#Try any of the feature engineering techniques discussed in the course to reduce the CV and test log-loss to a value less than 1.0

SO WE GONNA USE THE EFATURE ENGINEERING TECHNIQUE TOREDUCE THE LOGLOSS LESS THAN 1. WE USE THE MODELS AND REST FEATURES STACK THEM AND GET THE MODEL PERFORMANCE IN TERMS OF LOGLOSS WHICH IS LESS THAN 1 FOR THE TEST DATA. ¶

In [0]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from imblearn.over_sampling import SMOTE
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC

from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier
warnings.filterwarnings("ignore")

from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
```

In [0]:
```
# Code to read csv file into Colaboratory:
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

In [0]:
```
link = 'https://drive.google.com/open?id=138eFlNxRXZAZGDTIoC7U9N1LlTmw1JwQ' # Th
```

In [0]:
```
fluff, id = link.split('=')
print (id) # Verify that you have everything after '='
```

```
138eFlNxRXZAZGDTIoC7U9N1LlTmw1JwQ
```

In [0]:
```
import pandas as pd
downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('trainingtext')
```

In [0]:
```
data_text = pd.read_csv('trainingtext',sep="\|\|",engine="python",names=["ID","TI
print('Number of data points : ', data_text.shape[0])
print('Number of features : ', data_text.shape[1])
print('Features : ', data_text.columns.values)
data_text.head()
```

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

Out[7]:

	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 [0]:
```
# Code to read csv file into Colaboratory:
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

In [0]:
```
link = 'https://drive.google.com/open?id=1pqKC1FvjAje0WV1mEZexzeLRVQDzxxSL' # Th
```

In [0]:
```
fluff, id = link.split('=')
print (id) # Verify that you have everything after '='
```

```
1pqKC1FvjAje0WV1mEZexzeLRVQDzxxSL
```

In [0]:
```
import pandas as pd
downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('trainingvariants')
```

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

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

Out[12]:

	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 [0]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import log_loss
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.calibration import CalibratedClassifierCV
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import re
from nltk.corpus import stopwords
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import TruncatedSVD
warnings.filterwarnings('ignore')
from sklearn.ensemble import RandomForestClassifier
from scipy.sparse import hstack
from sklearn.metrics.classification import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.multiclass import OneVsRestClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [0]:
```
import nltk
nltk.download("popular")
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to /root/nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package omw to /root/nltk_data...
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection popular
```

Out[14]:
```
True
```

In [0]:
```
# loading stop words from nltk library
stop_words = set(stopwords.words('english'))


def nlp_preprocessing(total_text, index, column):
    if type(total_text) is not int:
        string = ""
        # replace every special char with space
        total_text = re.sub('[^a-zA-Z0-9\n]', ' ', total_text)
        # replace multiple spaces with single space
        total_text = re.sub('\s+',' ', total_text)
        # converting all the chars into lower-case.
        total_text = total_text.lower()

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

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

In [0]:
```
#text processing stage.
start_time = time.clock()
for index, row in data_text.iterrows():
    if type(row['TEXT']) is str:
        nlp_preprocessing(row['TEXT'], index, 'TEXT')
    else:
        print("there is no text description for id:",index)
print('Time took for preprocessing the text :',time.clock() - start_time, "secon
```

```
there is no text description for id: 1109
there is no text description for id: 1277
there is no text description for id: 1407
there is no text description for id: 1639
there is no text description for id: 2755
Time took for preprocessing the text : 372.203488 seconds
```

In [0]:
```
#merging both gene_variations and text data based on ID
result = pd.merge(data, data_text,on='ID', how='left')
result.head()
```

Out[17]:

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

In [0]:
```
result[result.isnull().any(axis=1)]
```

Out[18]:

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

In [0]:
```
result.loc[result['TEXT'].isnull(),'TEXT'] = result['Gene'] +' '+result['Variati
```

In [0]:
```
result[result['ID']==1109]
```

Out[20]:

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

In [0]:
```
result.to_csv('personalcancerassignmentdata.csv')
```

In [0]:
```
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
# This only needs to be done once in a notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# Create & upload a file.
uploaded = drive.CreateFile({'title': 'personalcancerassignmentdata.csv'})
uploaded.SetContentFile('personalcancerassignmentdata.csv')
uploaded.Upload()
print('Uploaded file with ID {}'.format(uploaded.get('id')))
```

```
Uploaded file with ID 1ClZQUuBTtIatBFZpPz7rXimqukBUDPWc
```

3.1.4. Test, Train and Cross Validation Split

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

In [0]:
```
y_true = result['Class'].values
result.Gene      = result.Gene.str.replace('\s+', '_')
result.Variation = result.Variation.str.replace('\s+', '_')

# split the data into test and train by maintaining same distribution of output v
X_train, test_df, y_train, y_test = train_test_split(result, y_true, stratify=y_t
# split the train data into train and cross validation by maintaining same distri
train_df, cv_df, y_train, y_cv = train_test_split(X_train, y_train, stratify=y_tr
```

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

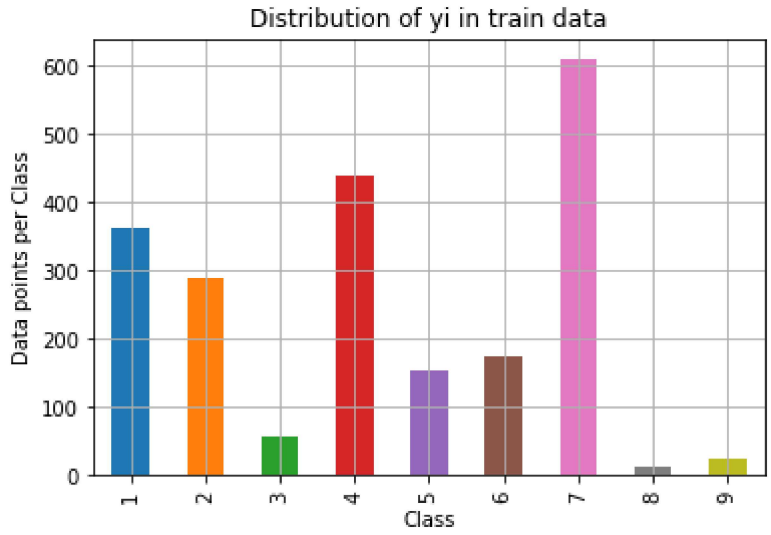
In [0]:
```
print('Number of data points in train data:', train_df.shape[0])
print('Number of data points in test data:', test_df.shape[0])
print('Number of data points in cross validation data:', cv_df.shape[0])
```

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

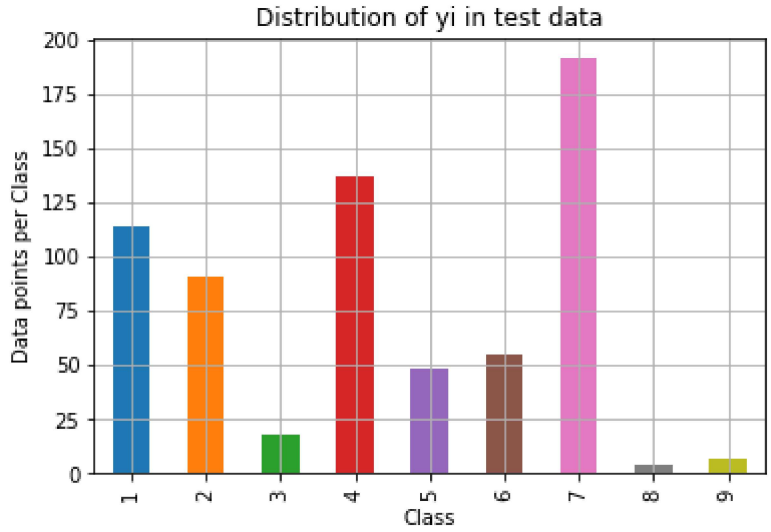
3.1.4.2. Distribution of y_i's in Train, Test and Cross Validation datasets

In [0]:

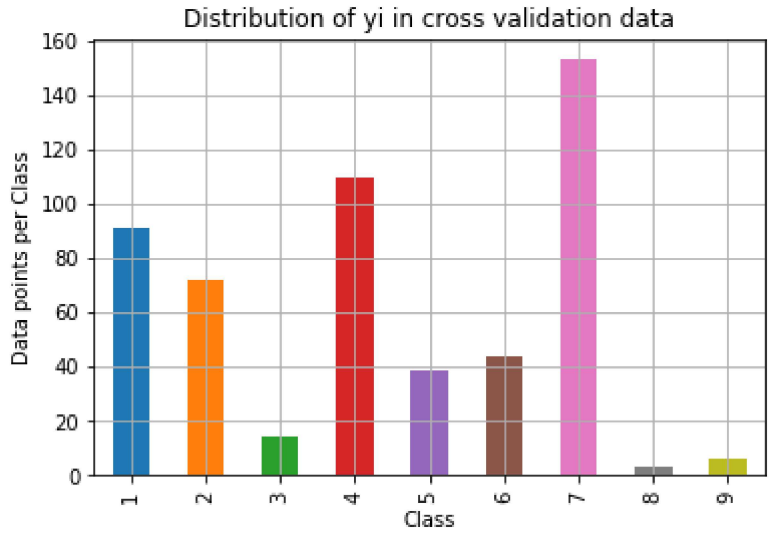
```
# it returns a dict, keys as class labels and values as the number of data points
train_class_distribution = train_df['Class'].value_counts().sortlevel()
test_class_distribution = test_df['Class'].value_counts().sortlevel()
cv_class_distribution = cv_df['Class'].value_counts().sortlevel()

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

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort
# -(train_class_distribution.values): the minus sign will give us in decreasing o
sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',train_class_distribution.val


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

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort
# -(train_class_distribution.values): the minus sign will give us in decreasing o
sorted_yi = np.argsort(-test_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',test_class_distribution.valu

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

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort
# -(train_class_distribution.values): the minus sign will give us in decreasing o
sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',cv_class_distribution.values
```




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




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




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

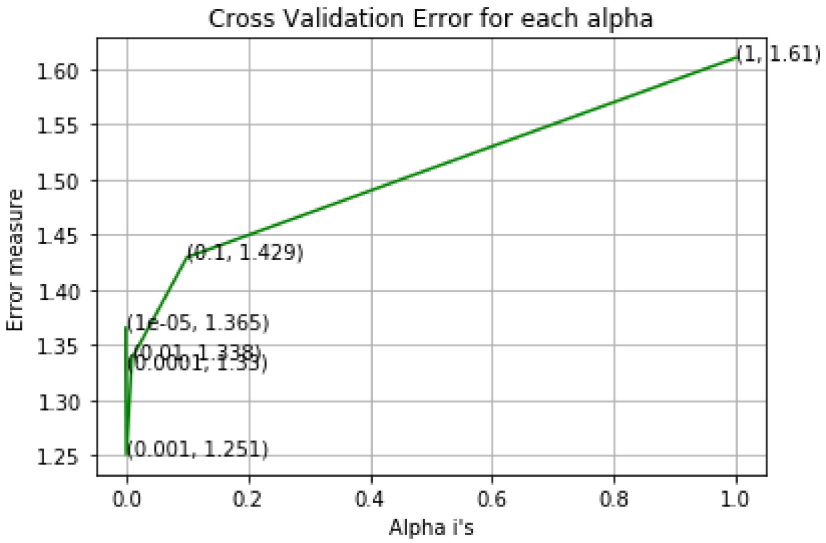
In [0]:
```
# building a CountVectorizer with all the words that occured minimum 3 times in t
text_vectorizer = CountVectorizer(min_df=3)
train_text_feature_onehotCoding = text_vectorizer.fit_transform(train_df['TEXT']
# getting all the feature names (words)
train_text_features= text_vectorizer.get_feature_names()

# train_text_feature_onehotCoding.sum(axis=0).A1 will sum every row and returns
train_text_fea_counts = train_text_feature_onehotCoding.sum(axis=0).A1

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


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

```
Total number of unique words in train data : 53099
```

In [0]:
```
train_text_feature_onehotCoding = normalize(train_text_feature_onehotCoding, axi
test_text_feature_onehotCoding = text_vectorizer.transform(test_df['TEXT'])

test_text_feature_onehotCoding = normalize(test_text_feature_onehotCoding, axis=

cv_text_feature_onehotCoding = text_vectorizer.transform(cv_df['TEXT'])

cv_text_feature_onehotCoding = normalize(cv_text_feature_onehotCoding, axis=0)
```

In [0]:
```
```

```
For values of alpha =  1e-05 The log loss is: 1.365134636438072
For values of alpha =  0.0001 The log loss is: 1.330123246189539
For values of alpha =  0.001 The log loss is: 1.2506545346314497
For values of alpha =  0.01 The log loss is: 1.3377680791331388
For values of alpha =  0.1 The log loss is: 1.429148512279061
For values of alpha =  1 The log loss is: 1.60992699780569
```

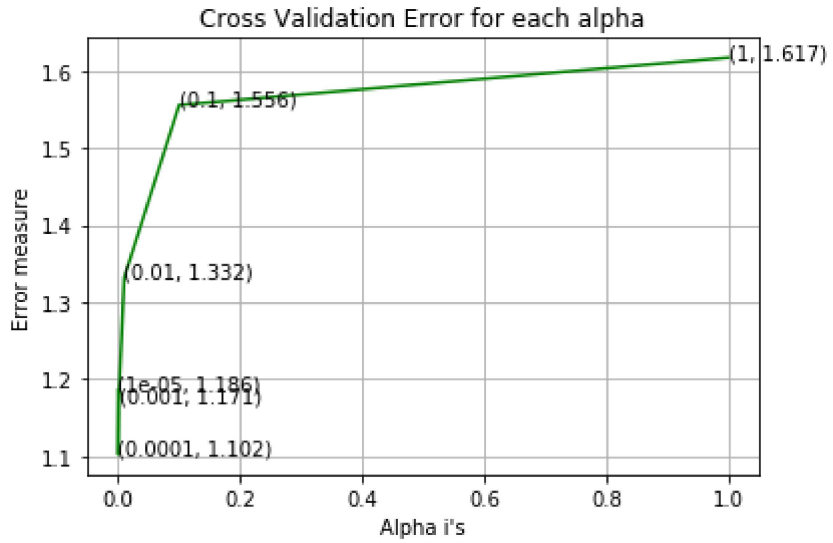



```
For values of best alpha =  0.001 The train log loss is: 0.7818685431302675
For values of best alpha =  0.001 The cross validation log loss is: 1.250654534
6314497
For values of best alpha =  0.001 The test log loss is: 1.1642257777705138
```

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidftextvec=TfidfVectorizer(min_df=3)
traintextfeaturetfidfcoding=tfidftextvec.fit_transform(train_df['TEXT'])
traintextfeaturetfidfcoding=normalize(traintextfeaturetfidfcoding)
testtextfeaturetfidfcoding=tfidftextvec.transform(test_df['TEXT'])
testtextfeaturetfidfcoding=normalize(testtextfeaturetfidfcoding)
cvtextfeaturetfidfcoding=tfidftextvec.transform(cv_df['TEXT'])
cvtextfeaturetfidfcoding=normalize(cvtextfeaturetfidfcoding)
```

In [0]:
```
alpha = [10 ** x for x in range(-5, 1)]


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

    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(traintextfeaturetfidfcoding, y_train)
    predict_y = sig_clf.predict_proba(cvtextfeaturetfidfcoding)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=clf.classes_, eps
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_cv, predict_

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


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

predict_y = sig_clf.predict_proba(traintextfeaturetfidfcoding)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:"
predict_y = sig_clf.predict_proba(cvtextfeaturetfidfcoding)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation lo
predict_y = sig_clf.predict_proba(testtextfeaturetfidfcoding)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",
```

```
For values of alpha =  1e-05 The log loss is: 1.1857761596796688
For values of alpha =  0.0001 The log loss is: 1.1019567587095758
For values of alpha =  0.001 The log loss is: 1.1705842362179595
For values of alpha =  0.01 The log loss is: 1.3315850823008422
For values of alpha =  0.1 The log loss is: 1.5556600692869964
For values of alpha =  1 The log loss is: 1.6170859658524854
```




```
For values of best alpha =  0.0001 The train log loss is: 0.7074137555062421
For values of best alpha =  0.0001 The cross validation log loss is: 1.10195675
87095758
For values of best alpha =  0.0001 The test log loss is: 1.0030328617447106
```

WITH OUT USING THE FEATURE ENGINEERING TECHNIQUES USING THE TFIDF CODING OF THE DATA WE HAVE ACHIEVED THE LOG LOSS OF 1.0.NOW ALONG WIHT THIS FEATURE WE WILL PERFORM FEATRE ENGINEEINRG TECHINIQUE OF CONCATENATION OF THE GENE AND VARIATION FEATUREAND DEPLOY INTO THE MODELS AND CHECK THE PERFORMANCE.

In [0]:
```
result.head(5)
```

Out[44]:

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

In [0]:
```
resultnew=result
resultnew['genevariation']=resultnew['Gene']+' '+resultnew['Variation']
```

In [0]:
```
resultnew.head(5)
```

Out[51]:

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

In [0]:
```
y_true = resultnew['Class'].values
resultnew.Gene      = resultnew.Gene.str.replace('\s+', '_')
resultnew.Variation = resultnew.Variation.str.replace('\s+', '_')

# split the data into test and train by maintaining same distribution of output v
X_train, test_df, y_train, y_test = train_test_split(resultnew, y_true, stratify=
# split the train data into train and cross validation by maintaining same distr
train_df, cv_df, y_train, y_cv = train_test_split(X_train, y_train, stratify=y_tr
```

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

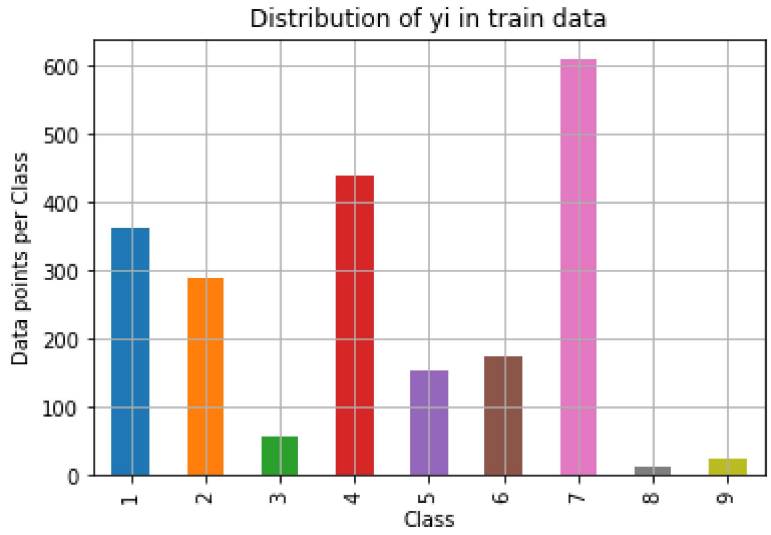
In [0]:
```
print('Number of data points in train data:', train_df.shape[0])
print('Number of data points in test data:', test_df.shape[0])
print('Number of data points in cross validation data:', cv_df.shape[0])
```

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

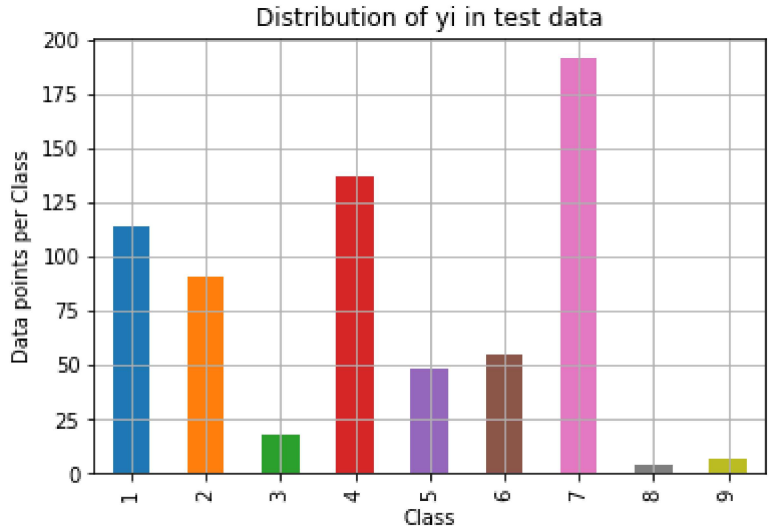
3.1.4.2. Distribution of y_i's in Train, Test and Cross Validation datasets

In [0]:

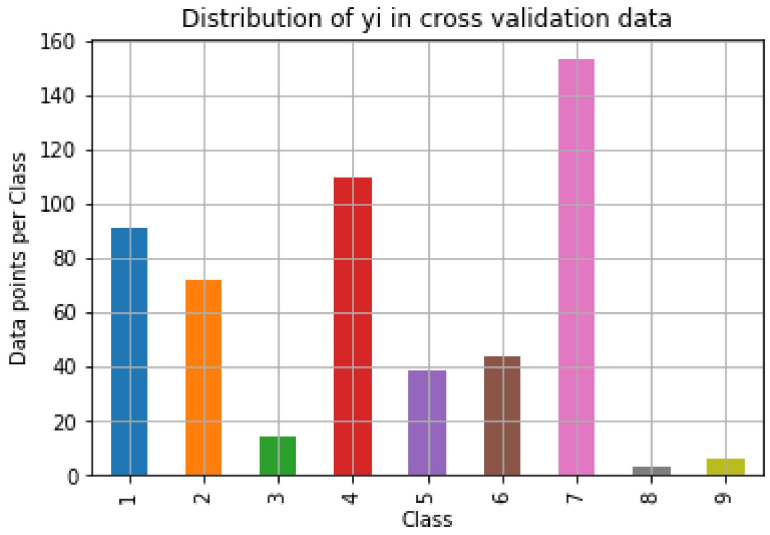
```
# it returns a dict, keys as class labels and values as the number of data points
train_class_distribution = train_df['Class'].value_counts().sortlevel()
test_class_distribution = test_df['Class'].value_counts().sortlevel()
cv_class_distribution = cv_df['Class'].value_counts().sortlevel()

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

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort
# -(train_class_distribution.values): the minus sign will give us in decreasing o
sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',train_class_distribution.val


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

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort
# -(train_class_distribution.values): the minus sign will give us in decreasing o
sorted_yi = np.argsort(-test_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',test_class_distribution.valu

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

# ref: argsort https://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort
# -(train_class_distribution.values): the minus sign will give us in decreasing o
sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',cv_class_distribution.values
```




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




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




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

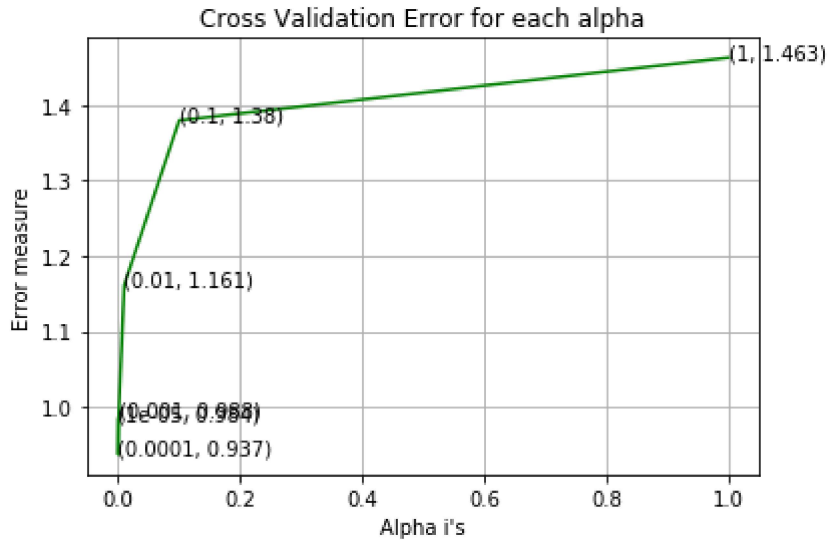
In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidftextvec=TfidfVectorizer(min_df=3)
traintextfeaturetfidfcoding=tfidftextvec.fit_transform(train_df['TEXT'])
traintextfeaturetfidfcoding=normalize(traintextfeaturetfidfcoding)
testtextfeaturetfidfcoding=tfidftextvec.transform(test_df['TEXT'])
testtextfeaturetfidfcoding=normalize(testtextfeaturetfidfcoding)
cvtextfeaturetfidfcoding=tfidftextvec.transform(cv_df['TEXT'])
cvtextfeaturetfidfcoding=normalize(cvtextfeaturetfidfcoding)
```

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
genevariationtfidfvect=TfidfVectorizer()
genevariationtraintfidfcoding=genevariationtfidfvect.fit_transform(train_df['gen
genevariationtraintfidfcoding=normalize(genevariationtraintfidfcoding)
genevariationtesttfidfcoding=genevariationtfidfvect.transform(test_df['genevaria
genevariationtesttfidfcoding=normalize(genevariationtesttfidfcoding)
genevariationcvtfidfcoding=genevariationtfidfvect.transform(cv_df['genevariation
genevariationcvtfidfcoding=normalize(genevariationcvtfidfcoding)
```

In [0]:
```
from scipy.sparse import hstack
task4traindata=hstack((genevariationtraintfidfcoding,traintextfeaturetfidfcoding
task4cvdata=hstack((genevariationcvtfidfcoding,cvtextfeaturetfidfcoding)).tocsr(
task4testdata=hstack((genevariationtesttfidfcoding,testtextfeaturetfidfcoding)).
```

In [0]:
```
alpha = [10 ** x for x in range(-5, 1)]


cv_log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(task4traindata, y_train)

    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(task4traindata, y_train)
    predict_y = sig_clf.predict_proba(task4cvdata)
    cv_log_error_array.append(log_loss(y_cv, predict_y, labels=clf.classes_, eps
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_cv, predict_

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


best_alpha = np.argmin(cv_log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_st
clf.fit(task4traindata, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(task4traindata, y_train)

predict_y = sig_clf.predict_proba(task4traindata)
print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:"
predict_y = sig_clf.predict_proba(task4cvdata)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation lo
predict_y = sig_clf.predict_proba(task4testdata)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",
```

```
For values of alpha =  1e-05 The log loss is: 0.9836849377952454
For values of alpha =  0.0001 The log loss is: 0.936712446223448
For values of alpha =  0.001 The log loss is: 0.98811318763304
For values of alpha =  0.01 The log loss is: 1.1612293838083256
For values of alpha =  0.1 The log loss is: 1.3795796719180566
For values of alpha =  1 The log loss is: 1.4627081202181018
```




```
For values of best alpha =  0.0001 The train log loss is: 0.4219797187908997
For values of best alpha =  0.0001 The cross validation log loss is: 0.93671244
6223448
For values of best alpha =  0.0001 The test log loss is: 0.9806514463343825
```

###PERFORMANCE AND DOCUMENTATION

- WE HAVE PERFORMED THE TFIDF CODING OGF THE TEXT VECTOR AND ABLE TO ACHIEVE THE LOGLOSS OF 1.0
- WE HAVE PERFORMED THE FEATURE ENGINEERING TECHNIQUE AND OBTAINRD TEST LOGLOSS OF 0.98 WHICH IS LESS THAN 1.0
- WE AVE USED THE TFIDF ENCODING OBTAINED THE LOGLOSS. ####PERFORMANCE OF THE MODEL IS AS FOLLOWS

In [0]:
```
import pandas as pd
dta=[]
dta = [['WITH OUT FEATURE ENGINEERING','LOGISTIC REGRESSION','ALPHA=0.0001',0.7,1
aa=pd.DataFrame(dta, columns=['feature engineering','model','BEST HYPER PARAMETEI
```

In [4]:
```
aa
```

Out[4]:

	feature engineering	model	BEST HYPER PARAMETER	TRAIN_LOG_LOSS	CROSS_VALIDATAION_LOGLOS
0	WITH OUT FEATURE ENGINEERING	LOGISTIC REGRESSION	ALPHA=0.0001	0.700	1.10
1	WITH FEATURE ENGINEERING	LOGISTIC REGRESSION	ALPHA=0.0001	0.421	0.93

In [0]:
```
```