HW 4

CS 216, Everything Data, Spring 2020

DUE: Monday Feb. 17 by 4:40 pm (class time)

In this assignment, you will practice some statistical and probabilistic arguments that are useful for exploratory data analysis. You will include all of your answers for this assignment within this notebook. You will then convert your notebook to a .pdf and a .py file to submit to gradescope (submission instructions are included at the bottom).

Please take note of the course collaboration policy (https://sites.duke.edu/compsci216s2020/policies/). You may work alone or with a single partner. If you work with a partner, you may not split up the assignment; you should work together in-person or complete parts independently and come together to discuss your solutions. In either case, you are individually responsible for your work, and should understand everything in your submission.

Part 1: Team Probabilities

Suppose that we take 150 students and randomly assign them to groups of 5 students each by the following procedure. First, we generate a uniform random permutation of the students (that is, a random ordering where for a given student there is an equal probability of appearing at any position in the ordering from 1, 2, 3, ..., 150). Then we simply assign the first five students in the ordering to the first group, the second five to the second group, and so on. Compute the following probabilities.

Note that you can arrive at your answer and show your work either analytically (show your math) or computationally (show code for your simulation). With the latter approach, you simulate the process many times, and see what fraction of these times the event of interest occcurs. In either case, we will count your answer as correct if it is correct rounded to the second decimal place (that is, if the real answer is 0.0312 and you get 0.0297, we will count that as correct).

Problem A

Suppose a student has a single best friend in the class. What is the probability that a given student gets placed in a group with their best friend?

Student A and Student B are best friends. Given that A is already selected as 1, with 30 groups of 5 it can be either (1&2,1&3,1&4,1&5).If B is not selected from 2-5, they are not in the same group. So the probability is equal to 4/149.

Problem B

Suppose a student has a single worst enemy in the class. What is the probability that a given student is *not* placed in a group with their worst enemy?

Take the complement of Problem A, 1 - 4/149 = 145/149.

Problem C

Suppose that the best friend relationship is symmetric: if Bob is Alice's best friend, then Alice is Bob's best friend; there are exactly 75 such best friend pairs. What is the expected number of best friend pairs that get placed into the same group accross all of the groups?

X = number of best friend pairs that get placed into the same group The probability of a best friends pair getting selected is 4/149. E[X] = 75*(4/149) = 2.01. About 2 pairs. ??

Problem D

What is the probability that every best friend pair gets placed into the same group? (*hint* Don't overthink it).

This is impossible because there are groups of 5. If 1A and 1B, 2A and 2B and 3A and 3B are best friends, 1A and 1B, 2A and 2B and 3A could be a group, and the last one 3B would always be left off because their is an odd number of group members. So the probability is 0.

Problem E

What is the probability that no best friend pairs get placed into the same group?

It is near 0. It should be less than one percent.

Problem F

Now suppose that we couldn't get a big enough room, and there are 5 sections of the class that meet at different times, each with 30 students. Suppose we randomly assign groups as before, but only within sections so that there are 6 random groups with students from the first section, 6 random groups with students from the second section, and so on.

Reasonably enough, students may prefer to sign up for sections with their best friends. Suppose that 50 out of the 75 best friend pairs signed up for the same sections, and 25 of the best friend pairs had to take different sections. Given this information, what is the expected number of best friend pairs that get placed into the same group accross all of the groups?

In [7]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-901aba5ba902> in <module>
      1 n = 1000
----> 2 friendcount = np.empty(n,dtype = int)
      3 for x in range(n):
      4     count = 0
      5     students = np.arrange(150)

NameError: name 'np' is not defined
```

Part 2: Birthmonth Probabilities

To simplify this problem, assume that a year has twelve equal duration months, rather than the unequal months in the standard calendar. As before, show your work, either mathematical reasoning or simulation code.

Problem A

Assuming that people are equally likely to be born in any month, how many people must be selected uniformly at random before the probability that at least two of the selected people have the same birth month is at least 1/2?

P(Any birthmonth) = 1/12 When the probability that at least two of the selected people have the birthmonth is 1/2, that means that there is a 50% chance of selecting two people with the same birthmonth. This probability is true when there is at least 12 pairs of same birthdays. When there are 12 pairs of same birthdays, there is a 50% chance of picking a pair that has the same birthday. 12 pairs is equal to 24 people so 23 people must be selected uniformly at random before the probability that at least two of the selected people have the same birthmonth.

Problem B

Now assume that for each month among May, June, July, and August, the probability of being born in that month is 1/10, whereas for every other month the probability of being born in that month is 3/40. What is the probability that two people selected uniformly at random have the same birth month? Is this more or less than in problem A?

4(1/10 *1/10) + 8(3/40* 3/40) =

4 represents May, June, July, and August, with 8 representing the other months. We multiply 1/10 and 3/40 by themselves because we are comparing the birth months of two randomly selected people (with these birth months being the same)

4/100 + 72/1600 =

8/200 + 9/200 =

17/200

Part 3: Exploring Movie Lens Data

The movie lens dataset contains about 100,000 ratings from 1000 users on 1700 movies, made available by grouplen.org. This information is in three files: `u.user` , `u.data` , and `u.item` . Below, we import the relevant information into three dataframes (about the users, the ratings, and the movies). You are welcome to work directly with these dataframes, or to transform the data however you like for solving the problems in this part. Most of the data is self explanatory, but be sure to note that a given movie can be categorized as multiple genres (see the movies dataframe / `u.item` file below). To denote this, each row of the table corresponds to a given movie, and there is a column for every genre; there is a 1 in the column if the movie is of that genre, and a 0 otherwise.

In [1]:
```python
import pandas as pd

df_users = pd.read_table('u.user', sep='|', names = ['user_id', 'age', 'sex',
'occupation', 'zip'])
df_users.head()
```

Out[1]:
```
   user_id  age  sex  occupation    zip
0        1   24    M  technician  85711
1        2   53    F       other  94043
2        3   23    M      writer  32067
3        4   24    M  technician  43537
4        5   33    F       other  15213
```

In [4]:
```
df_ratings = pd.read_table('u.data', sep=',', names = ['user_id', 'movie_id',
'rating', 'timestamp'])
df_ratings.head()
```

Out[4]:

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

In [2]:
```
df_movies = pd.read_table('u.item', sep='|', encoding='latin',\
                          names = ['movie_id', 'movie_title', 'release_date',
'video_release_date',\
                                   'imdb_url', 'unknown', 'action', 'adventur
e', 'animation',\
                                   'children', 'comedy', 'crime', 'documentar
y', 'drama', 'fantasy',\
                                   'film_noir', 'horror', 'musical', 'myster
y', 'romance', 'sci_fi',\
                                   'thriller', 'war', 'western'])
df_movies.head()
```

Out[2]:

	movie_id	movie_title	release_date	video_release_date	imdb_url	unknown	act
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	

5 rows × 24 columns

Problem A

Compute the average rating for every genre of movie (for your convenience, we define the genres as a list below corresponding to the column names in the movies dataframe). Which genre is rated the highest on average?

In [4]:
```
genres = ['unknown', 'action', 'adventure', 'animation', 'children',
          'comedy', 'crime', 'documentary', 'drama', 'fantasy',
          'film_noir', 'horror', 'musical', 'mystery', 'romance',
          'sci_fi','thriller', 'war', 'western']
```

In [ ]:
```
for genre in genres:
    total = 0
    count = 0
    for x in range(len(df_movies['movie_id'])):
        if df_movies[genre][x] == 1:
            for movie in range(len(df_ratings['movie_id'])):
                if df_ratings['movie_id'][movie] == df_movies['movie_id'][x]:
                    count += 1
                    total += df_ratings['rating'][movie]
    print(genre, total/count)
```

```
unknown 3.2
action 3.480245417953027
adventure 3.503526503308369
animation 3.5766990291262135
children 3.3532442216652742
comedy 3.3940734781442745
crime 3.6322780881440098
documentary 3.6728232189973613
```

This was running slow because of the nested forloop looping around 1700 times 100000. I individually ran each genre and found out that film_noir was the highest rated average with 3.92.

In [ ]:
```
total2 = 0
count2 = 0
for x in range(len(df_movies['movie_id'])):
    if df_movies['film_noir'][x] == 1:
        for movie in range(len(df_ratings['movie_id'])):
            if df_ratings['movie_id'][movie] == df_movies['movie_id'][x]:
                count2 += 1
                total2 += df_ratings['rating'][movie]
print('film_noir', total2/count2)
```

film_noir 3.9215233698788228

Problem B

What is the probability that a random user likes (rates 4 or 5) at least one horror movies given that the the user likes (rates 4 or 5) at least one musical.

```
In [65]: userIDsetMusic = set()
         userIDsetHorror = set()
         for x in range(len(df_movies['movie_id'])):
             if df_movies['musical'][x] == 1:
                 for y in range(len(df_ratings['movie_id'])):
                     if df_movies['movie_id'][x] == df_ratings['movie_id'][y]:
                         if df_ratings['rating'][y] >= 4:
                             userIDsetMusic.add(df_ratings['user_id'][y])
             if df_movies['horror'][x] == 1:
                 for z in range(len(df_ratings['movie_id'])):
                     if df_movies['movie_id'][x] == df_ratings['movie_id'][z]:
                         if df_ratings['rating'][z] >= 4:
                             userIDsetHorror.add(df_ratings['user_id'][z])
         print(len(userIDsetMusic))
         print(len(userIDsetHorror))
         print(len(userIDsetMusic.intersection(userIDsetHorror)))
```

```
594
608
449
0
```

The probability is 449/594.

Problem C

What is the expected age of a random user that likes (rates a 4 or 5) at least one children's movie?

In [4]:
```python
agesetID = set()
for x in range(len(df_movies['movie_id'])):
    if df_movies['children'][x] == 1:
        for k in range(len(df_ratings['movie_id'])):
            if df_movies['movie_id'][x] == df_ratings['movie_id'][k]:
                if df_ratings['rating'][k] >= 4:
                    agesetID.add(df_ratings['user_id'][k])
agelistID = list(agesetID)
ages = []
for ID in agelistID:
    for age in range(len(df_users['age'])):
        if ID == df_users['user_id'][age]:
            ages.append(df_users['age'][age])

ages.sort()
print(len(ages))
print(ages)
```

```
662
[7, 10, 13, 13, 13, 13, 14, 15, 15, 15, 15, 15, 16, 16, 16, 16, 17, 17, 17, 1
7, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19, 1
9, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 2
0, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 2
1, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 2
1, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 2
2, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 24, 2
4, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 2
4, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 2
5, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 2
6, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 2
7, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 2
7, 27, 27, 27, 27, 27, 27, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 2
8, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 29, 29, 2
9, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 2
9, 29, 29, 29, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 3
0, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 3
1, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 3
2, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33, 3
3, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 3
3, 33, 34, 34, 34, 34, 34, 34, 34, 34, 34, 35, 35, 35, 35, 35, 35, 35, 35, 3
5, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 36, 36, 36, 36, 36, 3
6, 36, 36, 36, 36, 36, 36, 36, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 3
7, 37, 37, 38, 38, 38, 38, 38, 38, 38, 38, 38, 38, 38, 38, 39, 39, 39, 39, 3
9, 39, 39, 39, 39, 39, 39, 39, 39, 39, 39, 40, 40, 40, 40, 40, 40, 40, 40, 4
0, 40, 40, 40, 40, 40, 40, 40, 41, 41, 41, 41, 41, 41, 42, 42, 42, 42, 42, 4
2, 42, 42, 42, 42, 42, 42, 42, 42, 43, 43, 43, 43, 43, 43, 43, 43, 43, 44, 4
4, 44, 44, 44, 44, 44, 44, 44, 44, 44, 44, 44, 44, 44, 45, 45, 45, 45, 45, 4
5, 45, 45, 46, 46, 46, 46, 46, 46, 46, 46, 47, 47, 47, 47, 47, 47, 47, 47, 4
7, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 49, 49, 49, 4
9, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 50, 50, 50, 50, 50, 50, 50, 50, 5
0, 50, 50, 50, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 5
2, 52, 52, 52, 52, 53, 53, 53, 53, 53, 53, 53, 53, 54, 54, 54, 55, 55, 55, 5
5, 55, 56, 56, 56, 57, 57, 57, 57, 57, 57, 58, 58, 58, 59, 59, 59, 60, 60, 6
0, 60, 60, 60, 60, 61, 61, 61, 63, 64, 64, 65, 66, 70, 70, 73]
```

In [15]:
```
Eval = 7*(1/662) + 10*(1/662) + 13*(4/662) + 14*(1/662) + 15*(5/662) + 16*(4/6
62) + 17*(6/662) + 18*(12/662) + 19*(17/662) + 20*(25/662) + 21*(22/662) + 22*
(23/662) + 23*(11/662) + 24*(21/662) + 25*(25/662) + 26*(27/662) + 27*(29/662)
+ 28*(29/662) + 29*(25/662) + 30*(29/662) + 31*(16/662) + 32*(23/662) + 33*(21
/662) + 34*(9/662) + 35*(22/662) + 36*(13/662) + 37*(14/662) + 38*(10/662) + 3
9*(14//662) + 40*(16/662) + 41*(6/662) + 42*(14/662) + 43*(9/662) + 44*(15/662
) + 45*(7/662) + 46*(8/662) + 47*(9/662) + 48*(15/662) + 49*(14/662) + 50*(12/
662) + 51*(15/662) + 52*(5/662) + 53*(8/662) + 54*(2/662) + 55*(5/662) + 56*(3
/662) + 57*(6/662) + 58*(3/662) + 59*(3/662) + 60*(7/662) + 61*(3/662) + 63*(1
/662) + 64*(2/662) + 65*(1/662) + 66*(1/662) + 70*(2/662) + 73*(1/662)

print(Eval)
```

```
32.38066465256798
```

Problem D

Is the probability that a random user likes (rates a 4 or 5) at least one drama independent of the probability that a random user likes (rates a 4 or 5) at least one comedy? Explain your answer.

In [6]:
```
userIDsetDrama = set()
userIDsetComedy = set()
for x in range(len(df_movies['movie_id'])):
    if df_movies['drama'][x] == 1:
        for y in range(len(df_ratings['movie_id'])):
            if df_movies['movie_id'][x] == df_ratings['movie_id'][y]:
                if df_ratings['rating'][y] >= 4:
                    userIDsetDrama.add(df_ratings['user_id'][y])
    if df_movies['comedy'][x] == 1:
        for z in range(len(df_ratings['movie_id'])):
            if df_movies['movie_id'][x] == df_ratings['movie_id'][z]:
                if df_ratings['rating'][z] >= 4:
                    userIDsetComedy.add(df_ratings['user_id'][z])
print(len(userIDsetComedy))
print(len(userIDsetDrama.intersection(userIDsetComedy)))
```

```
916
914
```

The probability that a random user likes at least one drama is independent of the probability that a random user likes comedy. This means that given that one of them is true it doesn't change the probability of the other. Given that one is true, that doesn't affect the probability of the other. If they were independent, the users would have a significant difference. The intersection of Drama and Comedy is 914 which is just two off of the the amount of users in comedy.

Submitting HW 4

1. Double check that you have written all of your answers along with your supporting work in this notebook. Make sure you save the complete notebook.
2. Double check that your entire notebook runs correctly and generates the expected output. To do so, you can simply select Kernel -> Restart and Run All.
3. You will download two versions of your notebook to submit, a .pdf and a .py. To create a PDF, we reccomend that you select File --> Download as --> HTML (.html). Open the downloaded .html file; it should open in your web broser. Double check that it looks like your notebook, then print a .pdf using your web browser (you should be able to select to print to a pdf on most major web browsers and operating systems). Check your .pdf for readability: If some long cells are being cut off, go back to your notebook and split them into multiple smaller cells. To get the .py file from your notebook, simply select File -> Download as -> Python (.py) (note, we recognize that you may not have written any Python code for this assignment, but will continue the usual workflow for consistency).
4. Upload the .pdf to gradescope under hw4 report and the .py to gradescope under hw4 code. If you work with a partner, only submit one document for both of you, but be sure to add your partner using the group feature on gradescope (https://www.gradescope.com/help#help-center-item-student-group-members).

```
In [ ]:
```