Mohamed Aturban (maturban)
CS595 — Fall 2013

Assignment 8

November 15, 2013

All files mentioned in this document should be uploaded into the *github* repository. Answers to all questions will be described into three sections: description, code lines, and output samples. The file *recommendations.py* contains the Python code for solving all 10 questions. start symbol (*), in the output tables, indicates that I choose the marked line as my result.

Problem 1

- To answer this question, we need to know how many ratings each movie has received and the summation of these ratings. After that, we can simply compute the average. The following piece of code will return movies that have the highest average ratings (I got 10 movies with average ratings of 5.0; the first five are marked by (*)):
- Source code

```
rating = {}
print (" ##### number [1] solution ##### ")
for user in prefs.keys():
  for key,value in prefs[user].iteritems():
              rating.setdefault(key,[])
              rating[key].append(value)
average = {}
for movie in rating.keys():
  avg = np.mean(rating[movie])
  average[movie] = avg
sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
for (key,value) in sorted_x[0:10]:
 print key,' ... ',value
```

- Output

```
     Movie                              Average-ratings
-------------------------------------------------------
Great Day in Harlem, A (1994)           5.0    (*)
Prefontaine (1997)                      5.0    (*)
Aiqing wansui (1994)                    5.0    (*)
Star Kid (1997)                         5.0    (*)
Marlene Dietrich: Shadow and Light (1996) 5.0  (*)
Entertaining Angels:
The Dorothy Day Story (1996)            5.0
Saint of Fort Washington, The (1993)    5.0
Someone Else's America (1995)           5.0
Santa with Muscles (1996)               5.0
```

```
They Made Me a Criminal (1939)           5.0
```

Problem 2

- Here, we are considering only number of ratings each movie has received, so I simply modified the code from question 1 to answer this question:
- Source code

```
rating = {}
print (" ##### number [2] solution ##### ")
for user in prefs.keys():
  for key,value in prefs[user].iteritems():
             rating.setdefault(key,[])
             rating[key].append(value)
lengthList = {}
for movie in rating.keys():
  lenlist = len(rating[movie])
  lengthList[movie] = lenlist
sorted_x = sorted(lengthList.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
for (key,value) in sorted_x[0:5]:
 print key,' ... ',value
```

- Output

```
    Movie                       Number of ratings
-------------------------------------------------------
Star Wars (1977)                   583
Contact (1997)                     509
Fargo (1996)                       508
Return of the Jedi (1983)          507
Liar Liar (1997)                   485
```

Problem 3

- This question requires loading more information that was not originally retrieved; I am talking about the genders and ages (ages will be sed in question 9 and 10). After loading, I only added *if* statement to the source code, from question 1, to check whether a user is male (M) or Female (F) (I got 11 movies with average ratings 5.0; the first five are marked by (*)):
- Source code

```
# Load more data: gender and age
gender={}
age={}
for line in open('u.user'):
  (id,ag,g,x,y) = line.split('|')
  gender.setdefault(id,g)
  age.setdefault(id,ag)
 ...
 ...
rating = {}
for user in prefs.keys():
  if gender[user] == 'M':
    continue
  for key,value in prefs[user].iteritems():
              rating.setdefault(key,[])
              rating[key].append(value)
average = {}
for movie in rating.keys():
  avg = np.mean(rating[movie])
  average[movie] = avg

sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print (" ##### number [3] solution ##### ")
for (key,value) in sorted_x[0:15]:
 print key,' ... ',value
```

- Output

```
     Movie                        Average ratings(Women)
------------------------------------------------------------
Prefontaine (1997)                        5.0         (*)
Telling Lies in America (1997)            5.0         (*)
Foreign Correspondent (1940)              5.0         (*)
Faster Pussycat! Kill! Kill! (1965)       5.0         (*)
Year of the Horse (1997)                  5.0         (*)
Mina Tannenbaum (1994)                    5.0
Maya Lin: A Strong Clear Vision (1994)  5.0
Everest (1998)                            5.0
Someone Else's America (1995)             5.0
Visitors, The (Visiteurs, Les) (1993)   5.0
Stripes (1981)                            5.0
```

Problem 4

- One small change to the previous question source code is sufficient to answer this question –changing 'M'(Male) to 'F' (Female)(I got 15 movies with average ratings of 5.0; the first five are marked by (*):
- Source code

```
rating = {}
for user in prefs.keys():
  if gender[user] == 'F':
    continue
  for key,value in prefs[user].iteritems():
              rating.setdefault(key,[])
              rating[key].append(value)
average = {}
for movie in rating.keys():
  avg = np.mean(rating[movie])
  average[movie] = avg

sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print (" ##### number [4] solution ##### ")
for (key,value) in sorted_x[0:15]:
 print key,' ... ',value
```

- Output

```
     Movie                          Average ratings(Men)
------------------------------------------------------------
Delta of Venus (1994)                          5.0   (*)
Great Day in Harlem, A (1994)                  5.0   (*)
Leading Man, The (1996)                        5.0   (*)
Love Serenade (1996)                           5.0   (*)
Prefontaine (1997)                             5.0   (*)
Aiqing wansui (1994)                           5.0
Little City (1998)                             5.0
Star Kid (1997)                                5.0
Marlene Dietrich: Shadow and Light (1996)      5.0
Entertaining Angels:
The Dorothy Day Story (1996)                   5.0
Quiet Room, The (1996)                         5.0
Saint of Fort Washington, The (1993)           5.0
Letter From Death Row, A (1998)                5.0
Santa with Muscles (1996)                      5.0
They Made Me a Criminal (1939)                 5.0
```

Problem 5

- the function *topMatches()* is considered to answer this question. The function returns the best *n* matches for person (or movies) , but before calling the function, we should swap movies and raters within *prefs* dictionary using the function it transformPrefs(). I just made a copy of the function body inside my code:
- Source code

```
result={}
for person in prefs:
  for item in prefs[person]:
    result.setdefault(item,{})
    # Flip item and person
    result[item][person]=prefs[person][item]
print (" ##### number [5] solution ##### ")
print('Top matches : ',topMatches(result,'Top Gun (1986)')[0:8])
print('Least matches : ',topMatches(result,'Top Gun (1986)')[-31:])
```

- Output

```
     Movie(*) received ratings most like Top Gun
-------------------------------------------------------
     Movie                                       r
-------------------------------------------------------
Wild America (1997)                             1.0   (*)
Wedding Gift, The (1994)                        1.0
Underground (1995)                              1.0
Two or Three Things I Know About Her (1966)     1.0
Two Bits (1995)                                 1.0
Total Eclipse (1995)                            1.0
The Innocent (1994)                             1.0
That Old Feeling (1997)                         1.0



     Movie(*) received ratings least like Top Gun
-------------------------------------------------------
     Movie                                       r
-------------------------------------------------------
Year of the Horse (1997)                        -1.0   (*)
World of Apu, The (Apur Sansar) (1959)          -1.0
Two Much (1996)                                 -1.0
Tetsuo II: Body Hammer (1992)                   -1.0
Telling Lies in America (1997)                  -1.0
Switchback (1997)                               -1.0
```

```
Safe Passage (1994)                             -1.0
Roseanna's Grave (For Roseanna) (1997)          -1.0
Romper Stomper (1992)                           -1.0
Nil By Mouth (1997)                             -1.0
Nico Icon (1995)                                -1.0
Naked in New York (1994)                        -1.0
Midnight Dancers (Sibak) (1994)                 -1.0
Meet Wally Sparks (1997)                        -1.0
Lover's Knot (1996)                             -1.0
Love and Death on Long Island (1997)            -1.0
Loch Ness (1995)                                -1.0
Lamerica (1994)                                 -1.0
Joy Luck Club, The (1993)                       -1.0
Heidi Fleiss: Hollywood Madam (1995)            -1.0
Frisk (1995)                                    -1.0
Everest (1998)                                  -1.0
Carried Away (1996)                             -1.0
Carpool (1996)                                  -1.0
Caro Diario(Dear Diary) (1994)                  -1.0
Broken English (1996)                           -1.0
Bitter Sugar (Azucar Amargo) (1996)             -1.0
Bewegte Mann, Der (1994)                        -1.0
Beat the Devil (1954)                           -1.0
Bad Moon (1996)                                 -1.0
Babysitter, The (1995)                          -1.0
```

Problem 6

- Simply, using the function *len()* is sufficient to return number of movies rated by a specific user in the dictionary *prefs*:
- Source code

```
userMostRating = {}
for user in prefs.keys():
   userMostRating.setdefault(user,len(prefs[user]))

sorted_x = sorted(userMostRating.iteritems()
                 ,key=operator.itemgetter(1))
sorted_x.reverse()
print (" ##### number [6] solution ##### ")
for (key,value) in sorted_x[0:5]:
 print key,' ... ',value
```

- Output

```
             5 raters rated the most films
-------------------------------------------------------
    ID                             Number of movies
-------------------------------------------------------
   405                                 736
   655                                 678
   13                                  632
   450                                 538
   276                                 516
```

Problem 7

- By computing r for everyone to everyone, I got 806 raters whose more than 4 raters having r equal to 1. Please, see the file *q7.txt* for more detail. it has all groups of raters which are similar to each other ($r = 1$). For example, in the forth line, you will find:

  `( 343, 729, 341, 273, and 261 )`

  Which means that any pair of users from this list, with ids 343, 729, 341, 273, and 261, has an r value equal 1.0.

- Source code

```
print (" ##### number [7] solution ##### ")
lis = []
allsim = calculateSimilarUser(prefs,1000)
c = 0
k = 0
lessThanFive = 0
equalZero = 0
grearterThanFive = 0
Max = 0
for user in allsim.keys():
 k = k + 1
 totalError = 0.0
 t = 0
 for (key,value) in allsim[user]:#[0:4]:
  totalError = totalError + (1.0 - float(key))
  if(float(key) == 1.0):
    t = t + 1
 lis.append(t)
 if(t == 0):
  equalZero = equalZero + 1
 if(t >= 4):
  grearterThanFive = grearterThanFive + 1
 if(t < 4 )&( t > 0):
  lessThanFive = lessThanFive + 1
 if(t > Max):
  Max = t

print('Max = ',Max,'k=',k,' total = ',(equalZero + lessThanFive +
    grearterThanFive))
print('equal zero = ',equalZero,'lessThanFive',lessThanFive,
                  'grearterThanFive',grearterThanFive)
counter = 0
f = open('q7.txt', 'w')
for user in allsim.keys():
  if (lis[counter] > 3):
    s = '( '+user
    f.write(str(s))
    for (key,value) in allsim[user][0:(lis[counter])]:
      f.write(' + '+str(value))
    f.write(' )\n')
  counter = counter + 1
f.close()
```

- Output

```
              5 raters most agreed with each other
    -------------------------------------------------------
The are as many as 806 groups of raters which are most agreed with each
    other, see q7.txt. I have chosen the following group whose ids are:
    343, 729, 341, 273, and 261 (all r values are equal to 1.0).

In the file, q7.txt, you can also find, for example, another group
    which consists of 96 raters! and all agreed with each other (r =
    1.0 between any pair of raters)
```

Problem 8

- By computing r for everyone to everyone, I got 766 raters whose more than 4 raters having r equal to -1.0. Please, see the file *q8.txt* for more detail. it has all groups of raters which are not similar to each other (r = -1.0). For example, in the seventeenth line, you will find:

```
( 715, 29, 520, 651, and 858 )
```

  Which means that any pair of users from this list, with ids 715, 29, 520, 651, and 858, has an r value of -1.0.

- Source code

```
print (" ##### number [8] solution ##### ")
lis = []
allsim = calculateSimilarUser2(prefs,1000)
c = 0
k = 0
lessThanFive = 0
equalZero = 0
grearterThanFive = 0
Max = 0
for user in allsim.keys():
 k = k + 1
 totalError = 0.0
 t = 0
 for (key,value) in allsim[user]:
  totalError = totalError + (1.0 - float(key))
  if(float(key) == (-1.0)):
    t = t + 1
 lis.append(t)
 if(t == 0):
  equalZero = equalZero + 1
 if(t >= 4):
  grearterThanFive = grearterThanFive + 1
 if(t < 4 )&( t > 0):
  lessThanFive = lessThanFive + 1
 if(t > Max):
  Max = t

print('Max = ',Max,'k=',k,' total = ',(equalZero + lessThanFive +
    grearterThanFive))
print('equal zero =
    ',equalZero,'lessThanFive',lessThanFive,'grearterThanFive',grearterThanFive)
counter = 0
f = open('q8.txt', 'w')
for user in allsim.keys():
  if (lis[counter] > 3):
    s = '( '+user
    f.write(str(s))
    for (key,value) in allsim[user][0:(lis[counter])]:
      f.write(' + '+str(value))
    f.write(' )\n')
  counter = counter + 1
f.close()
```

- Output

```
                  5 raters most disagreed with each other
       -------------------------------------------------------
     The are as many as 766 groups of raters which are most disagreed with
        each other, see q8.txt. I have chosen the following group whose ids
        are: 715, 29, 520, 651, and 858 (all r values are equal to -1.0).

     In the file, q8.txt, you can also find, for example, another group
        which consists of 76 raters! and all disagreed with each other (r =
        -1.0 between any pair of rates)
```

Problem 9

- I only made a small change to the source code of question 3 which is a new *if* statement to check the age:
- Source code

```
rating = {}
for user in prefs.keys():
  if gender[user] == 'F':
    continue
  if int(age[user]) <= 40:
    continue
  for key,value in prefs[user].iteritems():
            rating.setdefault(key,[])
            rating[key].append(value)
average = {}
for movie in rating.keys():
  avg = np.mean(rating[movie])
  average[movie] = avg

sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print (" ##### number [9] solution ##### ")
for (key,value) in sorted_x[0:19]:
 print key,' ... ',value
```

- Output

```
     Movie                 Average-ratings(by men over 40)
------------------------------------------------------------
Hearts and Minds (1996)                           5.0  (*)
Faithful (1996)                                   5.0
Marlene Dietrich: Shadow and Light (1996)         5.0
Strawberry and Chocolate (Fresa y chocolate) (1993) 5.0
```

```
Late Bloomers (1996)                                  5.0
Solo (1996)                                           5.0
Grateful Dead (1995)                                  5.0
Prefontaine (1997)                                    5.0
Rendezvous in Paris (Rendez-vous de Paris, Les) (1995) 5.0
World of Apu, The (Apur Sansar) (1959)                5.0
Aparajito (1956)                                      5.0
Ace Ventura: When Nature Calls (1995)                 5.0
Star Kid (1997)                                       5.0
Two or Three Things I Know About Her (1966)           5.0
Poison Ivy II (1995)                                  5.0
Double Happiness (1994)                               5.0
Little City (1998)                                    5.0
Boxing Helena (1993)                                  5.0
Spice World (1997)                                    5.0
They Made Me a Criminal (1939)                        5.0
Great Day in Harlem, A (1994)                         5.0
Little Princess, The (1939)                           5.0
Unstrung Heroes (1995)                                5.0
Leading Man, The (1996)                               5.0
Indian Summer (1996)                                  5.0
```

Problem 10

- I only made a small change to the source code of question 9 –changing 'F' to 'M':
- Source code

```
rating = {}
for user in prefs.keys():
  if gender[user] == 'M':
    continue
  if int(age[user]) <= 40:
    continue
  for key,value in prefs[user].iteritems():
             rating.setdefault(key,[])
             rating[key].append(value)
average = {}
for movie in rating.keys():
  avg = np.mean(rating[movie])
  average[movie] = avg

sorted_x = sorted(average.iteritems(), key=operator.itemgetter(1))
sorted_x.reverse()
print (" ##### number [10] solution ##### ")
```

```
for (key,value) in sorted_x[0:19]:
 print key,' ... ',value
```

- Output

```
    Movie                Average-ratings(by women over 40)
------------------------------------------------------------
Shallow Grave (1994)                                 5.0  (*)
Great Dictator, The (1940)                           5.0
Visitors, The (Visiteurs, Les) (1993)                5.0
Shall We Dance? (1937)                               5.0
In the Bleak Midwinter (1995)                        5.0
Funny Face (1957)                                    5.0
Ma vie en rose (My Life in Pink) (1997)              5.0
Swept from the Sea (1997)                            5.0
Best Men (1997)                                      5.0
Foreign Correspondent (1940)                         5.0
Tombstone (1993)                                     5.0
Wrong Trousers, The (1993)                           5.0
Top Hat (1935)                                       5.0
Quest, The (1996)                                    5.0
Balto (1995)                                         5.0
Angel Baby (1995)                                    5.0
Band Wagon, The (1953)                               5.0
Letter From Death Row, A (1998)                      5.0
Mina Tannenbaum (1994)                               5.0
Mary Shelley's Frankenstein (1994)                   5.0
Gold Diggers: The Secret of Bear Mountain (1995)     5.0
Nightmare Before Christmas, The (1993)               5.0
Grand Day Out, A (1992)                              5.0
Bride of Frankenstein (1935)                         5.0
Pocahontas (1995)                                    5.0
```

References

[1] On-line GroupLens Research, `http://grouplens.org/datasets/movielens/`.

[2] J. Venge, On-line Stackoverflow, `http://stackoverflow.com/questions/783897/truncating-floats-in-python`.