

# CS595 Assignment 8

Jon Robison

November 14, 2013

See Appendix A for changes to recommendations.py  
See Appendix B for driver.py

1. What 5 movies have the highest average ratings? Show the movies and their ratings sorted by their average ratings.

Santa with Muscles (1996)	5.0
Great Day in Harlem, A (1994)	5.0
Prefontaine (1997)	5.0
Aiqing wansui (1994)	5.0
Marlene Dietrich: Shadow and Light (1996)	5.0
Entertaining Angels: The Dorothy Day Story (1996)	5.0
Star Kid (1997)	5.0
Saint of Fort Washington, The (1993)	5.0
Someone Else's America (1995)	5.0
They Made Me a Criminal (1939)	5.0

2. What 5 movies received the most ratings? Show the movies and the number of ratings sorted by number of ratings.

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

3. What 5 movies were rated the highest on average by women? Show

the movies and their ratings sorted by ratings.

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

4. What 5 movies were rated the highest on average by men? Show the movies and their ratings sorted by ratings.

Santa with Muscles (1996)	5.0
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
Marlene Dietrich: Shadow and Light (1996)	5.0
Entertaining Angels: The Dorothy Day Story (1996)	5.0
Star Kid (1997)	5.0
Quiet Room, The (1996)	5.0
Saint of Fort Washington, The (1993)	5.0
Letter From Death Row, A (1998)	5.0
They Made Me a Criminal (1939)	5.0

5. What movie received ratings most like Top Gun? Which movie received ratings that were least like Top Gun (negative correlation)? This bit was interesting, as I had some floating point errors. There were several returned with 1, but the rounding error must mean something, thus I left it.

Most similar ot Top Gun: 1.0000000000000027, Shiloh (1997)

Least similar ot Top Gun: -1.0000000000000007, Babysitter, The (1995)

6. Which 5 raters rated the most films? Show the raters' IDs and the number of films each rated.

276	516
450	538
13	632
655	678
405	736

7. Which 5 raters most agreed with each other? Show the raters' IDs and Pearson's  $r$ , sorted by  $r$ .

341	143	1.0
810	135	1.0
813	756	1.0
349	46	1.0
889	772	1.0

8. Which 5 raters most disagreed with each other (negative correlation)? Show the raters' IDs and Pearson's  $r$ , sorted by  $r$ .

60	872	-1.0
793	412	-1.0
344	598	-1.0
340	203	-1.0
340	68	-1.0

9. What movie was rated highest on average by men over 40? By men under 40?

Over 40:	
Hearts and Minds (1996)	5.0
Faithful (1996)	5.0
Marlene Dietrich: Shadow and Light (1996)	5.0
Grateful Dead (1995)	5.0
Prefontaine (1997)	5.0
World of Apu, The (Apu Sansar) (1959)	5.0
Rendezvous in Paris (Rendez-vous de Paris, Les) (1995)	5.0
Ace Ventura: When Nature Calls (1995)	5.0
Solo (1996)	5.0
Aparajito (1956)	5.0
Double Happiness (1994)	5.0
Little Princess, The (1939)	5.0
Strawberry and Chocolate (Fresa y chocolate) (1993)	5.0
Late Bloomers (1996)	5.0
Leading Man, The (1996)	5.0
Great Day in Harlem, A (1994)	5.0
Little City (1998)	5.0
Star Kid (1997)	5.0
They Made Me a Criminal (1939)	5.0
Poison Ivy II (1995)	5.0
Boxing Helena (1993)	5.0
Unstrung Heroes (1995)	5.0
Two or Three Things I Know About Her (1966)	5.0
Indian Summer (1996)	5.0
Spice World (1997)	5.0

Under 40:	
Delta of Venus (1994)	5.0
Santa with Muscles (1996)	5.0
Crossfire (1947)	5.0
Leading Man, The (1996)	5.0
Love Serenade (1996)	5.0
Prefontaine (1997)	5.0
Aiqing wansui (1994)	5.0
Love in the Afternoon (1957)	5.0
Angel Baby (1995)	5.0
Maya Lin: A Strong Clear Vision (1994)	5.0
Entertaining Angels: The Dorothy Day Story (1996)	5.0
Star Kid (1997)	5.0
Magic Hour, The (1998)	5.0
Quiet Room, The (1996)	5.0
Saint of Fort Washington, The (1993)	5.0
Perfect Candidate, A (1996)	5.0
Letter From Death Row, A (1998)	5.0

10. What movie was rated highest on average by women over 40? By women under 40?

Over 40:	
Shallow Grave (1994)	5.0
Shall We Dance? (1937)	5.0
In the Bleak Midwinter (1995)	5.0
Visitors, The (Visiteurs, Les) (1993)	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
Balto (1995)	5.0
Wrong Trousers, The (1993)	5.0
Funny Face (1957)	5.0
Quest, The (1996)	5.0
Tombstone (1993)	5.0
Bride of Frankenstein (1935)	5.0
Angel Baby (1995)	5.0
Great Dictator, The (1940)	5.0
Band Wagon, The (1953)	5.0
Letter From Death Row, A (1998)	5.0
Mina Tannenbaum (1994)	5.0
Top Hat (1935)	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
Pocahontas (1995)	5.0
Safe (1995)	5.0

Under 40:	
Backbeat (1993)	5.0
Prefontaine (1997)	5.0
Telling Lies in America (1997)	5.0
Year of the Horse (1997)	5.0
Mina Tannenbaum (1994)	5.0
Maya Lin: A Strong Clear Vision (1994)	5.0
Nico Icon (1995)	5.0
Umbrellas of Cherbourg, The (Parapluies de Cherbourg, Les) (1964)	5.0
Everest (1998)	5.0
Heaven's Prisoners (1996)	5.0
Wedding Gift, The (1994)	5.0
Faster Pussycat! Kill! Kill! (1965)	5.0
Horseman on the Roof, The (Hussard sur le toit, Le) (1995)	5.0
Grace of My Heart (1996)	5.0
Someone Else's America (1995)	5.0
Don't Be a Menace to South Central While Drinking Your Juice in the Hood (1996)	5.0
Stripes (1981)	5.0

## Appendix A

```
#!/usr/bin/python2
# A dictionary of movie critics and their ratings of a small
# set of movies
from math import sqrt
import operator

critics={'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5,
    'Just My Luck': 3.0, 'Superman Returns': 3.5, 'You, Me and Dupree': 2.5,
    'The Night Listener': 3.0},
    'Gene Seymour': {'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5,
    'Just My Luck': 1.5, 'Superman Returns': 5.0, 'The Night Listener': 3.0,
    'You, Me and Dupree': 3.5},
    'Michael Phillips': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.0,
    'Superman Returns': 3.5, 'The Night Listener': 4.0},
    'Claudia Puig': {'Snakes on a Plane': 3.5, 'Just My Luck': 3.0,
    'The Night Listener': 4.5, 'Superman Returns': 4.0,
    'You, Me and Dupree': 2.5},
    'Mick LaSalle': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
    'Just My Luck': 2.0, 'Superman Returns': 3.0, 'The Night Listener': 3.0,
    'You, Me and Dupree': 2.0},
    'Jack Matthews': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
    'The Night Listener': 3.0, 'Superman Returns': 5.0, 'You, Me and Dupree': 3.0},
    'Toby': {'Snakes on a Plane': 4.5, 'You, Me and Dupree': 1.0, 'Superman Returns': 3.0}}
users={}

# Returns a distance-based similarity score for person1 and person2
def sim_distance(prefs, person1, person2):
    # Get the list of shared_items
    si={}
    for item in prefs[person1]:
        if item in prefs[person2]: si[item]=1

    # if they have no ratings in common, return 0
    if len(si)==0: return 0

    # Add up the squares of all the differences
    sum_of_squares=sum([pow(prefs[person1][item]-prefs[person2][item],2)
        for item in prefs[person1] if item in prefs[person2]])

    return 1/(1+sum_of_squares)

# Returns the Pearson correlation coefficient for p1 and p2
def sim_pearson(prefs, p1, p2):
    # Get the list of mutually rated items
```



```

si={}
for item in prefs[p1]:
    if item in prefs[p2]: si[item]=1

# if they are no ratings in common, return 0
if len(si)==0: return 0

# Sum calculations
n=len(si)

# Sums of all the preferences
sum1=sum([prefs[p1][it] for it in si])
sum2=sum([prefs[p2][it] for it in si])

# Sums of the squares
sum1Sq=sum([pow(prefs[p1][it],2) for it in si])
sum2Sq=sum([pow(prefs[p2][it],2) for it in si])

# Sum of the products
pSum=sum([prefs[p1][it]*prefs[p2][it] for it in si])

# Calculate r (Pearson score)
num=pSum-(sum1*sum2/n)
den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
if den==0: return 0

r=num/den

return r

# Returns the best matches for person from the prefs dictionary.
# Number of results and similarity function are optional params.
def topMatches(prefs, person, n=5, similarity=sim_pearson):
    scores=[(similarity(prefs, person, other), other)
             for other in prefs if other!=person]

    scores.sort()
    scores.reverse()
    return scores[0:n]

# Gets recommendations for a person by using a weighted average
# of every other user's rankings
def getRecommendations(prefs, person, similarity=sim_pearson):
    totals={}
    simSums={}
    for other in prefs:
        # don't compare me to myself

```

```

    if other==person: continue
    sim=similarity(prefs, person, other)

    # ignore scores of zero or lower
    if sim<=0: continue
    for item in prefs[other]:

        # only score movies I haven't seen yet
        if item not in prefs[person] or prefs[person][item]==0:
            # Similarity * Score
            totals.setdefault(item,0)
            totals[item]+=prefs[other][item]*sim
            # Sum of similarities
            simSums.setdefault(item,0)
            simSums[item]+=sim

    # Create the normalized list
    rankings=[(total/simSums[item], item) for item, total in totals.items()]

    # Return the sorted list
    rankings.sort()
    rankings.reverse()
    return rankings

def transformPrefs(prefs):
    result={}
    for person in prefs:
        for item in prefs[person]:
            result.setdefault(item, {})

            # Flip item and person
            result[item][person]=prefs[person][item]
    return result

def calculateSimilarItems(prefs, n=10):
    # Create a dictionary of items showing which other items they
    # are most similar to.
    result={}
    # Invert the preference matrix to be item-centric
    itemPrefs=transformPrefs(prefs)
    c=0
    for item in itemPrefs:
        # Status updates for large datasets
        c+=1
        if c%100==0: print "%d / %d" % (c, len(itemPrefs))

```

```

        # Find the most similar items to this one
        scores=topMatches(itemPrefs,item,n=n,similarity=sim_distance)
        result[item]=scores
    return result

def getRecommendedItems(prefs,itemMatch,user):
    userRatings=prefs[user]
    scores={}
    totalSim={}
    # Loop over items rated by this user
    for (item,rating) in userRatings.items( ):

        # Loop over items similar to this one
        for (similarity,item2) in itemMatch[item]:

            # Ignore if this user has already rated this item
            if item2 in userRatings: continue
            # Weighted sum of rating times similarity
            scores.setdefault(item2,0)
            scores[item2]+=similarity*rating
            # Sum of all the similarities
            totalSim.setdefault(item2,0)
            totalSim[item2]+=similarity

    # Divide each total score by total weighting to get an average
    rankings=[(score/totalSim[item],item) for item,score in scores.items( )]

    # Return the rankings from highest to lowest
    rankings.sort( )
    rankings.reverse( )
    return rankings

def loadMovieLens(path='.'):
    global users

    # Get movie titles
    movies={}
    for line in open(path+'/u.item'):
        (id,title)=line.split('|')[0:2]
        movies[id]=title

    # Load data
    prefs={}
    for line in open(path+'/u.data'):
        (user,movieid,rating,ts)=line.split('\t')
        prefs.setdefault(user,{})

```

```

        prefs[user][movies[movieid]]=float(rating)

    for line in open(path+'/u.user'):
        (user,age,gender,tite,zip)=line.split('|')
        users[user]=(user,age,gender,tite,zip)
    return prefs

# Return dictionary of movies with all their scores
#Gender can be A for all, M for men, F for female
#Age can be A for all, U for under 40, O for over 40
def getMovieRatings(prefs, gender='A', age='A'):
    global users
    movies={}
    for user in prefs.keys():
        ugender=users[user][2]
        uage=int(users[user][1])
        isGender=(gender=='A' or (ugender is 'F' and gender is 'F') or
                  (ugender is 'M' and gender is 'M'))
        isAge=(age=='A' or (uage < 40 and age=='U') or
               (uage > 40 and age=='O'))
        if isGender and isAge:
            for movie in prefs[user].keys():
                if not movie in movies:
                    movies[movie]=[]
                movies[movie].append(prefs[user][movie])
    return movies

# Get movies sorted by their average score
def getMoviesAverageScore(prefs, gender='A', age='A'):
    movies=getMovieRatings(prefs, gender, age)
    for movie in movies.keys():
        score=0
        for rating in movies[movie]:
            score += rating
        movies[movie]=score/len(movies[movie])
    return sorted(movies.items(), key=lambda x: x[1])

#Return last value in a tuple
def getItemScore(item):
    return item[len(item)-1]

def getItemScoreReverse(top,x=1):
    return getItemScore(top[len(top)-x])

#Given the sorted list (prefs), return top 5
def getTop(sortedMovies,n=5):

```

```

top=[]
length=len(sortedMovies)
i=0
while i < n or getItemScoreReverse(top) == getItemScore(sortedMovies[length-1-i])
    top.append(sortedMovies[length-1-i])
    i+=1
return top

def getTopMovies(prefs, gender='A', age='A'):
    return getTop(getMoviesAverageScore(prefs, gender, age))

#Return folks with most ratings
def getTopRaters(prefs):
    raters={}
    for user in prefs.keys():
        raters[user]=len(prefs[user])
    return getTop(sorted(raters.items(), key=lambda x: x[1]))

def getSimilarRatings(prefs,movie,similar,n=2000):
    itemPrefs=transformPrefs(prefs)
    matches=topMatches(itemPrefs,movie,n=n,similarity=sim_pearson)
    sortedMatches=sorted(matches, key=lambda x: x[0], reverse=similar)
    return getTop(sortedMatches[::-1],1)

def getTopMovieRatingCounts(prefs):
    movies=getMovieRatings(prefs)
    for movie in movies.keys():
        movies[movie]=len(movies[movie])
    return getTop(sorted(movies.items(), key=lambda x: x[1]))

def getSimilarRaters(prefs,n=5,similar=True):
    results=[]
    for user in prefs.keys():
        for target in prefs.keys():
            if user != target:
                score=sim_pearson(prefs, user, target)
                if ((len(results) < n or ( (similar and score > results[0][2]
                    (not similar and score < results[n-1][2]))) and
                    not (target,user,score) in results):
                    results.append((user,target,score))
                if len(results) > n:
                    index = 0 if similar else n-1
                    results.remove(results[index])
                results=sorted(results, key=lambda x: x[2])
    return results

```

## Appendix B

```
#!/usr/bin/python2
import recommendations

#Return prettier string representing a set of movie/rating pairs
def stringify(movieData):
    string=''
    for item in movieData:
        for x in range(0,len(item)):
            string+=str(item[x]) + ' '
            if x < len(item)-1:
                string+= ' & '
        string+='\\\\\\n'
    return string + '\\n'

prefs=recommendations.loadMovieLens()
print 'Top 5 average score movies:\\n' + stringify(recommendations.getTopMovies(prefs))
print 'Top 5 movie rating counts:\\n' + stringify(recommendations.getTopMovieCounts(prefs))
print 'Top 5 movie rating by women:\\n' + stringify(recommendations.getTopMovieRatings(prefs, 'women'))
print 'Top 5 movie rating by men:\\n' + stringify(recommendations.getTopMovieRatings(prefs, 'men'))
print 'Most similar ot Top Gun: ' + str(recommendations.getSimilarRatings(prefs, 'Top Gun'))
print 'Least similar ot Top Gun: ' + str(recommendations.getSimilarRatings(prefs, 'Top Gun'))
print 'Top 5 raters:\\n' + stringify(recommendations.getTopRaters(prefs))
print '5 most similar raters:\\n' + stringify(recommendations.getSimilarRaters(prefs, 'Top Gun'))
print '5 least similar raters:\\n' + stringify(recommendations.getSimilarRaters(prefs, 'Top Gun'))
print 'Top 5 movie rating men over 40:\\n' + stringify(recommendations.getTopMovieRatings(prefs, 'men', 'over 40'))
print 'Top 5 movie rating men under 40:\\n' + stringify(recommendations.getTopMovieRatings(prefs, 'men', 'under 40'))
print 'Top 5 movie rating women over 40:\\n' + stringify(recommendations.getTopMovieRatings(prefs, 'women', 'over 40'))
print 'Top 5 movie rating women under 40:\\n' + stringify(recommendations.getTopMovieRatings(prefs, 'women', 'under 40'))
```