test) on the VotingClassifier :"+str(np.round(log_loss(gene_final_Y_TEST, vclf.pr
edict_proba(gene_final_X_TEST), labels=vclf.classes_), 2)))
print("Percentage of mis-classified for cv points :"+str(np.round((np.count_nonzero(vclf.predict(g
ene_final_crossval_X) - gene_final_crossval_Y)/gene_final_crossval_X.shape[0]*100), 2))+"%")
print("Percentage of mis-classified for Test points :"+str(np.round((np.count_nonzero(vclf.predict
(gene_final_X_TEST) - gene_final_Y_TEST)/gene_final_X_TEST.shape[0]*100), 2))+"%")
generate_conf_mat(gene_final_Y_TEST, vclf.predict(gene_final_X_TEST))
```

```
Log loss (train) on the VotingClassifier :0.68
Log loss (CV) on the VotingClassifier :1.13
Log loss (test) on the VotingClassifier :1.11
Percentage of mis-classified for cv points :39.55%
```

```
Percentage of mis-classified for Test points :35.84%
```







In [53]:

```
table = table.append(pd.DataFrame([["Maximum Voting Classifier", 0.68, 1.13, 1.11, "39.55%", "35.84
%", "Best Fit"]], columns = ["Model", "Train Log-loss", "CV Log-loss", "Test Log-loss", "Mis-Classi
fied CV", "Mis-Classified Test", "Remarks"]))
table.reset_index(drop = True, inplace = True)
```

Summary

In [54]:

```
table
```

Out[54]:

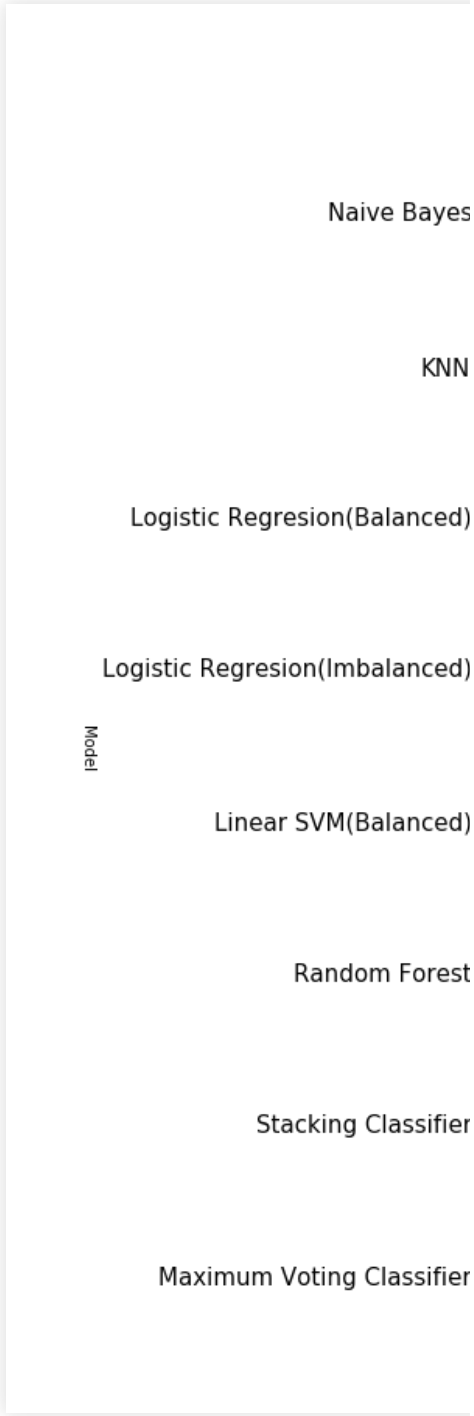
	Model	Train Log-loss	CV Log-loss	Test Log-loss	Mis-Classified CV	Mis-Classified Test	Remarks
0	Naive Bayes	0.9031	1.3471	1.2976	42.75%	39.46%	GoodFit
1	KNN	0.0952	1.4864	1.5290	45.39%	44.88%	OverFit
2	Logistic Regresion(Balanced)	0.6738	1.1314	1.1010	40.49%	36.9%	Good Fit
3	Logistic Regresion(Imbalanced)	0.6640	1.1335	1.0956	39.36%	37.5%	Good Fit
4	Linear SVM(Balanced)	0.8022	1.2399	1.2217	39.36%	38.7%	Good Fit
5	Random Forest	0.6165	1.0961	1.0994	38.23%	35.69%	Best Fit
6	Stacking Classifier	0.3780	1.1311	1.0578	38.98%	35.09%	Best Fit
7	Maximum Voting Classifier	0.6800	1.1300	1.1100	39.55%	35.84%	Best Fit

In [55]:

```
log_loss_table = table.drop(["Mis-Classified CV", "Mis-Classified Test", "Remarks"], axis = 1)
```

In [73]:

```
log_loss_table.plot(x = "Model", kind = "bar", figsize = (14, 8), grid = True, fontsize = 15)
plt.title("Log losses of all the classifiers", fontsize = 20)
plt.ylabel("Log-Loss", fontsize = 20)
plt.show()
```


Naive Bayes
KNN
Logistic Regresion(Balanced)
Logistic Regresion(Imbalanced)
Model
Linear SVM(Balanced)
Random Forest
Stacking Classifier
Maximum Voting Classifier