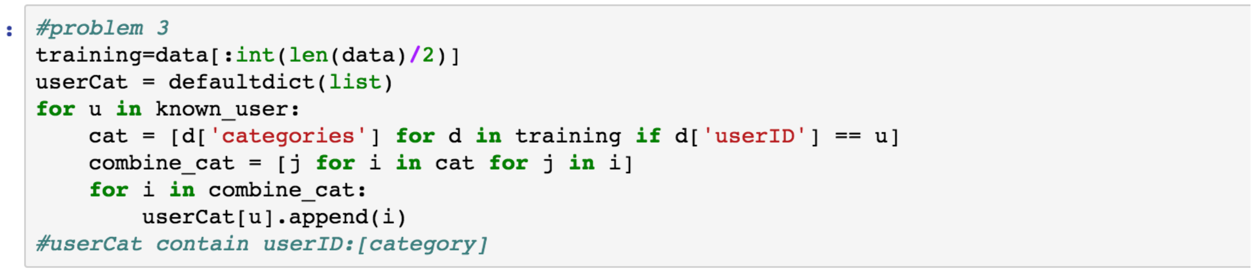
Problem1:

accuracy of the baseline model on the validation set is 62.6075 %

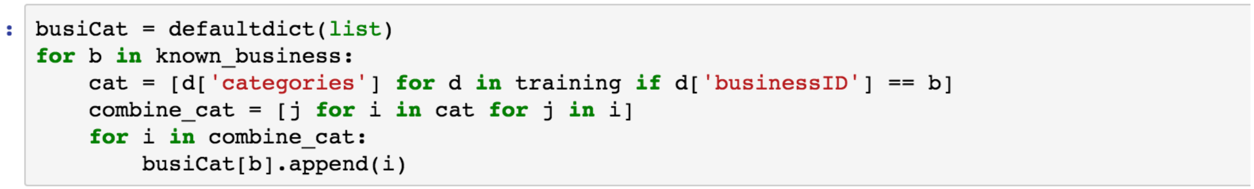
Problem2:

The threshold in problem1 is not the best. When I change the threshold from top 50% purchase to top 58% purchase, the accuracy changes to 62.924%

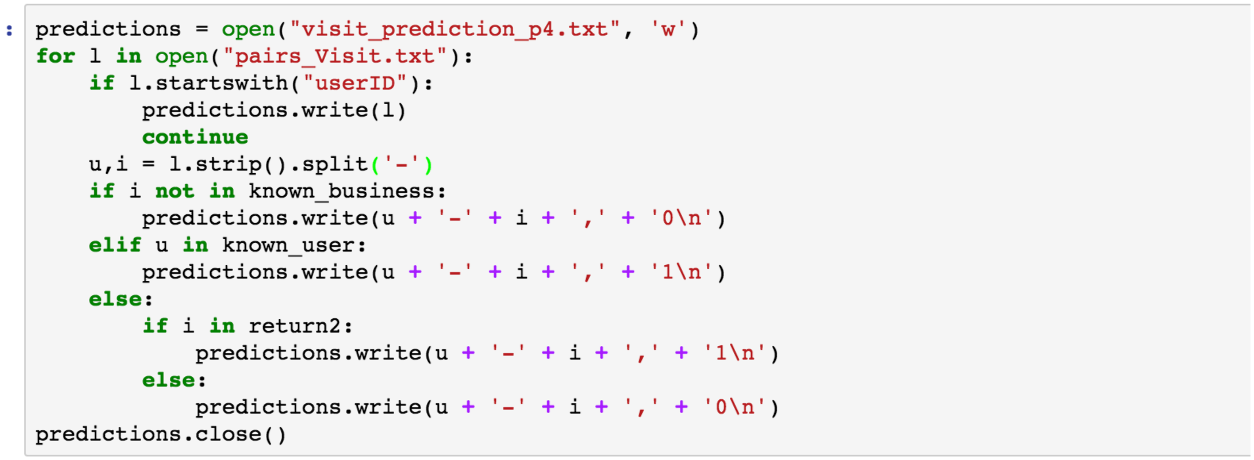
Problem3:



```
#problem 3
training=data[:int(len(data)/2)]
userCat = defaultdict(list)
for u in known_user:
    cat = [d['categories'] for d in training if d['userID'] == u]
    combine_cat = [j for i in cat for j in i]
    for i in combine_cat:
        userCat[u].append(i)
#userCat contain userID:[category]
```



```
busiCat = defaultdict(list)
for b in known_business:
    cat = [d['categories'] for d in training if d['businessID'] == b]
    combine_cat = [j for i in cat for j in i]
    for i in combine_cat:
        busiCat[b].append(i)
```



```
predictions = open("visit_prediction_p4.txt", 'w')
for l in open("pairs_Visit.txt"):
    if l.startswith("userID"):
        predictions.write(l)
        continue
    u,i = l.strip().split('-')
    if i not in known_business:
        predictions.write(u + '-' + i + ',' + '0\n')
    elif u in known_user:
        predictions.write(u + '-' + i + ',' + '1\n')
    else:
        if i in return2:
            predictions.write(u + '-' + i + ',' + '1\n')
        else:
            predictions.write(u + '-' + i + ',' + '0\n')
predictions.close()
```

Problem4

Name: thundering

Problem5

The trivial predictor I use is for users show in training set, alpha = user_average. For the users in validation set who doesn't show up in training set, alpha = global average. The MSE is 0.7163571192872857

The value for global average is 4.18703

Problem 6

For users that show up in the train data but their corresponding business didn't show up in the training data, I give Ru,i the value of average rating of that user in training data.

For users that didn’t show up in the train data but their corresponding business show up in the training data, I give Ru,i the average of that business in training data.

For users and their corresponding business both didn’t show up in training set, I give Ru,i the value of global average rating.

For convergence, the condition I use is the difference of MSE of training data in iteration step is smaller than 0.001

MSE for validation set is: 16.899844257640158

Problem 7

userID with largest value of beta is ('U357799541', 1.163195073560459)

userID with smallest value of beta is ('U417838537', -2.8340258769043882)

businessID with largest value of beta is ('B093985406', 1.169733638163109)

businessID with smallest value of beta is ('B241777680', -2.2339837734013415)

Problem8

Lambda = 7

MSE for validation set is: 16.7210142504965

Source code:

Task1

```
import gzip
from collections import defaultdict

def readGz(f):
    for l in gzip.open(f):
        yield eval(l)
```

```
print('reading data...')
data = list(readGz('train.json.gz'))
print('done')
```

```
reading data...
done
```

```
#problem1
import itertools
import random
known_ub_pair = [(d['userID'],d['businessID']) for d in data]  #get known user,business pair from original 200,000 trai
known_user = [d['userID'] for d in data]
known_business = [d['businessID'] for d in data]
known_user = list(set(known_user))
known_business = list(set(known_business))
```

```
count = 0
count_total = 100000
while True:
    index_u = random.randint(0,len(known_user)-1)
    index_b = random.randint(0,len(known_business)-1)
    sample = (known_user[index_u],known_business[index_b])
    if sample not in known_ub_pair:
        validation.append(sample)
        count += 1
    if count == count_total:
        break
```

```
#calculate accuracy of the baseline model on the validation set
y=[1]*100000+[0]*100000
businessCount = defaultdict(int)
totalPurchases = 0
for i in training:
    user,business = i[0],i[1]
    businessCount[business] += 1
    totalPurchases += 1
mostPopular = [(businessCount[x],x) for x in businessCount]
mostPopular.sort()
mostPopular.reverse()
return1 = set() #contain the business ID for the top 50% purchase
count = 0
for ic,i in mostPopular:
    count += ic
    return1.add(i)
    if count>totalPurchases/2:break
prediction = []
for i in range(len(validation)):
    if validation[i][1] in return1:
        prediction.append(1)
    else:
        prediction.append(0)
correct = [(a == b) for a,b in zip(y,prediction)]
accuracy = sum(correct)/len(correct)
print('accuracy of the baseline model on the validation set is',accuracy*100,' % ')
```

```
accuracy of the baseline model on the validation set is 62.6075  %
```

```
#problem 2
return2 = set() #contain the business ID for the top 58%
count = 0
for ic,i in mostPopular:
    count += ic
    return2.add(i)
    if count>totalPurchases*5.8/10:break
prediction2 = []
for i in range(len(validation)):
    if validation[i][1] in return2:
        prediction2.append(1)
    else:
        prediction2.append(0)
correct = [(a == b) for a,b in zip(y,prediction2)]
accuracy = sum(correct)/len(correct)
print('accuracy of the baseline model on the validation set is',accuracy*100,' % ')
```

```
accuracy of the baseline model on the validation set is 62.924  %
```

```
#problem 3
training=data[:int(len(data)/2)]
userCat = defaultdict(list)
for u in known_user:
    cat = [d['categories'] for d in training if d['userID'] == u]
    combine_cat = [j for i in cat for j in i]
    for i in combine_cat:
        userCat[u].append(i)
#userCat contain userID:[category]
```

```
busiCat = defaultdict(list)
for b in known_business:
    cat = [d['categories'] for d in training if d['businessID'] == b]
    combine_cat = [j for i in cat for j in i]
    for i in combine_cat:
        busiCat[b].append(i)
```

```
predictions = open("visit_prediction_p4.txt", 'w')
for l in open("pairs_Visit.txt"):
    if l.startswith("userID"):
        predictions.write(l)
        continue
    u,i = l.strip().split('-')
    if i not in known_business:
        predictions.write(u + '-' + i + ',' + '0\n')
    elif u in known_user:
        predictions.write(u + '-' + i + ',' + '1\n')
    else:
        if i in return2:
            predictions.write(u + '-' + i + ',' + '1\n')
        else:
            predictions.write(u + '-' + i + ',' + '0\n')
predictions.close()
```

task2

```
#problem 5
known_user_tr = [d['userID'] for d in data[:int(len(simplify_data)/2)]]
known_user_tr = list(set(known_user_tr))
```

```
simplify_data = [(d['userID'],d['businessID'],d['rating']) for d in data]
training_data = simplify_data[:int(len(simplify_data)/2)]
validation_data = simplify_data[int(len(simplify_data)/2):]

allRatings = []
userRating = defaultdict(list)
for i in training_data:
    allRatings.append(i[2])
    userRating[i[0]].append(i[2])
globalAverage = sum(allRatings)/len(allRatings)
userAverage = {}
for u in userRating:
    userAverage[u] = sum(userRating[u])/len(userRating[u])
MSE = 0
for d in validation_data:
    if d[0] in known_user_tr:
        MSE += (d[2]-userAverage[d[0]])**2
    else:
        MSE += (d[2]-globalAverage)**2
print('the globalaverage is ',globalAverage)
print('the MSE is ',MSE/len(validation_data))
```

```python
b_allRating = []
bRating = defaultdict(list)
for i in training_data:
    b_allRating.append(i[2])
    bRating[i[1]].append(i[2])
b_globalAverage = sum(b_allRating)/len(b_allRating)
b_average = {}
for b in bRating:
    b_average[b] = sum(bRating[b])/len(bRating[b])
print('global busines rating is ',b_globalAverage)
```

```python
#problem 6
from random import randint
simplify_data = [(d['userID'],d['businessID'],d['rating']) for d in data]
training_data = simplify_data[:int(len(simplify_data)/2)]
user_iden = [d[0] for d in training_data]
user_iden = list(set(user_iden))
business_iden = [d[1] for d in training_data]
business_iden = list(set(business_iden))
u_b_r = defaultdict(float)
user_business = defaultdict(list)
business_user = defaultdict(list)
for (u,b,r) in training_data:
    u_b_r[(u,b)] = r
    user_business[u].append(b)
    business_user[b].append(u)
for u in user_business:
    user_business[u] = list(set(user_business[u]))
for b in business_user:
    business_user[b] = list(set(business_user[b]))
beta_user = defaultdict(float)
beta_business = defaultdict(float)
alpha = randint(30,50)/10
for u in user_iden:
    beta_user[u] = randint(-20,20)/100
for b in business_iden:
    beta_business[b] = randint(-20,20)/100
lam =1
```

```python
#---------update-------------------
SE_old = 0
SE_new = 3
while(abs(SE_new-SE_old)>0.001):
    sum_alpha = 0
    for d in training_data:
        sum_alpha += u_b_r[(d[0],d[1])]-(beta_user[d[0]]+beta_business[d[1]])
    alpha = sum_alpha/len(training_data)

    for u in user_iden:
        update_beta_user = 0
        for b in user_business[u]:
            update_beta_user += (u_b_r[(u,b)]-(alpha+beta_business[b]))/(lam+len(user_business[u]))
        beta_user[u] = update_beta_user

    for b in business_iden:
        update_beta_business = 0
        for u in business_user[b]:
            update_beta_business += (u_b_r[(u,b)]-(alpha+beta_user[u]))/(lam+len(business_user[b]))
        beta_business[b] = update_beta_business
    SE = 0
    for i in training_data:
        SE +=(alpha+beta_user[i[0]]+beta_business[i[1]]-u_b_r[(i[0],i[1])])**2
    #print( ' SE = ',SE)
    SE_old = SE_new
    SE_new = SE
```

```
MSE = 0
for (u,b,r) in validation_data:
    if u in user_iden:
        if b in business_iden:
            MSE += (alpha+beta_user[u]+beta_business[b]-u_b_r[(u,b)])**2
        else:
            MSE += (alpha+beta_user[u]-userAverage[u])**2
    else:
        if b in business_iden:
            MSE += (alpha+beta_business[b]-b_average[b])**2
        else:
            MSE += (alpha-globalAverage)**2
print('MSE for validation set is: ',MSE/len(validation_data))
```

```
MSE for validation set is:  16.899844257640158
```

```
#problem7
from operator import itemgetter
b_beta_user=sorted(beta_user.items(),key=itemgetter(1),reverse = True)
print('userID with largest value of beta is ',b_beta_user[0])
s_beta_user=sorted(beta_user.items(),key=itemgetter(1))
print('userID with smallest value of beta is ',s_beta_user[0])
b_beta_business = sorted(beta_business.items(),key = itemgetter(1), reverse = True)
print('businessID with largest value of beta is ',b_beta_business[0])
s_beta_business = sorted(beta_business.items(),key = itemgetter(1))
print('businessID with smallest value of beta is',s_beta_business[0])
```

```
userID with largest value of beta is  ('U357799541', 1.163195073560459)
userID with smallest value of beta is  ('U417838537', -2.8340258769043882)
businessID with largest value of beta is  ('B093985406', 1.169733638163109)
businessID with smallest value of beta is ('B241777680', -2.2339837734013415)
```

```
#####problem 8
from random import randint
simplify_data = [(d['userID'],d['businessID'],d['rating']) for d in data]
training_data = simplify_data[:int(len(simplify_data)/2)]
user_iden = [d[0] for d in training_data]
user_iden = list(set(user_iden))
business_iden = [d[1] for d in training_data]
business_iden = list(set(business_iden))
u_b_r = defaultdict(float)
user_business = defaultdict(list)
business_user = defaultdict(list)
for (u,b,r) in training_data:
    u_b_r[(u,b)] = r
    user_business[u].append(b)
    business_user[b].append(u)
for u in user_business:
    user_business[u] = list(set(user_business[u]))
for b in business_user:
    business_user[b] = list(set(business_user[b]))
beta_user = defaultdict(float)
beta_business = defaultdict(float)
```

```python
alpha = randint(30,50)/10
for u in user_iden:
    beta_user[u] = randint(-20,20)/100
for b in business_iden:
    beta_business[b] = randint(-20,20)/100
lam =7
#---------update-------------------
SE_old = 0
SE_new = 3
while(abs(SE_new-SE_old)>0.001):
    sum_alpha = 0
    for d in training_data:
        sum_alpha += u_b_r[(d[0],d[1])]-(beta_user[d[0]]+beta_business[d[1]])
    alpha = sum_alpha/len(training_data)

    for u in user_iden:
        update_beta_user = 0
        for b in user_business[u]:
            update_beta_user += (u_b_r[(u,b)]-(alpha+beta_business[b]))/(lam+len(user_business[u]))
        beta_user[u] = update_beta_user

    for b in business_iden:
        update_beta_business = 0
        for u in business_user[b]:
            update_beta_business += (u_b_r[(u,b)]-(alpha+beta_user[u]))/(lam+len(business_user[b]))
        beta_business[b] = update_beta_business
    SE = 0
    for i in training_data:
        SE +=(alpha+beta_user[i[0]]+beta_business[i[1]]-u_b_r[(i[0],i[1])])**2

    SE_old = SE_new
    SE_new = SE
MSE = 0
for (u,b,r) in validation_data:
    if u in user_iden:
        if b in business_iden:
            MSE += (alpha+beta_user[u]+beta_business[b]-u_b_r[(u,b)])**2
        else:
            MSE += (alpha+beta_user[u]-userAverage[u])**2
    else:
        if b in business_iden:
            MSE += (alpha+beta_business[b]-b_average[b])**2
        else:
            MSE += (alpha-globalAverage)**2
print('MSE for validation set is: ',MSE/len(validation_data))
```

```python
predictions = open('prediction_Rating_q8.txt','w')
for l in open('pairs_Rating.txt'):
    if l.startswith('userID'):
        predictions.write(l)
        continue
    u,i = l.strip().split('-')
    if u in user_iden:
        if b in business_iden:
            rating = alpha+beta_user[u]+beta_user[b]
            predictions.write(u+'-'+i+','+str(rating)+'\n')
        else:
            rating = alpha+beta_user[u]
            predictions.write(u+'-'+i+','+str(rating)+'\n')
    else:
        if b in business_iden:
            rating = alpha+beta_business[b]
            predictions.write(u+'-'+i+','+str(rating)+'\n')
        else:
            rating = alpha
            predictions.write(u+'-'+i+','+str(rating)+'\n')
predictions.close()
```