Ryan Condotta

<u>Final Report Assignment 7</u>

Question 1:

In order to accomplish the task of finding a user similar to me so I searched for user's who were 22 of age, occupation was engineer, and gender was male. I accomplished this in the A7.py file in the loadUsers function as shown in Figure 1. It opened the u.user file and searched for the requirements that I defined. I also used the parser from Programming Collective Intelligence to parse through the files in loadMovieLens().

```
def loadUsers():
        #get People
        me={}
        count = 0
        for line in open(r'C:\Users\Ryan\Documents\WebScience\Assignment7\ml-100k\ml-100k\u.user'):
                (id,age,gender,occupation,zipcode)= line.split('|')[0:5]
                #print(id,age,gender,occupation,zipcode)
                if int(age) == 22:
                        if gender == 'M':
                                if occupation == 'engineer':
                                        count +=1
                                        print(id,age,gender,occupation)
                                        me[count]=id

        return me
```

Figure 1

Once I had all the user's that met my requirements, I began to search for their favorite and least favorite movies that they had rated. The process of getting the favorite and least favorite ratings is defined in Figure 2. The function worked by taking in the user id and searching for each movie it rated by its id and then storing the rating in a list of favorites for movies rated 5.0 unless the list didn't have enough movies, then it would add 4.0 movies. It would store their least favorite movie if the rating was 1.0 and add 2.0 if the list did not have enough movies.

```
        def Movie(prefs, userid):
                userRatings = prefs[userid]
                favorite={}
                least={}

                for(item,rating) in userRatings.items():
                        #print(item,rating)
                        if int(rating) == 5.0:
                                #print(item,rating)
                                favorite[item]=rating
                        if int(rating) == 1.0:
                                #print(item,rating)
                                least[item]= rating
                        if len(least) < 3:
                                if int(rating) == 2.0:
                                #print(item,rating)
                                least[item]= rating
                        if len(favorite) < 3:
                                if int(rating) == 4.0:
                                #print(item,rating)
                                        favorite[item]= rating

                return favorite, least
```

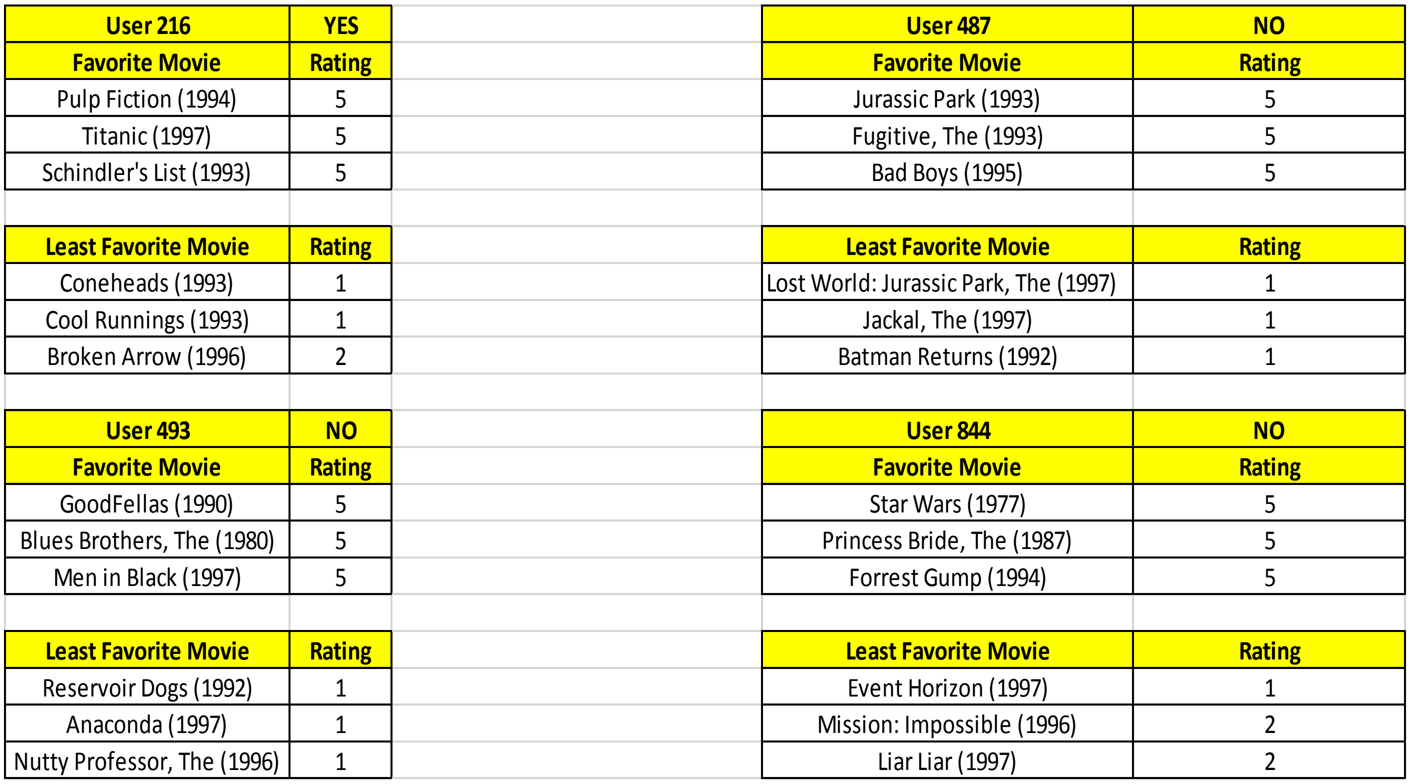
Figure 2

The last function I used to accomplish the task was the output function where it would use the Movie function from Figure 2 and get the users favorite and least favorite and then output it to a file that was the user's name and fav for favorite and least for least favorite. I stored all these output files into the Fav_least directory and the output function is shown in Figure 3.

```
def output(prefs):
        for (id,user) in person.items():
                favs, Less = Movie(prefs,user)
                for(item, rating) in favs.items():
                        print(item,rating)
                        filename = r"C:\Users\Ryan\Documents\WebScience\Assignment7\'"+user+"Fav.txt"
                        outfile = open(filename, 'a')
                        outfile.write(item +" "+str(rating))
                        outfile.write("\n")
                        outfile.close()
        #print(Less)
                for(item, rating) in Less.items():
                        print(item,rating)
                        filename = r"C:\Users\Ryan\Documents\WebScience\Assignment7\'"+user+"Least.txt"
                        outfile = open(filename, 'a')
                        outfile.write(item +" "+str(rating))
                        outfile.write("\n")
```

Figure 3

I took all the information from the txt files and created the excel file of Fav_Least_Users.xlxs where I organized all the information into tables as shown in Figure 4. These tables had the user's id and their least and favorite movies. This would I could see all the user's at the same time and decide who I was most like. After, carefully looking through each of the user's from Figure 4. I determined that user 216 was most like me. All his movies that were his favorite, I would rate the same as well, however, I would disagree with Cool Runnings being a 1 because it is at least a 3. That was my only disagreement with user 216 while the other user's, I had more disagreements with on their least and favorite movies.



User 216	YES
Favorite Movie	Rating
Pulp Fiction (1994)	5
Titanic (1997)	5
Schindler's List (1993)	5
Least Favorite Movie	Rating
Coneheads (1993)	1
Cool Runnings (1993)	1
Broken Arrow (1996)	2

User 487	NO
Favorite Movie	Rating
Jurassic Park (1993)	5
Fugitive, The (1993)	5
Bad Boys (1995)	5
Least Favorite Movie	Rating
Lost World: Jurassic Park, The (1997)	1
Jackal, The (1997)	1
Batman Returns (1992)	1

User 493	NO
Favorite Movie	Rating
GoodFellas (1990)	5
Blues Brothers, The (1980)	5
Men in Black (1997)	5
Least Favorite Movie	Rating
Reservoir Dogs (1992)	1
Anaconda (1997)	1
Nutty Professor, The (1996)	1

User 844	NO
Favorite Movie	Rating
Star Wars (1977)	5
Princess Bride, The (1987)	5
Forrest Gump (1994)	5
Least Favorite Movie	Rating
Event Horizon (1997)	1
Mission: Impossible (1996)	2
Liar Liar (1997)	2

Figure 4

Question 2:

In order to accomplish task 2, I used the same parser in loadMovieLens() to get all of the movie titles and load the ratings from the users for that and created the file A7Q2.py. I also used the load users function from Figure 1 without the requirements to get all the users information. Since I had all of the information stored and ready to use, I could begin the correlation test. I used the function CorrelationTest to perform the correlations between two users. User 1 is defined as the substitute me and user 2 will change by all the other individuals that rated movies. The function as shown in Figure 6 worked by calculating the mean for all the ratings that the each individual rated. Then, I got the ratings for each movie that both the user's rated. These pieces were needed for the pearson's correlation formula. This formula is as shown in Figure 5. The formula is used for the function and then outputs the R value for the user into the file 216ID.txt defined in CorrelationQ2.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Figure 5

```
def CorrelationTest(prefs, userid1, userid2):
    userRatings = prefs[userid1]
    userRatings1 = prefs[userid2]
    r={}
    meanuser1={}
    meanuser2={}
    tot1=0
    tot2=0
    mean1=0
    mean2=0
    r=0
    top = 0
    bottomright = 0
    bottomleft = 0

    #Calculate mean of user 1
    for(item,rating) in userRatings.items():
        tot1+= rating
    mean1= tot1/len(userRatings)
    #print("Mean1:",mean1)

    #Calculate mean of user 2
    for(item2,rating2) in userRatings1.items():
        tot2+= rating2
    mean2= tot2/len(userRatings1)
    #print("Mean2:", mean2)

    for(item,rating) in userRatings.items():
        for(item2,rating2) in userRatings1.items():
            if item == item2:
                top += (rating-mean1)*(rating2-mean2)
                bottomleft +=  (rating-mean1)*(rating-mean1)
                bottomright += (rating2-mean2)*(rating2-mean2)

    if bottomleft == 0 and bottomright == 0:
        r = 5
        return r
    else:
        r = top/(math.sqrt(bottomleft)*math.sqrt(bottomright))
        print("R:",r)
        filename = r"C:\Users\Ryan\Documents\WebScience\Assignment7\'"+user+"ID.txt"
        outfile = open(filename, 'a')
        outfile.write("R: "+str®+" user1: " + str(user1))
        outfile.write("\n")
        outfile.close()

    return r
```

Figure 6

Lastly, I took that R value and outputted the most similar people into the file of MaxCorrelated.txt and least correlated into the file LeastCorrelated.txt. These files showed the user's number and the R value. Least Correlated is shown in Figure 7 and Max Correlated is shown in Figure 8. Since the number of people was small enough for least and most correlated, I listed all the values to show the similarities and differences and how there was more than 5 for each case.

```
86 -1.0
172 -1.0
335 -1.0
440 -1.0
783 -1.0
824 -1.0
909 -1.0
```

Figure 7- Least Correlated

```
33 1.0
34 1.0
155 1.0
206 1.0
260 1.0
282 1.0
309 1.0
317 1.0
355 1.0
502 1.0
511 1.0
519 1.0
529 1.0
565 1.0
681 1.0
688 1.0
729 1.0
732 1.0
750 1.0
772 1.0
803 1.0
813 1.0
816 1.0
857 1.0
```

Figure 8- Max Correlated

Question 3:

In order to compute the ratings for all the films that the substitute me hasn't seen. I first created the file A7Q3.py and went about determining what type of movies genre each movie was. I limited it to one for simplicity of the assignment and calculations. This way each movie only has one genre and can't be defined as multiple ones. Next, I created a function named movie_type_notrated() that takes in the movie type and the user. It is used to figure out which genre of movie the user likes, and which genre of movie he dislikes in order to get a better calculations for the movies that user has not seen. From there, I outputted the information to a file named favmovietypes.txt which information is shown in Figure 9. The top genres are his favorite and the lowest ranked genres are below that. It shows that he rated comedy the most 1.0 as well as second for most 5.0 which would give it around 2.0. Action was the second worst so it will be the worst in this case. The genre that he favorited the most was drama so it will be given a 5.0 while those genre's with 0 will be given 3.0 ratings.

```
action 7 adventure1
animation 0 comedy7
crime 7 drama13
fantasy 0 mystery0
thriller 1 scifi0
war 0 western0

action 3 adventure0
animation 0 comedy9
crime 0 drama1
fantasy 0 mystery0
thriller 0 scifi0
war 0 western0
```

Figure 9

Next, I created the function of MoviesNotRated which takes in the movies, movie types, and user id. It iterates through the movies and finds the movie types and outputs the rating system I have given it and outputs the ratings to the file AllRated.txt as shown in Figure 10.

```
Eve's Bayou (1997) 5.0
Courage Under Fire (1996) 5.0
Fish Called Wanda, A (1988) 4.0
My Own Private Idaho (1991) 5.0
Little City (1998) 2.0
Broken Arrow (1996) 2.0
Van, The (1996) 2.0
```

Figure 10

Lastly, I created the files NotRecommended and Recommend.txt which are shown in Figure 11 and Figure 12. They show which movies the substitute me will most likely love and most likely hate.

```
Coldblooded (1995) 1.0
Top Gun (1986) 1.0
Star Trek: First Contact (1996) 1.0
Boot, Das (1981) 1.0
Tank Girl (1995) 1.0
```

Figure 11- Not Recommended

```
Eve's Bayou (1997) 5.0
Courage Under Fire (1996) 5.0
My Own Private Idaho (1991) 5.0
Mary Reilly (1996) 5.0
```

Being Human (1993) 5.0

Figure 12- Recommended

Question 4:

In order to accomplish task 4, I created the files of A7Q4.py, MyFav.txt, and MyLeastFav.txt for the purposes of the assignment. MyFav.txt is shown in Figure 13 and MyLeastFav.txt is shown in Figure 14 where the movies I shown are in Bold.

Billy Madison (1995)
Pulp Fiction (1994)
Shawshank Redemption, The (1994)
Ace Ventura: Pet Detective (1994)
Maverick (1994)

Figure 13- My Fav

Terminator 2: Judgment Day (1991)
Gone with the Wind (1939)
Aliens (1986)
Terminator, The (1984)
Shining, The (1980)

Figure 14- Least Fav

Next, I needed to get the movies that are least correlated and least correlated with each of these movies. So, I used the pearson correlation formula but instead of comparing user's, I compared the movies. I would get the average rating of each movie as defined in the getMean function as shown in Figure 15.

```
def getMean(prefs, movie1):
    mean1=0
    tot1 = 0
    count = 0
    for(user, movie) in prefs.items():
        userRatings = prefs[user]
        for (item, rating) in userRatings.items():
            if movie1 == item:
                #print(movie1,user,item,rating)
                count+=1
                tot1+= rating

    mean1= tot1/count
    return mean1
```

Figure 15

Once, I had the mean value ratings for each movie, I created the function rvalue as shown in Figure 16, which uses those mean values as well as the same individuals rating for the comparing movies and plug those values in to the equation.

```
def rvalue(mean1,mean2,prefs, movie1,movie2):
r={}
r=0
top = 0
bottomright = 0
bottomleft = 0

for(user, movie) in prefs.items():
    userRatings = prefs[user]
    for (item, rating) in userRatings.items():
        if movie1 == item:
            for(user1, move) in prefs.items():
                userRatings = prefs[user1]
                for (item1, rating1) in userRatings.items():
                    if movie2 == item1:
                        if user1 == user:
                            top += (rating-mean1)*(rating1-mean2)
                            bottomleft +=  (rating-mean1)*(rating-mean1)
                            bottomright += (rating1-mean2)*(rating1-
mean2)

if bottomleft == 0 or bottomright == 0:
    r = 5
    return r
else:
    r = top/(math.sqrt(bottomleft)*math.sqrt(bottomright))
    return r
```

Figure 16

This would return the r value for the comparison between the movies and I outputted the information to relative files of MostCorrelated.txt and LeastCorrelated depending on which movie I was getting a correlation for. Figure 17 and Figure 18 show the Correlations values for my Favorite movie Billy Madison. Figure 19 and Figure 20 show the Correlations values for my Favorite movie Billy Madison.

```
R: -1.0 movie1: Safe Passage (1994)
R: -1.0 movie1: Evening Star, The (1996)
R: -1.0 movie1: Shadow Conspiracy (1997)
R: -1.0 movie1: When the Cats Away (Chacun cherche son chat) (1996)
R: -1.0 movie1: American in Paris, An (1951)
```

Figure 17- Billy Madison Least Correlated

```
R: 1.0 movie1: Dream Man (1995)
R: 1.0 movie1: 1-900 (1994)
```

```
R: 1.0 movie1: Prisoner of the Mountains (Kavkazsky Plennik) (1996)
R: 1.0 movie1: Withnail and I (1987)
R: 1.0 movie1: Relative Fear (1994)
```

Figure 18-Billy Madison Most Correlated

```
R: -1.0 movie1: Solo (1996)
R: -1.0 movie1: Gabbeh (1996)
R: -1.0 movie1: Nico Icon (1995)
R: -1.0 movie1: Children of the Revolution (1996)
R: -1.0 movie1: Kicked in the Head (1997)
```

Figure 19- Gone With the Wind Least Correlated

```
R: 1.0 movie1: Paris, France (1993)
R: 1.0 movie1: New York Cop (1996)
R: 1.0 movie1: Line King: Al Hirschfeld, The (1996)
R: 1.0 movie1: Collectionneuse, La (1967)
R: 1.0 movie1: Losing Chase (1996)
```

Figure 20- Gone With the Wind Most Correlated

I did not like the results from both of the correlation tests for the movies that I used. They do not seem accurate in my opinion to the type of movie that I enjoy. Even though I have not seen the majority of the movies or even heard of them, I still believe that these results are not accurate according to what I like to watch. However, I cannot say I personally like or dislike the movies on the list since I have not seen the majority of them. Unfortunate to say the least..