3.6 Featurizing text data with tfidf weighted word-vectors

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
from nltk.corpus import stopwords
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
warnings.filterwarnings("ignore")
import sys
import os
import pandas as pd
import numpy as np
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from scipy.sparse import coo_matrix, vstack
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier



from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
```

In [2]:

```
# avoid decoding problems
df = pd.read_csv(r"D:\AppliedAI\Homework-n-Assignments\# 20 Quora\train.csv")
df.sort_values(by=['id'])
df = df.head(70000)
# encode questions to unicode
# https://stackoverflow.com/a/6812069
# ---------------- python 2 ---------------------
# df['question1'] = df['question1'].apply(lambda x: unicode(str(x),"utf-8"))
# df['question2'] = df['question2'].apply(lambda x: unicode(str(x),"utf-8"))
# ---------------- python 3 ---------------------
df['question1'] = df['question1'].apply(lambda x: str(x))
df['question2'] = df['question2'].apply(lambda x: str(x))
```

In [3]:

```
print (df.shape)
df.head()
```

```
(70000, 6)
```

Out[3]:

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

In [4]:

```
df_without_qid = df.drop(columns=['qid1', 'qid2','is_duplicate'])
df_without_qid.head()
```

Out[4]:

	id	question1	question2
0	0	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...
1	1	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...
2	2	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...
3	3	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...
4	4	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?

In [5]:

```
if os.path.isfile(r'D:\AppliedAI\Homework-n-Assignments\# 20 Quora\nlp_features_train.csv'):
    dfnlp = pd.read_csv("nlp_features_train.csv",encoding='latin-1')
    dfnlp = dfnlp.head(70000)
else:
    print("download nlp_features_train.csv from drive or run previous notebook")

if os.path.isfile(r'D:\AppliedAI\Homework-n-Assignments\# 20
Quora\df_fe_without_preprocessing_train.csv'):
    dfppro = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')
    dfppro = dfppro.head(70000)

else:
```

```
    print("download df_fe_without_preprocessing_train.csv from drive or run previous notebook")


df1 = dfnlp.drop(['qid1','qid2','question1','question2'],axis=1)
df2 = dfppro.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
#df3 = df.drop(['qid1','qid2','question1','question2','is_duplicate'],axis=1)
```

In [6]:

```
print (df1.shape)
df1.head()
```

```
(70000, 17)
```

Out[6]:

	id	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_word_eq	first_word_eq	abs_len_diff	
0	0	0	0.999980	0.833319	0.999983	0.999983	0.916659	0.785709	0.0	1.0	2.0	
1	1	0	0.799984	0.399996	0.749981	0.599988	0.699993	0.466664	0.0	1.0	5.0	
2	2	0	0.399992	0.333328	0.399992	0.249997	0.399996	0.285712	0.0	1.0	4.0	
3	3	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	2.0	
4	4	0	0.399992	0.199998	0.999950	0.666644	0.571420	0.307690	0.0	1.0	6.0	

In [7]:

```
print (df2.shape)
df2.head()
```

```
(70000, 12)
```

Out[7]:

	id	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_Common	word_Total	word_share	freq_q1+q2	f
0	0	1	1	66	57	14	12	10.0	23.0	0.434783	2	0
1	1	4	1	51	88	8	13	4.0	20.0	0.200000	5	3
2	2	1	1	73	59	14	10	4.0	24.0	0.166667	2	0
3	3	1	1	50	65	11	9	0.0	19.0	0.000000	2	0
4	4	3	1	76	39	13	7	2.0	20.0	0.100000	4	2

In [8]:

```
df4 =pd.merge(df1, df2, on='id')
df4.head()
```

Out[8]:

	id	is_duplicate	cwc_min	cwc_max	csc_min	csc_max	ctc_min	ctc_max	last_word_eq	first_word_eq	...	freq_qid2
0	0	0	0.999980	0.833319	0.999983	0.999983	0.916659	0.785709	0.0	1.0	...	1
1	1	0	0.799984	0.399996	0.749981	0.599988	0.699993	0.466664	0.0	1.0	...	1
2	2	0	0.399992	0.333328	0.399992	0.249997	0.399996	0.285712	0.0	1.0	...	1
3	3	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	...	1
4	4	0	0.399992	0.199998	0.999950	0.666644	0.571420	0.307690	0.0	1.0	...	1

5 rows × 28 columns

In [9]:

```
df_final =pd.merge(df_without_qid, df4, on='id')
#print (df_final.shape)
```

In [10]:

```
df_final_train, df_final_test = train_test_split(df_final, test_size=0.33, shuffle=False)

#print (df_final_train.shape)
```

In [11]:

```
q1_train = df_final_train['question1'].values
q1_test = df_final_test['question1'].values

q2_train = df_final_train['question2'].values
q2_test = df_final_test['question2'].values

y_train = df_final_train['is_duplicate'].values
y_test = df_final_test['is_duplicate'].values
```

In [12]:

```
tf_idf_vect = TfidfVectorizer(min_df=4,ngram_range=(1,3),max_features=2500)
tfidf_q1_train = tf_idf_vect.fit_transform(q1_train)
tfidf_q1_test = tf_idf_vect.transform(q1_test)

tfidf_q2_train = tf_idf_vect.fit_transform(q2_train)
tfidf_q2_test = tf_idf_vect.transform(q2_test)

q1_train_std =StandardScaler(with_mean=False,with_std=False).fit_transform(tfidf_q1_train)
q1_test_std =StandardScaler(with_mean=False,with_std=False).fit_transform(tfidf_q1_test)

q2_train_std =StandardScaler(with_mean=False,with_std=False).fit_transform(tfidf_q2_train)
q2_test_std =StandardScaler(with_mean=False,with_std=False).fit_transform(tfidf_q2_test)

print (q1_train_std.shape)
print (q1_test_std.shape)
print ("########")
print (q2_train_std.shape)
print (q2_test_std.shape)
```

```
(46900, 2500)
(23100, 2500)
########
(46900, 2500)
(23100, 2500)
```

In [13]:

```
#https://stackoverflow.com/questions/45961747/append-tfidf-to-pandas-dataframe
#https://www.researchgate.net/post/How_to_append_TF-IDF_vector_into_pandas_dataframe

q1_train_std_df = pd.DataFrame(q1_train_std.toarray())
q1_test_std_df = pd.DataFrame(q1_test_std.toarray())

q2_train_std_df = pd.DataFrame(q2_train_std.toarray())
q2_test_std_df = pd.DataFrame(q2_test_std.toarray())

print (q1_train_std_df.shape)
print (q1_test_std_df.shape)
print ("########")
print (q2_train_std_df.shape)
print (q2_test_std_df.shape)
```

```
(46900, 2500)
(23100, 2500)
########
(46900, 2500)
(23100, 2500)
```

(25100, 2500)

In [14]:

```
from scipy.sparse import coo_matrix, hstack
import scipy.sparse as ss

train_after_dropping = df_final_train.drop(['id','question1','question2','is_duplicate'],axis=1)
print (train_after_dropping.shape)
X_train = pd.concat([train_after_dropping, q2_train_std_df,q1_train_std_df], axis=1)

test_after_dropping = df_final_test.drop(['id','question1','question2','is_duplicate'],axis=1)
X_test1 = pd.concat([q2_test_std_df,q1_test_std_df],axis=1)
X_test = ss.hstack([test_after_dropping,X_test1])

print("Number of data points in train data :",X_train.shape)
print("Number of data points in test data :",X_test.shape)
```

```
(46900, 26)
Number of data points in train data : (46900, 5026)
Number of data points in test data : (23100, 5026)
```

In [15]:

```
print("-"*10, "Distribution of output variable in train data", "-"*10)
train_distr = Counter(y_train)
train_len = len(y_train)
print("Class 0: ",int(train_distr[0])/train_len,"Class 1: ", int(train_distr[1])/train_len)
print("-"*10, "Distribution of output variable in train data", "-"*10)
test_distr = Counter(y_test)
test_len = len(y_test)
print("Class 0: ",int(test_distr[1])/test_len, "Class 1: ",int(test_distr[1])/test_len)
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.626226012793177 Class 1:  0.37377398720682303
---------- Distribution of output variable in train data ----------
Class 0:  0.36978354978354977 Class 1:  0.36978354978354977
```

In [16]:

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

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

    # C = [[1, 2],
    #     [3, 4]]
    # C.T = [[1, 3],
    #        [2, 4]]
    # C.sum(axis = 1)  axis=0 corresonds to columns and axis=1 corresponds to rows in two
diamensional array
    # C.sum(axix =1) = [[3, 7]]
    # ((C.T)/(C.sum(axis=1))) = [[1/3, 3/7]
    #                            [2/3, 4/7]]

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

    B =(C/C.sum(axis=0))
    #divid each element of the confusion matrix with the sum of elements in that row
    # C = [[1, 2],
    #     [3, 4]]
    # C.sum(axis = 0)  axis=0 corresonds to columns and axis=1 corresponds to rows in two
diamensional array
    # C.sum(axix =0) = [[4, 6]]
    # (C/C.sum(axis=0)) = [[1/4, 2/6],
    #                      [3/4, 4/6]]
    plt.figure(figsize=(20,4))

    labels = [1,2]
```

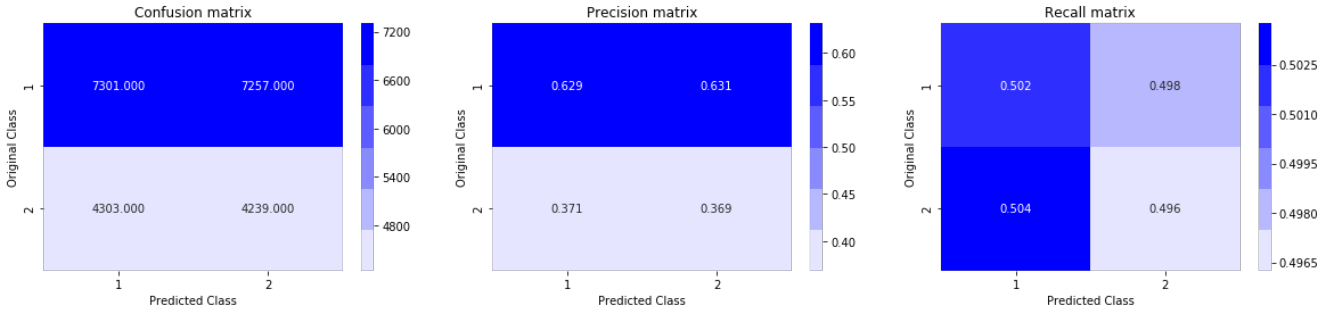
```
    labels = [1,2]
    # representing A in heatmap format
    cmap=sns.light_palette("blue")
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")

    plt.show()
```

4.4 Building a random model (Finding worst-case log-loss)

In [17]:

```
# we need to generate 9 numbers and the sum of numbers should be 1
# one solution is to genarate 9 numbers and divide each of the numbers by their sum
# ref: https://stackoverflow.com/a/18662466/4084039
# we create a output array that has exactly same size as the CV data
predicted_y = np.zeros((test_len,2))
for i in range(test_len):
    rand_probs = np.random.rand(1,2)
    predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Test Data using Random Model",log_loss(y_test, predicted_y, eps=1e-15))

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

```
Log loss on Test Data using Random Model 0.8828045187893093
```



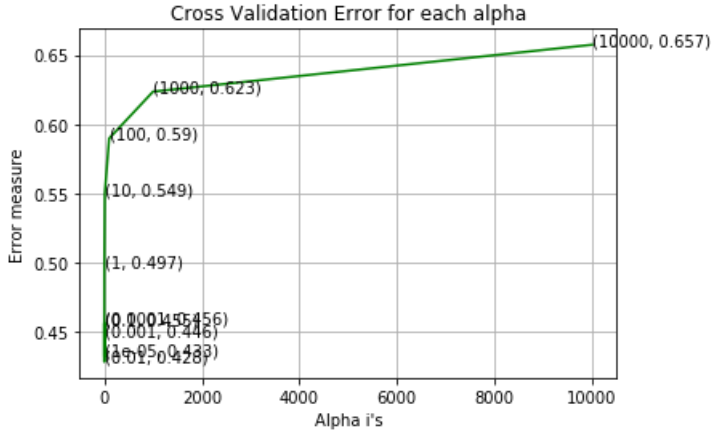
4.4 Logistic Regression with hyperparameter tuning

In [18]:

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

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

# some of methods
```

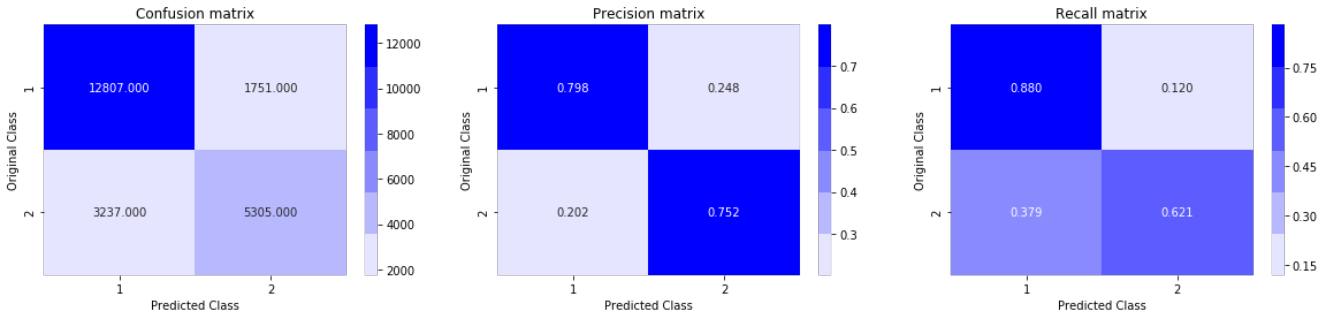
```
# some of methods
# fit(X, y[, coef_init, intercept_init, …]) Fit linear model with Stochastic Gradient Descent.
# predict(X) Predict class labels for samples in X.

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


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

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


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

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train,
predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, p
redict_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.4326349740884927
For values of alpha =  0.0001 The log loss is: 0.4559244732683729
For values of alpha =  0.001 The log loss is: 0.44649471595806595
For values of alpha =  0.01 The log loss is: 0.4277734055419205
For values of alpha =  0.1 The log loss is: 0.45450004535579464
For values of alpha =  1 The log loss is: 0.4967075198613771
For values of alpha =  10 The log loss is: 0.5492684510107138
For values of alpha =  100 The log loss is: 0.5895591902275101
For values of alpha =  1000 The log loss is: 0.6234599922698741
For values of alpha =  10000 The log loss is: 0.6574181624563182
```




```
For values of best alpha =  0.01 The train log loss is: 0.42679916662788203
For values of best alpha =  0.01 The test log loss is: 0.4277734055419205
```

```
for values of best alpha =  0.01 The test log loss is: 0.427773405541920
Total number of data points : 23100
```



4.5 Linear SVM with hyperparameter tuning

In [19]:

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

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

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

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


log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l1', loss='hinge', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_test, predict_y, labels=clf.cl
asses_, eps=1e-15))

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


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l1', loss='hinge', random_state=42)
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)

predict_y = sig_clf.predict_proba(X_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",log_loss(y_train,
predict_y, labels=clf.classes_, eps=1e-15))
predict_y = sig_clf.predict_proba(X_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",log_loss(y_test, p
redict_y, labels=clf.classes_, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
```

```
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.41880382365794316
For values of alpha =  0.0001 The log loss is: 0.43290410044318994
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:561:
ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing ma
x_iter to improve the fit.
  ConvergenceWarning)
```

```
For values of alpha =  0.001 The log loss is: 0.444666042506562
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:561:
ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing ma
x_iter to improve the fit.
  ConvergenceWarning)
```

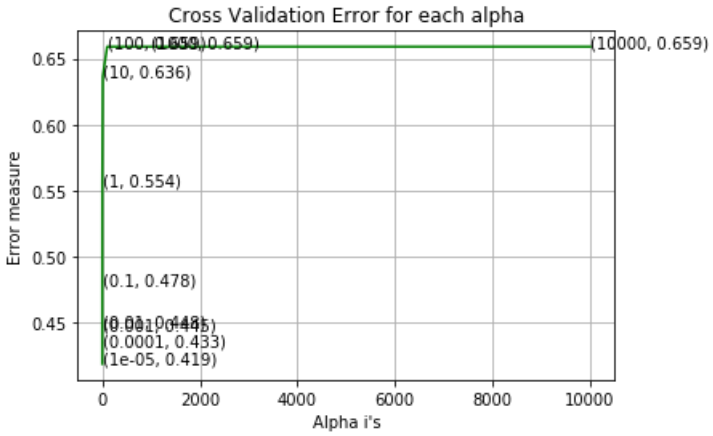
```
For values of alpha =  0.01 The log loss is: 0.44762996673875544
For values of alpha =  0.1 The log loss is: 0.47810780280316073
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:561:
ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing ma
x_iter to improve the fit.
  ConvergenceWarning)
```

```
For values of alpha =  1 The log loss is: 0.5544700856710907
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:561:
ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing ma
x_iter to improve the fit.
  ConvergenceWarning)
```

```
For values of alpha =  10 The log loss is: 0.6363697526809011
For values of alpha =  100 The log loss is: 0.6588744461963044
For values of alpha =  1000 The log loss is: 0.6588744461963039
For values of alpha =  10000 The log loss is: 0.6588744461963038
```




```
For values of best alpha =  1e-05 The train log loss is: 0.4137388154907093
For values of best alpha =  1e-05 The test log loss is: 0.41880382365794316
Total number of data points : 23100
```