**from** collections **import** defaultdict  
**import** numpy **as** np  
  
data = []  
count = 0;  
  
**for** l **in** readGz(**"train.json.gz"**):  
 data.append(l)  
  
trainData, validData = np.array\_split(data, 2)  
  
businessCount = defaultdict(int)  
visitedBusinesses = defaultdict(int)  
users = defaultdict(int)  
totalVisits = 0;  
  
**for** d **in** trainData:  
 user, business = d[**'userID'**], d[**'businessID'**]  
 businessCount[business] += 1  
 totalVisits += 1  
 visitedBusinesses[user + **'-'** + business] += 1  
 users[user] += 1  
  
print(len(businessCount))  
print(len(users))  
print(len(visitedBusinesses))  
  
mostPopular = [(businessCount[x], x) **for** x **in** businessCount]  
mostPopular.sort()  
mostPopular.reverse()  
  
return1 = set()  
count = 0  
**for** ic, i **in** mostPopular:  
 count += ic  
 return1.add(i)  
 **if** count > totalVisits / 2: **break**businessList = []  
usersList = []  
  
vbusinessCount = defaultdict(int)  
vvisitedBusinesses = defaultdict(int)  
vusers = defaultdict(int)  
vtotalVisits = 0;  
vnotvisited = defaultdict(int)  
  
**for** d **in** validData:  
 user, business = d[**'userID'**], d[**'businessID'**]  
 vbusinessCount[business] += 1  
 vtotalVisits += 1  
 vvisitedBusinesses[user + **'-'** + business] += 1  
 vusers[user] += 1  
  
**for** c **in** vbusinessCount:  
 businessList.append(c)  
**for** u **in** vusers:  
 usersList.append(u)  
  
**import** random  
  
**while** (len(vnotvisited) < 100000):  
 pair = random.choice(usersList) + **'-'** + random.choice(businessList)  
 **while** (pair **in** vvisitedBusinesses):  
 pair = random.choice(usersList) + **'-'** + random.choice(businessList)  
 vnotvisited[pair] += 1  
  
wrongPred = 0;  
correctPred = 0;  
  
  
**def** baselinePredict(userBusinessPair, visited):  
 **global** wrongPred  
 **global** correctPred  
 **for** p **in** userBusinessPair:  
 u, b = p.strip().split(**'-'**)  
 **if** ((b **in** return1 **and** visited) **or** (b **not in** return1 **and not** visited)):  
 correctPred += 1  
 **else**:  
 wrongPred += 1  
  
  
baselinePredict(vvisitedBusinesses, **True**)  
baselinePredict(vnotvisited, **False**)  
  
print(wrongPred / (wrongPred + correctPred) \* 100)

**Baseline prediction error:** 37.7345

|  |  |
| --- | --- |
| **Threshold** | **Error** |
| 1 | 49.303999999999995 |
| 0.9 | 43.4765 |
| 0.7 | 38.5635 |
| 0.8 | 40.426 |
| 0.4 | 38.6875 |
| 0.3 | 40.4025 |
| 0.2 | 42.933 |
| 0.1 | 46.077 |
| 0.6 | 37.606 |
| 0.65 | 37.915 |
| 0.55 | 37.4985 |

**Building if user visited same category before then true:**

Wrong: 69473

Right: 130527

Error rate: 0.347365

5. **Alpha =** 4.18703

**MSE for this model:** 0.7483437445

6. For model trained with betas and convergence wrt alpha val data with **alpha =** 4.224718130267249

**MSE for this model:** 0.645692830316

User U417838537 has least beta: -2.834463263979296

User U357799541 has max beta: 1.162953088597245

User B241777680 has least beta: -2.2356078402713204

User B093985406 has max beta: 0

**Lambda for min MSE =** 4.100000000000001

MSE : 0.607994393749