Task (Purchase Prediction)

In [1]:

```
import numpy
import urllib.request
import scipy.optimize
import random
from collections import defaultdict
import nltk
import string
import os
from nltk.stem.porter import *
from sklearn import linear_model
import matplotlib.pyplot as plt
```

Problem 1

In [2]:

```
def parseData(fname):
    for l in urllib.request.urlopen(fname):
        yield eval(l)
```

In [3]:

```
print("Reading data...")
data = list(parseData("file:train.json"))
print("done")
```

```
Reading data...
done
```

In [4]:

```
train_data = data[:100000]
valid_data = data[100000:]
```

In [5]:

```
allset = tuple([[data[i]['reviewerID'],data[i]['itemID']] for i in range(len
train_set = [[train_data[i]['reviewerID'],train_data[i]['itemID']] for i in
valid_set1 = [[valid_data[i]['reviewerID'],valid_data[i]['itemID']] for i in

# allset = ([[data[i]['reviewerID'], data[i]['itemID']] for i in range(len(d
# train_set = ([[train_data[i]['reviewerID'], train_data[i]['itemID']] for i
# valid_set1 = ([[valid_data[i]['reviewerID'], valid_data[i]['itemID']] for
```

In [10]:

```
valid_set2 = []
reviewerID = []
itemID = []
for l in data:
    reviewerID.append(l['reviewerID'])
    itemID.append(l['itemID'])
reviewers = list(set(reviewerID))
items = list(set(itemID))
```

In [11]:

```
i = 0
while i <= 100000:
    reviewerID = reviewers[random.randint(0, len(reviewers) - 1)]
    itemID = items[random.randint(0, len(items) - 1)]
    non_visited_pairs = [reviewerID, itemID]
    if tuple(non_visited_pairs) not in allset:
        valid_set2.append(non_visited_pairs)
        i += 1
```

In [12]:

```
itemCount = defaultdict(int)
totalPurchases = 0

for l in train_set:
    reviewer, item = l[0], l[1]
    itemCount[item] += 1
    totalPurchases += 1

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

In [25]:

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

In [26]:

```
pre_v = []
pre_unv = []
for i in range(len(train_set)):
    r = train_set[i][0]
    b = train_set[i][1]
    if i in return1:
        pre_v.append([r, b])
    else:
        pre_unv.append([r, b])
```

In [27]:

```
cnt = 0
for r, b in valid_set1:
    if b in return1:
        cnt += 1
for r, b in valid_set2:
    if b not in return1:
        cnt += 1
print("Performance/accuracy of the baseline model on the validation set is:
```

```
Performance/accuracy of the baseline model on the validation set is:0.629
02
```

Problem 2

In [18]:

```
percent = [0.3, 0.4, 0.45, 0.526, 0.527, 0.53, 0.6, 0.65, 0.7, 0.75, 0.8]

for p in percent:
    return1 = set()
    count = 0
    for ic, i in mostPopular:
        count += ic
        return1.add(i)
        if count > totalPurchases * p:
            break
    cnt = 0
    for r, i in valid_set1:
        if str(i) in return1:
            cnt += 1
    for r, i in valid_set2:
        if str(i) not in return1:
            cnt += 1
    print("percent at " + str(p) + ": accuracy is " + str(cnt/200000))
```

```
percent at 0.3: accuracy is 0.601625
percent at 0.4: accuracy is 0.620565
percent at 0.45: accuracy is 0.626465
percent at 0.526: accuracy is 0.630545
percent at 0.527: accuracy is 0.63054
percent at 0.53: accuracy is 0.63046
percent at 0.6: accuracy is 0.628415
percent at 0.65: accuracy is 0.621955
percent at 0.7: accuracy is 0.614135
percent at 0.75: accuracy is 0.60274
percent at 0.8: accuracy is 0.590885
```

In [76]:

```
# plt.plot(percent, threshold_accuracy)
# plt.xlabel("Values of different threshold percentiles")
# plt.ylabel("Accuracy measures")
# plt.show()
```

In [ ]:

A better value of accuracy is 0.630545, which occurs at 52.6 percentile.

Problem 3

In [22]:

```
reviewer_visited = defaultdict(list)
item_category = defaultdict(list)
reviewerID = []
itemID = []
for l in data:
    reviewer, item, category = l['reviewerID'], l['itemID'], l['categories'
    reviewerID.append(reviewer)
    itemID.append(item)
    reviewer_visited[reviewer].append(category)
    item_category[item] = category
```

In [23]:

```
pre_category = defaultdict(list)
for r, i in train_set:
    for c in item_category[i]:
        pre_category[r].append(c)
```

In [24]:

```
cnt = 0
for r, i in valid_set1:
    if sum([c in pre_category[r] for c in item_category[i]]) > 0:
        cnt += 1
for r, i in valid_set2:
    if sum([c in pre_category[r] for c in item_category[i]]) == 0:
        cnt += 1
print('accuracy is ' + str(cnt/200000))
```

```
accuracy is 0.59574
```

In [ ]:

Problem 4

In [25]:

```
reviewer_visited = defaultdict(list)
item_category = defaultdict(list)
reviewerID = []
itemID = []
for l in data:
    reviewer, item, category = l['reviewerID'], l['itemID'], l['categories'
    reviewerID.append(reviewer)
    itemID.append(item)
    reviewer_visited[reviewer].append(category)
    item_category[item] = category
```

In [26]:

```
pre_category = defaultdict(list)
for r, i in train_set:
    for c in item_category[i]:
        pre_category[r].append(c)
```

In [27]:

```
predictions = open("predictions_Purchase.txt", 'w')
for l in open("pairs_Purchase.txt"):
    if l.startswith("reviewerID"):
        predictions.write(l)
        continue
    reviewer, item = l.strip().split('-')
    if sum([c in pre_category[reviewer] for c in item_category[item]]) > 0:
        predictions.write(reviewer + '-' + item + ",1\n")
    if sum([c in pre_category[reviewer] for c in item_category[item]]) == 0
        predictions.write(reviewer + '-' + item + ",0\n")
predictions.close()
```

In [28]:

```
itemCount = defaultdict(int)
totalPurchases = 0

for l in train_set:
    reviewer,item = l[0],l[1]
    itemCount[item] += 1
    totalPurchases += 1
```

In [29]:

```
mostPupular = [(itemCount[x], x) for x in itemCount]
mostPopular.sort()
mostPopular.reverse()
```

In [42]:

```
return1 = set()
count = 0

for ic, i in mostPopular:
    count += ic
    return1.add(i)
    if count > totalPurchases* 0.526:
        break
```

In [43]:

```
predictions = open("predictions_Purchase.csv", 'w')
for l in open("pairs_Purchase.txt"):
    if l.startswith("reviewerID"):
        predictions.write(l)
        continue
    r, i = l.strip().split('-')
    if i in return1:
        predictions.write(r + '-' + i + ",1\n")
    else:
        predictions.write(r + '-' + i + ",0\n")
predictions.close()
```

Kaggle Name: Macchiato

In [ ]: