In [1]:

```
import pandas as pd
import random
import numpy as np
import gzip
import numpy
import json
import nltk
import string
from nltk.stem.porter import *
from collections import defaultdict
```

In [104]:

```
def readGz(path):
    for l in gzip.open(path, 'rt'):
        yield eval(l)

def readCSV(path):
    f = gzip.open(path, 'rt')
    f.readline()
    for l in f:
        yield l.strip().split(',')
```

In [105]:

```
df = pd.read_csv('train_Interactions.csv.gz')
```

In [106]:

```
df['read'] = 1
```

In [107]:

```
df.head()
```

Out[107]:

	userID	bookID	rating	read
0	u79354815	b14275065	4	1
1	u56917948	b82152306	5	1
2	u97915914	b44882292	5	1
3	u49688858	b79927466	5	1
4	u08384938	b05683889	2	1

Question1

In [10]:

```
train_set, vali_set = df[:190000], df[190000:]
```

In [11]:

```
dic_test = {}
book_lst = []
with open("pairs_Read.txt") as f:
    for line in f:
        (key, val) = line.split('-')
        book_lst.append(val.strip('\n'))
        dic_test[key] = val.strip('\n')
```

In [12]:

```
book_list = df['bookID'].unique()
read_list = df.groupby('userID')['bookID'].apply(list)
read_list = read_list.to_dict()
```

In [13]:

```
def choice(x):
    books = list(book_list)
    booksread = read_list[x]
    val = np.random.choice(books)
    if val not in booksread:
        return val
    else:
        return np.random.choice(books)
```

In [14]:

```
vali_notread = vali_set.copy()
vali_notread['bookID'] = vali_set['userID'].apply(choice)
vali_notread['read'] = 0
```

In [15]:

```
print(vali_notread.head())
print(vali_set.head())
```

```
           userID     bookID  rating  read
190000  u35176258  b69919870       3     0
190001  u30851063  b51193385       3     0
190002  u31368414  b36888197       5     0
190003  u71352502  b33332201       2     0
190004  u46986025  b55056005       3     0
           userID     bookID  rating  read
190000  u35176258  b30592470       3     1
190001  u30851063  b81941226       3     1
190002  u31368414  b40097012       5     1
190003  u71352502  b25118404       2     1
190004  u46986025  b89866434       3     1
```

In [16]:

```
balanced_vali = vali_set.append(vali_notread)
```

In [17]:

```
balanced_vali.shape
```

Out[17]:

```
(20000, 4)
```

In [18]:

```
balanced_vali.to_csv('balanced_validation.csv.gz')
train_set.to_csv('train_predcition_forbalance.csv.gz')
```

In [19]:

```
### Would-read baseline: just rank which books are popular and w
hich are not, and return '1' if a book is among the top-ranked

bookCount = defaultdict(int)
totalRead = 0
lst_pred = []

for _,user,book,_,_ in readCSV("train_predcition_forbalance.csv.
gz"):
    bookCount[book] += 1
    totalRead += 1

mostPopular = [(bookCount[x], x) for x in bookCount]
mostPopular.sort()
mostPopular.reverse()

return1 = set()
count = 0
for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalRead/2: break

for b in balanced_vali['bookID']:
    if b in return1:
        lst_pred.append(1)
    else:
        lst_pred.append(0)
```

In [20]:

```
# True positives, false positives, etc.

ytest = balanced_vali['read']
pred = lst_pred

TP_ = numpy.logical_and(pred, ytest)
FP_ = numpy.logical_and(pred, numpy.logical_not(ytest))
TN_ = numpy.logical_and(numpy.logical_not(pred), numpy.logical_n
ot(ytest))
FN_ = numpy.logical_and(numpy.logical_not(pred), ytest)

TP = sum(TP_)
FP = sum(FP_)
TN = sum(TN_)
FN = sum(FN_)

# accuracy
(TP + TN) / (TP + FP + TN + FN)
```

Out[20]:

```
0.6455
```

Question2

In [24]:

```
### Would-read baseline: just rank which books are popular and w
hich are not, and return '1' if a book is among the top-ranked

bookCount = defaultdict(int)
totalRead = 0
lst_pred = []

for _,user,book,_,_ in readCSV("train_predcition_forbalance.csv.
gz"):
    bookCount[book] += 1
    totalRead += 1

mostPopular = [(bookCount[x], x) for x in bookCount]
mostPopular.sort()
```

```
mostPopular.reverse()


return1 = set()
count = 0
for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalRead*(65/100): break

for b in balanced_vali['bookID']:
    if b in return1:
        lst_pred.append(1)
    else:
        lst_pred.append(0)


ytest = balanced_vali['read']
pred = lst_pred

TP_ = numpy.logical_and(pred, ytest)
FP_ = numpy.logical_and(pred, numpy.logical_not(ytest))
TN_ = numpy.logical_and(numpy.logical_not(pred), numpy.logical_n
ot(ytest))
FN_ = numpy.logical_and(numpy.logical_not(pred), ytest)

TP = sum(TP_)
FP = sum(FP_)
TN = sum(TN_)
FN = sum(FN_)

# accuracy
(TP + TN) / (TP + FP + TN + FN)
```

Out[24]:

```
0.6497
```

After changing threshold to 65 percentile, the accuracy gets higher which is 0.6497 compared to 50% percentile's value which is 0.6455.

Question3

In [26]:

```
readlst = train_set.groupby('userID')['bookID'].apply(list)
readlst = readlst.to_dict()

userlst = train_set.groupby('bookID')['userID'].apply(list)
userlst = userlst.to_dict()
```

In [55]:

```
def jaccard(x):
    if x['bookID'] in list(train_set['bookID'].values):
        u = set(userlst[x.bookID])
    else:
        u = set()
    jacc_lst = [0]
    for b in readlst[x['userID']]:
        j = len(set(userlst[b]).intersection(u))/len(set(userlst
[b]).union(u))
        jacc_lst.append(j)
        maxi_jacc = max(jacc_lst)
    return maxi_jacc
```

In [56]:

```
jaccs = balanced_vali.apply(jaccard, axis = 1)
```

In [58]:

```
pred = jaccs.apply(lambda x: 1 if x > 0.01 else 0)
```

In [61]:

```
sum(pred==balanced_vali['read'])/len(pred)
```

Out[61]:

```
0.624
```

Question4

We combine the prediction result from question2 and question3, and check whether they gets the same result as validation set.

In [69]:

```
both = (pred + lst_pred) == 2
```

In [70]:

```
sum(both == balanced_vali['read'])/len(pred)
```

Out[70]:

```
0.6589
```

The accuracy increases to 0.6589

Question5

My kaggle name: kkkkkk

In [79]:

```
testset = pd.DataFrame({'userID':[], 'bookID':[]})
for l in open("pairs_Read.txt"):
    if not l.startswith("userID"):
        u,b = l.strip().split('-')
        testset = testset.append({'userID': u , 'bookID': b}, ig
nore_index=True)
```

In [80]:

```
testset.head()
```

Out[80]:

	userID	bookID
0	u65407115	b69897799
1	u53740605	b39436893
2	u88031275	b83889575
3	u99759913	b39270822
4	u20090895	b47380623

In [147]:

```
bookCount = defaultdict(int)
totalRead = 0
lst_pred1_q5 = []

for user,book,_,_  in np.array(df.values):
    bookCount[book] += 1
    totalRead += 1

mostPopular = [(bookCount[x], x) for x in bookCount]
mostPopular.sort()
mostPopular.reverse()

return1 = set()
count = 0
for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalRead*(65/100): break

for u5,b5 in np.array(testset.values):
    if b5 in return1:
        lst_pred1_q5.append(1)
    else:
        lst_pred1_q5.append(0)
```

In [166]:

```
readlst5 = df.groupby('userID')['bookID'].apply(list)
readlst5 = readlst5.to_dict()

userlst5 = df.groupby('bookID')['userID'].apply(list)
userlst5 = userlst5.to_dict()

def jaccard5(x):
    if x['bookID'] in list(df['bookID'].values):
        u = set(userlst5[x['bookID']])
    else:
        u = set()
    jaccards = [0]

    for b in readlst5[x['userID']]:
        j = len(set(userlst5[b]).intersection(u))/len(set(userls
t5[b]).union(u))
        jaccards.append(j)
    return max(jaccards)
```

In [167]:

```
jaccs5 = testset.apply(jaccard5, axis = 1)
```

In [168]:

```
lst_pred2_q5 = jaccs5.apply(lambda x: 1 if x > 0.01 else 0)
```

In [170]:

```
testpred = (lst_pred1_q5 + lst_pred2_q5) == 2
```

In [171]:

```
sum(testpred)
```

Out[171]:

```
8032
```

In [172]:

```
sum(lst_pred2_q5)
```

Out[172]:

```
14249
```

In [183]:

```
predictions = open("predictions_Read.txt", 'w')
i = 0
for l in open("pairs_Read.txt"):
    if l.startswith("userID"):
    #header
        predictions.write(l)
        continue
    u,b = l.strip().split('-')
    if testpred[i] == 1:
        predictions.write(u + '-' + b + ",1\n")
    else:
        predictions.write(u + '-' + b + ",0\n")
    i+=1

predictions.close()
```

Question6

In [184]:

```
def readjson(path):
    f = gzip.open(path,'rt')
    for l in f:
        yield eval(l)
```

In [185]:

```
data_traincate = list(readjson("train_Category.json.gz"))
```

In [186]:

```
train_c = data_traincate[:190000]
valid_c = data_traincate[190000:]
```

In [187]:

```
wordCount = defaultdict(int)
punctuation = set(string.punctuation)
for d in train_c:
    r = ''.join([c for c in d['review_text'].lower() if not c in
punctuation])
    for w in r.split():
        wordCount[w] += 1
```

In [188]:

```
counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

counts[:10]
```

Out[188]:

```
[(1421431, 'the'),
 (858920, 'and'),
 (754131, 'a'),
 (716863, 'to'),
 (699878, 'i'),
 (622767, 'of'),
 (420605, 'is'),
 (408519, 'in'),
 (392647, 'it'),
 (370481, 'this')]
```

In [189]:

```
words = [x[1] for x in counts[:1000]]
wordSet = set(words)
wordId = dict(zip(words, range(len(words))))
```

Question7

In [225]:

```
def feature(datum):
    feat = [0]*len(wordSet)
    t = datum['review_text']
    t = t.lower()
    t = [c for c in t if not (c in string.punctuation)]
    t = ''.join(t)
    words = t.strip().split()
    for w in words:
        if not (w in wordSet): continue
        feat[wordId[w]] += 1
    feat.append(1)
    return feat
```

In [226]:

```
X = [feature(d) for d in train_c]
y = [d['genreID'] for d in train_c]
```

In [227]:

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

In [228]:

```
clf = LogisticRegression()
clf.fit(X,y)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Speci
fy the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/s
vm/base.py:929: ConvergenceWarning: Liblinear failed
to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[228]:

```
LogisticRegression(C=1.0, class_weight=None, dual=Fa
lse, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=Non
e, max_iter=100,
                   multi_class='warn', n_jobs=None,
penalty='l2',
                   random_state=None, solver='warn',
tol=0.0001, verbose=0,
                   warm_start=False)
```

In [229]:

```
X_valid = [feature(d) for d in valid_c]
y_valid = [d['genreID'] for d in valid_c]
```

In [230]:

```
clf.score(X_valid,y_valid)
```

Out[230]:

```
0.6927692769276927
```

Question8

In [198]:

```
C_param_range = [0.3, 0.5, 0.7, 0.85, 0.9]
scorelst = []

for i in C_param_range:

    clf = LogisticRegression(C = i)
    clf.fit(X,y)

    scorelst.append(clf.score(X_valid,y_valid))
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Speci
fy the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/s
vm/base.py:929: ConvergenceWarning: Liblinear failed
to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Speci
fy the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Speci
fy the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/s
vm/base.py:929: ConvergenceWarning: Liblinear failed
to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Speci
fy the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Speci
fy the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

In [199]:

```
scorelst
```

Out[199]:

```
[0.6436643664366437,
 0.6435643564356436,
 0.6441644164416441,
 0.6434643464346435,
 0.6434643464346435]
```

So, we choose C=0.7, which gives the highest accuracy.

In [231]:

```
X = [feature(d) for d in train_c]
y = [d['genreID'] for d in train_c]

clf = LogisticRegression(C = 0.7)
clf.fit(X,y)

clf.score(X_valid,y_valid)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:432: FutureWarning: Default
solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/l
inear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Speci
fy the multi_class option to silence this warning.
  "this warning.", FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/s
vm/base.py:929: ConvergenceWarning: Liblinear failed
to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[231]:

```
0.6927692769276927
```

The accuracy improves to 0.6928

In [249]:

```
datatest_cate = list(readjson("test_Category.json.gz"))
```

In [247]:

```
fnew = [feature(d) for d in datatest_cate]
```

In [248]:

```
y_prednew = clf.predict(fnew)
```

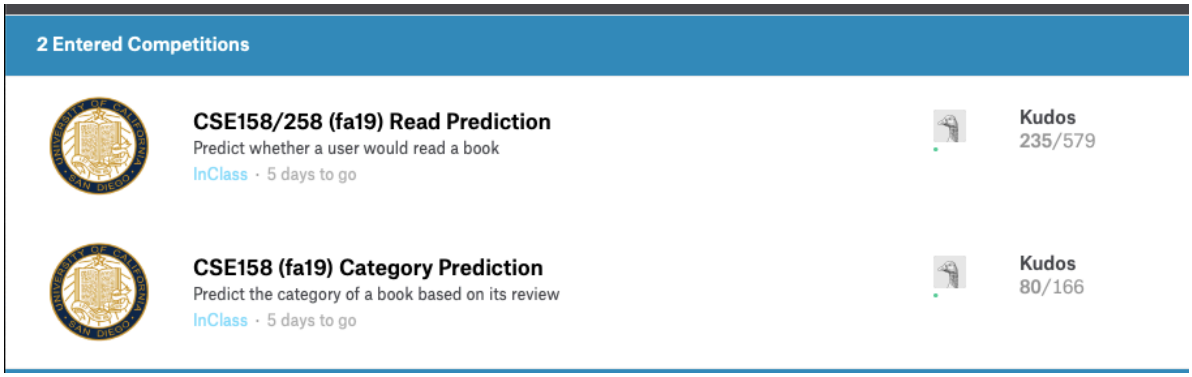
In [257]:

```
predictions = open("predictions_Category.txt", 'w')
predictions.write("userID-reviewID,prediction\n")

count = 0
for line in readGz('test_Category.json.gz'):
    predictions.write(line['user_id'] + '-' + line['review_id']
+ "," + str(y_prednew[count]) + "\n")
    count += 1

predictions.close()
```

In [260]:

```
from PIL import Image
myImage = Image.open("res.png");
myImage
```

Out[260]:



In [ ]:

```

```