

# Task ML. My recommender system: Subreddits

- Student name: Yori Verbist
- Number: r0697651

First find a topic which you can develop a recommender system for. Don't use a film recommender system this time, try to be a little more original. Maybe build a recommender system for TV series, holiday destinations, computer games, favorite (exotic) dishes or recipes, city trip destinations, computer games, ... Use your imagination, you should definitely be able to find something funny. List at least 20 different items on the chosen subject.

Reddit Subreddits, because there are a lot of them so there will always be some you have never heard of before.

Next think of your target group. Who do you want to set up the recommender system for? For friends, family, classmates, ...? Design and develop a tool to collect the data for your recommender system. A possible tool is *Google Forms*, but maybe you know a better way. Now, using the tool, ask your target group, to rate the (at least) 20 items. You should also at least have 20 different respondents.

**Make sure you provide the option "not yet tried" or "no rating" in your survey. If everyone gives a rating for all the data, there is nothing to recommend.**

Questions

Responses 20

## Subreddit ratings

Give a score for each subreddit if you know it. Select 'don't know' if you haven't heard about the subreddit yet.

r/math

☐ 3

☐ 2

☐ 1

☐ Don't know

r/learnmath \*

☐ 3

☐ 2

☐ 1

☐ Don't know

+

↶

⌵

🖼

▶

☰

The following step is to convert the data in the correct json-format. Insert below a part of the json-file:

```
{
  "Tijdstempel": {
    "0": "17-10-2020 17:39:09",
    "1": "17-10-2020 17:39:45",
    "2": "17-10-2020 17:40:48",
    "3": "17-10-2020 17:46:00",
    "4": "17-10-2020 17:51:51",
    "5": "17-10-2020 17:51:53",
    "6": "17-10-2020 18:08:21",
    "7": "17-10-2020 19:55:59",
    "8": "17-10-2020 21:27:32",
    "9": "17-10-2020 21:39:46",
    "10": "18-10-2020 12:47:46",
    "11": "18-10-2020 18:09:02",
```

First import the code library with the functions:

```
In [169... import sys
sys.path.append('recommender/collaborative_filtering.py')

from recommender.collaborative_filtering import euclidean_score, pearson_score
```

Read the json file and calculate the similarity score between two of your respondents using the two different methods. Does it make sense? Explain briefly.

```
In [170... ratings_file = 'recommender/subreddits.csv'

import pandas as pd

df = pd.read_csv(ratings_file)
df.to_json('recommender/subreddits.json')
```

```
In [185... #print(df := pd.read_json('recommender/subreddits.json'))
import json

def parse_json(data):
    users = ["User" + str(i) for i in range(len(data["r/math"]))]
    ratings = {}
    #print(data.keys())
    for i in range(len(users)):
        ratings[users[i]] = {}
        for value in data.keys():
            if value == 'Tijdstempel':
                pass
            elif data[value][str(i)] in ['1', '2', '3']:
                ratings[users[i]][str(value)] = int(data[value][str(i)])
            elif data[value][str(i)] == "Don't know":
                pass
            else:
                ratings[users[i]][str(value)] = data[value][str(i)]

    return ratings
```

```
In [186... with open('recommender/subreddits.json') as json_file:
    data = json.load(json_file)

data = parse_json(data)
print(data)
```

```
{'User0': {'r/programming': 2, 'r/aww': 1, 'r/EarthPorn': 1, 'r/AskReddit': 3,
'r/LifeProTips': 2, 'r/ProgrammerHumor': 2, 'r/wallstreetbets': 1, 'r/wholesomememes': 2, 'r/explainlikeimfive': 3, 'r/dankmemes': 3, 'r/todayilearned': 3},
'User1': {'r/programming': 2, 'r/aww': 2, 'r/NatureIsFuckingLit': 3, 'r/EarthPorn': 1, 'r/AskReddit': 1, 'r/LifeProTips': 1, 'r/ProgrammerHumor': 2, 'r/Tinder': 1, 'r/wallstreetbets': 2, 'r/wholesomememes': 1, 'r/me_irl': 1, 'r/IdiotsInCars': 3, 'r/oddlysatisfying': 2, 'r/explainlikeimfive': 2, 'r/dankmemes': 1, 'r/todayilearned': 1}, 'User2': {'r/learnmath': 2, 'r/aww': 3, 'r/NatureIsFuckingLit': 3, 'r/EarthPorn': 3, 'r/AskReddit': 1, 'r/LifeProTips': 1, 'r/Tinder': 3, 'r/wholesomememes': 3, 'r/me_irl': 3, 'r/IdiotsInCars': 3, 'r/oddlysatisfying': 3, 'r/explainlikeimfive': 3, 'r/dankmemes': 3, 'r/todayilearned': 3}, 'User3': {'r/math': 1, 'r/learnmath': 1, 'r/programming': 1, 'r/learnprogramming': 1, 'r/aww': 3, 'r/NatureIsFuckingLit': 3, 'r/EarthPorn': 3, 'r/AskReddit': 1, 'r/LifeProTips': 2, 'r/ProgrammerHumor': 2, 'r/Tinder': 3, 'r/wholesomememes': 3, 'r/me_irl': 2, 'r/IdiotsInCars': 3, 'r/oddlysatisfying': 3, 'r/explainlikeimfive': 3, 'r/dankmemes': 3, 'r/todayilearned': 3}, 'User4': {'r/math': 1, 'r/aww': 3, 'r/NatureIsFuckingLit': 2, 'r/EarthPorn': 2, 'r/AskReddit': 2, 'r/LifeProTips': 1, 'r/Tinder': 2, 'r/wholesomememes': 1, 'r/me_irl': 1, 'r/IdiotsInCars': 2, 'r/CozyPlaces': 3, 'r/oddlysatisfying': 2, 'r/explainlikeimfive': 3, 'r/dankmemes': 3, 'r/todayilearned': 3}, 'User5': {'r/math': 3, 'r/learnmath': 1, 'r/programming': 3, 'r/learnprogramming': 1, 'r/aww': 3, 'r/NatureIsFuckingLit': 3, 'r/EarthPorn': 2, 'r/AskReddit': 2, 'r/LifeProTips': 1, 'r/ProgrammerHumor': 2, 'r/Tinder': 1, 'r/wallstreetbets': 3, 'r/wholesomememes': 2, 'r/explainlikeimfive': 2, 'r/dankmemes': 2, 'r/todayilearned': 2}}
```

```
{
  'r/IdiotsInCars': 3, 'r/CozyPlaces': 3, 'r/oddlysatisfying': 2}, 'User6': {
    'r/EarthPorn': 1, 'r/AskReddit': 3, 'r/LifeProTips': 3, 'r/ProgrammerHumor': 2,
    'r/Tinder': 2, 'r/wallstreetbets': 1, 'r/wholesomememes': 1, 'r/IdiotsInCars': 1,
    'r/oddlysatisfying': 1, 'r/explainlikeimfive': 1, 'r/dankmemes': 3, 'r/todayilearned': 1},
  'User7': {'r/programming': 2, 'r/aww': 2, 'r/AskReddit': 1, 'r/ProgrammerHumor': 2,
    'r/wholesomememes': 2, 'r/me_irl': 1}, 'User8': {'r/math': 3, 'r/learnmath': 2,
    'r/AskReddit': 2, 'r/LifeProTips': 1, 'r/Tinder': 2, 'r/todayilearned': 2},
  'User9': {'r/programming': 2, 'r/learnprogramming': 2, 'r/aww': 1, 'r/NatureIsFuckingLit': 2,
    'r/EarthPorn': 1, 'r/AskReddit': 1, 'r/LifeProTips': 3, 'r/ProgrammerHumor': 2,
    'r/Tinder': 3, 'r/wallstreetbets': 2, 'r/wholesomememes': 3, 'r/me_irl': 2,
    'r/IdiotsInCars': 2, 'r/CozyPlaces': 1, 'r/oddlysatisfying': 3, 'r/dankmemes': 3},
  'User10': {'r/math': 1, 'r/learnmath': 1, 'r/programming': 3, 'r/learnprogramming': 3,
    'r/aww': 2, 'r/NatureIsFuckingLit': 3, 'r/EarthPorn': 3, 'r/AskReddit': 3,
    'r/LifeProTips': 2, 'r/ProgrammerHumor': 2, 'r/Tinder': 3, 'r/wallstreetbets': 2,
    'r/wholesomememes': 2, 'r/me_irl': 2, 'r/IdiotsInCars': 3, 'r/CozyPlaces': 1,
    'r/oddlysatisfying': 3, 'r/explainlikeimfive': 2, 'r/dankmemes': 3, 'r/todayilearned': 2},
  'User11': {'r/math': 2, 'r/learnmath': 3, 'r/aww': 1, 'r/NatureIsFuckingLit': 1,
    'r/AskReddit': 2, 'r/LifeProTips': 1, 'r/wallstreetbets': 1, 'r/me_irl': 1,
    'r/CozyPlaces': 2, 'r/oddlysatisfying': 3, 'r/todayilearned': 2},
  'User12': {'r/math': 1, 'r/programming': 2, 'r/aww': 1, 'r/NatureIsFuckingLit': 2,
    'r/EarthPorn': 3, 'r/ProgrammerHumor': 2, 'r/Tinder': 1, 'r/wallstreetbets': 3,
    'r/wholesomememes': 1, 'r/me_irl': 2, 'r/IdiotsInCars': 3, 'r/explainlikeimfive': 2,
    'r/dankmemes': 1, 'r/todayilearned': 3},
  'User13': {'r/learnmath': 2, 'r/programming': 2, 'r/NatureIsFuckingLit': 3,
    'r/EarthPorn': 1, 'r/AskReddit': 2, 'r/Tinder': 1, 'r/wholesomememes': 3,
    'r/me_irl': 1, 'r/CozyPlaces': 2, 'r/oddlysatisfying': 3, 'r/explainlikeimfive': 1,
    'r/dankmemes': 1, 'r/todayilearned': 3},
  'User14': {'r/math': 3, 'r/learnmath': 2, 'r/programming': 3, 'r/learnprogramming': 2,
    'r/AskReddit': 3, 'r/LifeProTips': 1, 'r/ProgrammerHumor': 3, 'r/Tinder': 2,
    'r/wallstreetbets': 3, 'r/me_irl': 1, 'r/IdiotsInCars': 3, 'r/dankmemes': 2,
    'r/todayilearned': 3},
  'User15': {'r/programming': 2, 'r/EarthPorn': 2, 'r/AskReddit': 2, 'r/ProgrammerHumor': 2,
    'r/Tinder': 1, 'r/wallstreetbets': 3, 'r/wholesomememes': 3, 'r/IdiotsInCars': 2,
    'r/oddlysatisfying': 3, 'r/explainlikeimfive': 1, 'r/dankmemes': 3},
  'User16': {'r/aww': 3, 'r/NatureIsFuckingLit': 3, 'r/EarthPorn': 3, 'r/AskReddit': 2,
    'r/Tinder': 2, 'r/wholesomememes': 2, 'r/me_irl': 1, 'r/CozyPlaces': 3,
    'r/oddlysatisfying': 3, 'r/todayilearned': 3},
  'User17': {'r/math': 3, 'r/learnmath': 3, 'r/aww': 3, 'r/AskReddit': 3,
    'r/LifeProTips': 2, 'r/me_irl': 1, 'r/dankmemes': 3, 'r/todayilearned': 3},
  'User18': {'r/math': 2, 'r/learnmath': 3, 'r/programming': 1, 'r/learnprogramming': 3,
    'r/aww': 2, 'r/NatureIsFuckingLit': 1, 'r/EarthPorn': 2, 'r/AskReddit': 3,
    'r/ProgrammerHumor': 3, 'r/Tinder': 2, 'r/wallstreetbets': 1, 'r/me_irl': 3,
    'r/IdiotsInCars': 1, 'r/CozyPlaces': 2, 'r/oddlysatisfying': 2, 'r/explainlikeimfive': 2,
    'r/dankmemes': 2, 'r/todayilearned': 2},
  'User19': {'r/programming': 3, 'r/learnprogramming': 3, 'r/aww': 3, 'r/NatureIsFuckingLit': 3,
    'r/EarthPorn': 2, 'r/ProgrammerHumor': 3, 'r/wholesomememes': 2, 'r/me_irl': 1,
    'r/IdiotsInCars': 3, 'r/CozyPlaces': 3, 'r/oddlysatisfying': 3, 'r/explainlikeimfive': 2,
    'r/dankmemes': 1, 'r/todayilearned': 3}}
```

In [187...

```
user1 = "User1"
user2 = "User2"

print("\nEuclidean score:")
print(euclidean_score(data, user1, user2))

print("\nPearson score:")
print(pearson_score(data, user1, user2))
```

Euclidean score:  
0.16139047779640892

Pearson score:  
0.30785964799347953

Find - given a respondent - five similar respondents.

In [190...

```

user = "User19"

print('\nUsers similar to ' + user + ':\n')
similar_users = find_similar_users(data, user, 5)
print('User\t\t\tPearson similarity score')
print('-'*48)
for item in similar_users:
    print(item[0], '\t\t\t', round(float(item[1]), 2))

```

Users similar to User19:

User	Pearson similarity score
-----	-----
User16	0.89
User7	0.87
User14	0.81
User13	0.72
User1	0.65

Use your recommendation system to give some recommendations for one respondent.

In [191...

```

print("\Subreddit recommendations for " + user + ":")
movies = get_recommendations(data, user)
for i, movie in enumerate(movies):
    print(str(i+1) + '. ' + movie)

```

\Subreddit recommendations for User19:

1. r/learnmath
2. r/math
3. r/wallstreetbets
4. r/AskReddit
5. r/Tinder
6. r/LifeProTips

List all the respondents in a table. For every respondent, you show the most similar respondent and the first recommendation.

In [210...

```

print('User\t\t\tPearson similar respondent \t\t\t first recommendation')
print('-'*90)
for user in (users := ["User" + str(i) for i in range(len(data))]):
    similar_users = find_similar_users(data, user, 5)
    movies = get_recommendations(data, user)
    print(user, "\t\t\t", similar_users[0][0] + "\t\t\t\t", movies[0])

```

User	Pearson similar respondent	first recommendation
-----	-----	-----
User0	User8	r/learnprogramming
User1	User5	r/CozyPlaces
User2	User3	r/CozyPlaces
User3	User2	r/wallstreetbets
User4	User16	r/learnprogramming
User5	User14	r/todayilearned
User6	User18	r/learnmath
User7	User19	r/CozyPlaces
User8	User0	r/oddlysatisfying
User9	User0	r/todayilearned
User10	User3	No recommendations po
ssible		
User11	User0	r/learnprogramming
User12	User14	r/CozyPlaces
User13	User14	r/learnprogramming

User14	User16	r/CozyPlaces
User15	User13	r/CozyPlaces
User16	User19	r/learnprogramming
User17	User16	r/learnprogramming
User18	User6	r/wholesomememes

Print/export your Jupyter Notebook to a pdf file and upload it using Canvas.

In [ ]: