Ime, priimek: ŽIGA FON

LV06: Priporočilni sistem za filme

Namen vaje:

- Spoznati podatke o ocenah filmov
- Ugotoviti podobnost uporabnikov (gledalcev) s Pearsonovim koeficientom
- Izračunati napoved ocene filma za izbranega uporabnika s postopkom kolaborativnega filtriranja
- Oceniti povprečno napako priporočilniega sistema

1.1 Podatkovni set MovieLens

```
MovieLens data sets were collected by the GroupLens Research Project
at the University of Minnesota.

This data set consists of:
        * 100,000 ratings (1-5) from 943 users on 1682 movies.
        * Each user has rated at least 20 movies.
        * Simple demographic info for the users (age, gender, occupation, zip)

The data was collected through the MovieLens web site
(movielens.umn.edu) during the seven-month period from September 19th,
1997 through April 22nd, 1998. This data has been cleaned up - users
who had less than 20 ratings or did not have complete demographic
information were removed from this data set. Detailed descriptions of
the data file can be found at the end of this file.
```

1.2 Nalaganje podatkov

Uvozimo podatke iz datoteke v data frame

```
import numpy as np
import pandas as pd
import sklearn.cross_decomposition
```

```
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('u.data', sep='\t', names=column_names)
```

Preglej in izpiši nekaj vrstic podatkov:

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

Preberemo naslove filmov, in jih dodamo v podatkovni objekt

```
movie_titles = pd.read_csv("Movie_Id_Titles")
movie_titles.head()
df = pd.merge(df,movie_titles,on='item_id')
print(df.shape)
```

Izpiši nekaj vrstic podatkov:

```
100003, 6)
  user_id  item_id  rating  timestamp                         title_x  \
        0       50       5  881250949                Star Wars (1977)
        0      172       5  881250949  Empire Strikes Back, The (1980)
        0      133       1  881250949       Gone with the Wind (1939)
      196      242       3  881250949                  Kolya (1996)
      186      302       3  891717742        L.A. Confidential (1997)

                      title_y
               Star Wars (1977)
  Empire Strikes Back, The (1980)
        Gone with the Wind (1939)
                  Kolya (1996)
         L.A. Confidential (1997)
```

1.3 Vprašanje 1

Ugotovi, koliko uporabnikov obsegajo podatki, ter koliko je različnih filmov. Uporabi metodo nunique().

Koda in rezultat:

```python
# Print the number of users
num_users = df['user_id'].nunique()
print(f"Number of users: {num_users}")

# Print the number of unique values in the dataframe
num_unique_values = df.nunique().sum()
print(f"Number of unique values in the dataframe: {num_unique_values}")

# Print the number of movies
num_movies = df['item_id'].nunique()
print(f"Number of movies: {num_movies}")

df1 = df.groupby('user_id')['user_id'].count()
```

✓ 0.0s

```
Number of users: 944
Number of unique values in the dataframe: 53577
Number of movies: 1682
```

1.4 Podaki po uporabnikih

```python
df1 = df.groupby('user_id')['user_id'].count()
```

Izpiši df1, uporabi metodo describe(), ter izpiši df1.values. Kaj predstavljajo vrednosti ? Koliko je največja in najmanjša vrednost?

```
count    944.000000
mean     105.935381
std      100.933948
min        3.000000
25%       33.000000
50%       64.500000
75%      148.000000
max      737.000000
Name: user_id, dtype: float64
```

```
[  3 272  62  54  24 175 211 403  59  22 184 181  51 636  98 104 140  28
 277  20  48 179 128 151  68  78 107  25  79  34  43  36  41  24  20  25
  20  57 121  22  35  52 183 221 151  48  27  25  66 215  24  23  56  28
  65  21 187 106 154 382 208  21 232  93 200  80  38  30  34  65 131  38
 137  66  39  79  82  72  21  55  29  58 168 155  68 288  23 211  21  76
 300  98 388  20 400 278  56  63  27 136  59  67 216  29 111  23  64  22
  33 234 133  24  46  51  48  92 143  86  71 181  26  74  61  54  24 182
  45  23 184  30 353  30  22  26  25  55  35  47  51  24  21 107  33  26
 206 316  29  20  65  36  32 307 106  23  51  22  37  51 173 107 120  58
  42  23  63  30  20  69  69  37  22  28  27  43 177  38  63 115 273  42
  63 435  28  53 251  48  92  56 112 187  59  27  35 121 305  96  39 118
 181  40 216 386  20  43  42  22  64 230  33  33 132  37  25 126 128  93
 131  76  53  28  22 146 387 105 141  27  50  58  21  30 133  21  93 116
 480  94 124  48  29 159  24  23  20  81 238  22 195  26  56 161 122  77
  21  97 159  83 207  57  23  46  24  27 160 123 120  46  23 185 328 323
 138 278  51  22  71  95 518  54  23 434 258  26  22  53  47  32 288  62
  75  27 150 296 124 388 150 196 147 192 127 280  20 275  21 484  26 222
  32 112 397  20  21 294 223 262 245  87  75  22 174  23 154 125  51  89
  66 142 187 284 283  65 147  70 183  26 333  22 130  34  75 254  44  21
 201 237 190 232 103 199  58  41  50  44  42  25 233  26  24  76  43  21
```

1.5 Sortiramo uporabnike

```python
series_user_nratings = df.groupby('user_id')['user_id'].count()

ids_nratings_desc = series_user_nratings.sort_values(ascending=False)
ids_100_nratings = ids_nratings_desc.iloc[0:100]
```

```
print(ids_nratings_desc)
print(ids_100_nratings)
```

Kaj predstavljata zadnji dve spremenljvki, kateri uporabnik ima največ in najmanj podatkov? S print izpiši index in values serije.

```
user_id
405    737
655    685
13     636
450    540
276    518
      ...
558     20
34      20
36      20
926     20
0        3
Name: user_id, Length: 944, dtype: int64
user_id
405    737
655    685
13     636
450    540
276    518
      ...
506    242
932    241
886    240
798    239
244    238
Name: user_id, Length: 100, dtype: int64
User with most data: 405 Number of ratings: 737
User with least data: 0 Number of ratings: 3
```

```
indeksi_df = df['user_id'].isin(ids_100_nratings.index)
# Nova serija podatkov
df5 = df.loc[indeksi_df].sort_values('user_id')
```

Izpiši df5. Koliko je podatkov, kaj prestavljajo ?

33k, predstavljajo filme

```
       user_id  item_id  rating  timestamp  \
47524        1      192       4  875072547
56070        1       45       5  875241687
17675        1      100       5  878543541
15273        1       54       3  878543308
30482        1      256       4  889751712
...        ...      ...     ...        ...
90937      932      600       2  891252412
96793      932      523       4  891250080
77751      932      218       3  891250915
91378      932      148       2  891252140
99838      932      416       3  891250498

                                                   title
47524                                  Raging Bull (1980)
56070                              Eat Drink Man Woman (1994)
17675                                      Fargo (1996)
15273                                   Outbreak (1995)
30482  When the Cats Away (Chacun cherche son chat) (...
...                                                  ...
90937              Daniel Defoe's Robinson Crusoe (1996)
96793                           Cool Hand Luke (1967)
77751                                Cape Fear (1991)
91378                 Ghost and the Darkness, The (1996)
99838                                Old Yeller (1957)
```

1.6 Priprava podatkov

```
# Podatki so naši originalni podatki
train_data = df

n_users = train_data.user_id.nunique()
n_items = train_data.item_id.nunique()

print('Num. of Users: '+ str(n_users))
print('Num of Movies: '+ str(n_items))
```

```
Num. of Users: 944
Num of Movies: 1682
```

0 bomo nadomestili z NaN

```
def replaceZero(input_arr):
    out_arr = np.copy(input_arr)
    out_arr[out_arr == 0.0] = np.NaN
    return out_arr
```

Matrika uporabnikov - filmov

```
#Create two user-item matrices, one for training and another for testing
train_data_matrix = np.zeros((n_users, n_items))
for line in train_data.itertuples():
    train_data_matrix[line[1]-1, line[2]-1] = line[3]

train_data_2 = replaceZero(train_data_matrix)
train_data_2
```

V kakšno obliko smo preoblikovali podatke o ocenah filmov, kaj so vrstice in stolpci ?

Vrstice so uporabniki, stolpci so filmi

```
# Sortiraj po vsoti ocen za uporabnika
sm = train_data_matrix.sum(axis = 1)
print(sm.shape)
sort_ind = np.argsort(-sm)
sm2 = sm[sort_ind]

#
```

```
train_data_3 = train_data_2[sort_ind,:]
#
train_data_4 = train_data_3[0:100,:]


print(train_data_4.shape)
print(train_data_4)
np.nansum(train_data_4, axis=1)
```

Koliko oseb vsebujejo podatki v train_data_4 ?

100

```
num_people = train_data_4.shape[0]
print(f"Number of people in train_data_4: {num_people}")
```

✓ 0.0s

```
Number of people in train_data_4: 944
```

2 Pearsonova korelacija

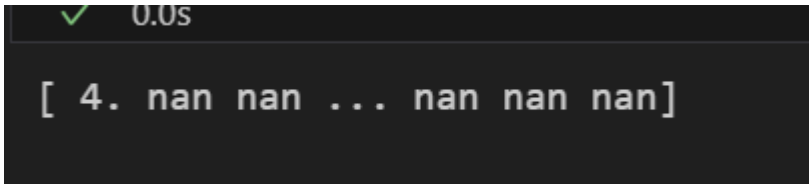
Definicija:

$$userSim(u,n) = \frac{\sum_{i \subset CR} (r_{ui} - \bar{r}_u) \cdot (r_{ni} - \bar{r}_n)}{\sqrt{\sum_{i \subset CR} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \subset CR} (r_{ni} - \bar{r}_n)^2}}$$

https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

2.1 Izberi podatke dveh izbranih uporabnikov

```
ur_1 = train_data_4[1,:]
ur_2 = train_data_4[50,:]
print(ur_1)
```



2.2 Izračunaj korelacijski koeficient

```
from scipy.stats import pearsonr

mask = ~np.isnan(ur_1) &  ~np.isnan(ur_2)
corr, _ = pearsonr(ur_1[mask],ur_2[mask])

print(corr)
```

Preskusi še izraučun za druge izbrane uporabnike. Kakšne so vrednosti koeficienta, ali so vedno pozitivne ?

Vedno je pozitiven.

2.3 Izračun matrike koeficientov

```
# Metoda za podobnost
def calcPearson(data_matr):
    pears = np.zeros([data_matr.shape[0],data_matr.shape[0]])
    for j in range(0, data_matr.shape[0]):
        for i in range(j, data_matr.shape[0]):
            pears[j,i] = np.nan;  # default value
            mask = ~np.isnan(data_matr[j,:]) & ~np.isnan(data_matr[i,:])
            if sum(mask) >= 2:
                pears[j,i], _ = pearsonr(data_matr[j,mask],data_matr[i,mask])
            pears[i,j] = pears[j,i]
        pears[j,j] = 0.0
    return pears
```

Izračunaj matriko Pearsonovih korelacij. Preglej vsebino matrike.

```
# Izracunaj pears_cor
```

```
pears_cor = calcPearson(train_data_4)

print(pears_cor)
print("Velikost: ",pears_cor.shape)
```

```
[[ 0.          0.19841011  0.22968366 ...  0.18835665  0.20816537
   0.25551185]
 [ 0.19841011  0.          0.10421103 ...  0.17698523 -0.03696457
   0.1732737 ]
 [ 0.22968366  0.10421103  0.         ...  0.1972609   0.03321634
   0.16781803]
 ...
 [ 0.18835665  0.17698523  0.1972609  ...  0.          0.21277649
   0.14222591]
 [ 0.20816537 -0.03696457  0.03321634 ...  0.21277649  0.
   0.34511434]
 [ 0.25551185  0.1732737   0.16781803 ...  0.14222591  0.34511434
   0.        ]]
Velikost:  (100, 100)
```

3 Izračun napovedi ocene filma za uporabnika

Predvideno oceno vsebine *i* za uporabnika *u* dobimo po formuli:

$$pred(u,i) = \bar{r}_u + \frac{\sum_n userSim(u,n) \cdot (r_{ni} - \bar{r}_n)}{\sum_n userSim(u,n)}$$

```
# Postopek

# Matrika podobnosti uporabnikov
user_similarity = np.copy(pears_cor)

# Povprecni rating vsakega uporabnika
mean_user_rating = np.nanmean(train_data_4, axis=1)
# Povpr rating kot stolpcni vektor
c3 = mean_user_rating[:, np.newaxis]

# Razlika rating - povprecje, vsebuje nan
ratings_diff = (train_data_4 - mean_user_rating[:, np.newaxis])
# Relativni ratnig, vsebuje 0 kjer ni ocene
```

```
rel_ratings = np.nan_to_num(ratings_diff)
```

Kaj pomenijo vrednosti v matriki rel_ratings ?

Relativne ocene uporabnikov. rel_ratings je matrika, kjer vsak element predstavlja razliko med oceno uporabnika in njegovo povprečno oceno, pri čemer so manjkajoče ocene nadomeščene z ničlami.

```
# Izracunaj napoved ocen filmov za izbranega uporabnika

# Izbrani ciljni uporabnik izmed 0 .. 100
user_id = 20

# Podobnost ostalih z izbranim uporabnikom
usr1 = user_similarity[:, user_id]


usr_sim_ind = usr1 < 0.35

usr1[usr_sim_ind] = 0.0

# Kaj pomeni ?
c2 = usr1.dot(rel_ratings)
# Rezultat : ulomek - napoved relativne ocene za vse filme za enega uporabnika
c4 = c2 / np.array([np.abs(usr1).sum()]).T

# Napoved ocene filmov : pristejemo povprecno oceno uporabnika
povpr_ocena = mean_user_rating[user_id]
napoved_ocen = povpr_ocena + c4
```

Izpiši napoved_ocen

```
[4.35918561 3.85031239 3.95487089 ... 4.08256881 4.08256881 4.08256881]
```

```
# naredi Dataframe in vstavi dejansko oceno in njeno napoved
rezultat = pd.DataFrame(napoved_ocen, columns=['Napoved'])
rezultat['Ocena'] = train_data_4[user_id,:]
rezultat['Napaka napovedi'] = rezultat['Napoved']- rezultat['Ocena']

print(rezultat)
```

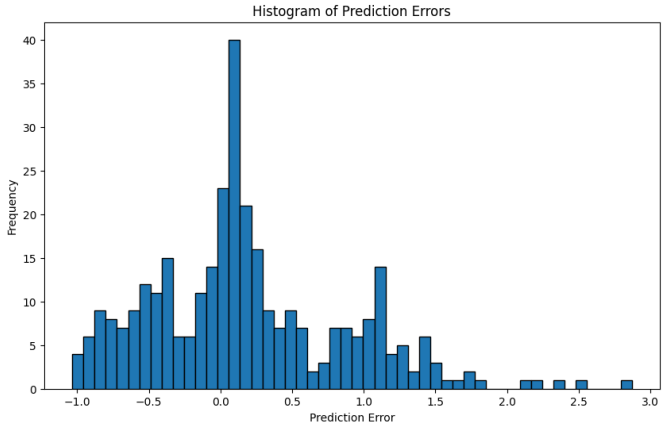
Kaj smo dobili v rezultatu ?

	Napoved	Ocena	Napaka napovedi
0	4.359186	NaN	NaN
1	3.850312	NaN	NaN
2	3.954871	NaN	NaN
3	4.422432	5.0	-0.577568
4	3.882596	NaN	NaN
...	...	...	...
1677	4.082569	NaN	NaN
1678	4.082569	NaN	NaN
1679	4.082569	NaN	NaN
1680	4.082569	NaN	NaN
1681	4.082569	NaN	NaN

[1682 rows x 3 columns]

3.1 Ocena napake

Izriši histogram napak v rezultatu. Vstavi:



Izračunaj povprečno kvadratično napako (RMSE) :

```
povpr = np.nanmean(((rezultat['Napaka napovedi'].to_numpy()) **2 ))
rmse = np.sqrt(povpr)
print('RMSE napaka: ', rmse)
```

Koliko je bila povprečna napaka napovedi, in koliko napovedi je bilo generiranih ?

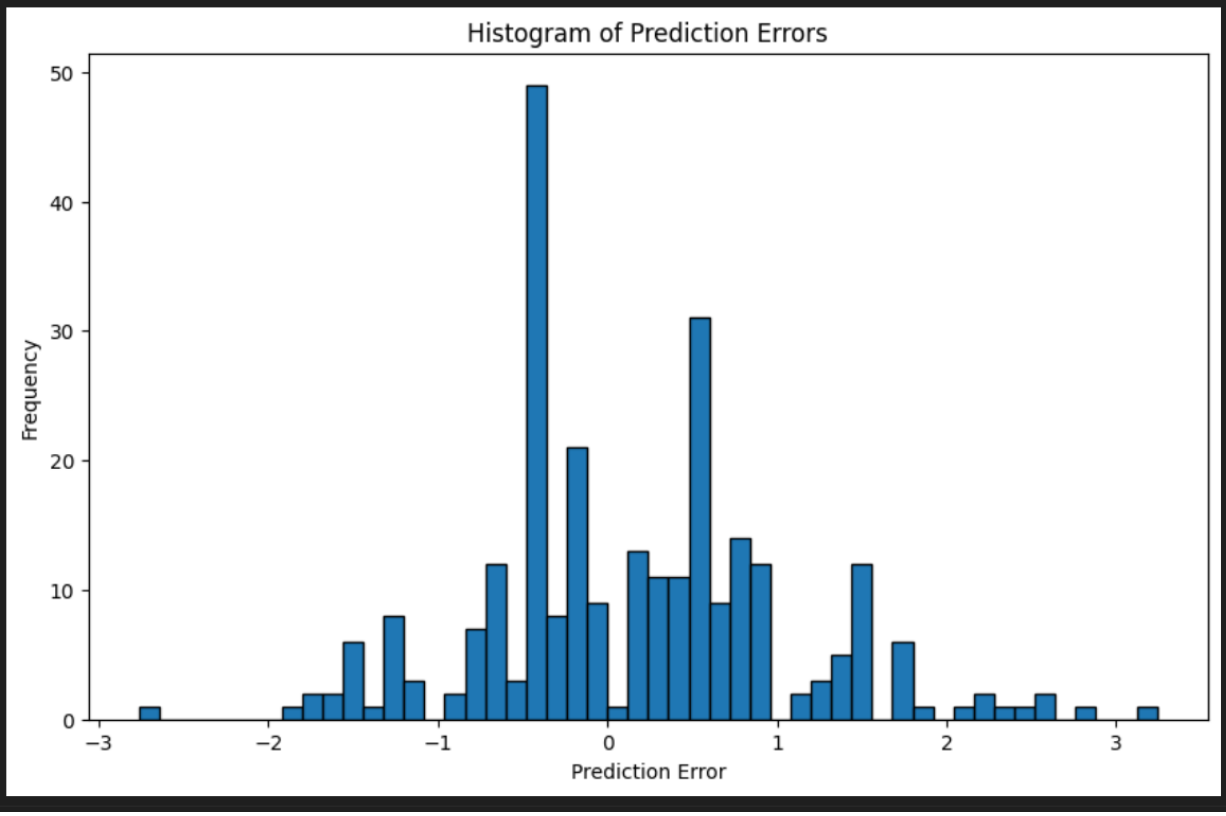
```
RMSE napaka:  0.7163323120254063
```

3.2 Izračun napovedi za drugega uporabnika

Izračunaj napovedane ocene in napako za drugega izbranega uporabnika. Vstavi rezultate:

Uporabnik 69

```
RMSE napaka:  0.9237754160298097
```



---

4 Napovedane ocene vseh uporabnikov

```
# Izbira uporabnikov: podobnost
min_podobnost = 0.5
```

```
# Katerih ne upostevamo, premajhna podobnost
i1 = pears_cor[:,:] < min_podobnost

user_similarity = np.copy(pears_cor);
user_similarity[i1] = 0.0

# To so zdaj vsote abs vrednosti podobnih userjev za vsakega userja
c5 = np.array([np.abs(user_similarity).sum(axis = 0)]).T

# Kaj je to. Zgoraj ulomek, zmnozek relat.ocene in podobnosti
calc1 = user_similarity.dot(rel_ratings)

# ulomek
calc2 = calc1 / c5

# napoved ocene : matrika za vse uporabnike in vse filme
calc3 = calc2 + c3
```

Preglej vsebino rezultata calc3, kaj predstavljajo vrednosti in kako smo jih izračunali (na kratko)?

```
[[4.30004484 3.75443941 3.73639465 ... 3.86481481 3.86499619 3.86170713]
 [3.23881194 2.77817521 2.71540882 ... 2.9080292  2.90824017 2.9020701 ]
 [3.40425765 2.9786871  2.9697109  ... 3.09748428 3.09782524 3.0929024 ]
 ...
 [4.43456879 3.93437221 3.88965405 ... 4.03240741 4.03263859 4.02937758]
 [4.29464561 3.67985674 3.53516616 ... 3.73390558 3.73464717 3.73390558]
 [3.98890817 3.51429702 3.52267719 ... 3.6512605  3.65155706 3.64322027]]
```

Kaj vsebuje matrika user_similarity?

Podobnosti uporabnikov.

```
[[0.         0.19841011 0.22968366 ... 0.18835665 0.20816537 0.25551185]
 [0.19841011 0.         0.10421103 ... 0.17698523 0.         0.1732737 ]
 [0.22968366 0.10421103 0.         ... 0.1972609  0.         0.16781803]
 ...
 [0.18835665 0.17698523 0.1972609  ... 0.         0.21277649 0.14222591]
 [0.20816537 0.         0.         ... 0.21277649 0.         0.34511434]
 [0.25551185 0.1732737  0.16781803 ... 0.14222591 0.34511434 0.        ]]
```

Kaj se spremeni, če spremeniš min_podobnost ? Kaj torej pomeni ta parameter ?

Meja za izločanje tistih ki si niso dovolj podobni.

Več uporabnikov poveže skupaj. Torej če ga preveč povečamo ne dobimo povezav, torej so 0 ali nan.

4.1 Ocena povprečne napake napovedi RMSE

Izračunamo srednjo kvadratično napako RMSE:

```
from sklearn.metrics import mean_squared_error
from math import sqrt

def rmse(prediction, ground_truth):
    """ Izracun RMSE napake za podane matrike

    Args:
        prediction (_type_): _description_
        ground_truth (_type_): _description_

    Returns:
        _type_: _description_
    """    ''''''
    ground_tr_2 = np.nan_to_num(ground_truth);
    prediction2 = np.nan_to_num(prediction);
    pred_valid = prediction2[:,0] > 0.0;

    # samo veljavne vrstice kjer pred ni nan
    prediction3 = prediction2[pred_valid,:]
    ground3 = ground_tr_2[pred_valid,:];

    prediction1 = prediction3[ground3.nonzero()].flatten()
    ground_truth1 = ground3[ground3.nonzero()].flatten()

    return sqrt(mean_squared_error(prediction1, ground_truth1))
```

```
# Preizkusi in izpisi
# calc3 so napovedi, train so dejanske ocene

napaka = rmse(calc3, train_data_4);

print("RMSE napaka napovedi: ", napaka)
```

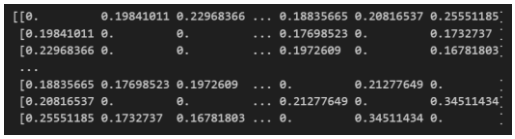
Koliko je bila RMSE napaka napovedanih ocen ?

```
RMSE napaka napovedi:  0.9448337397191773
```

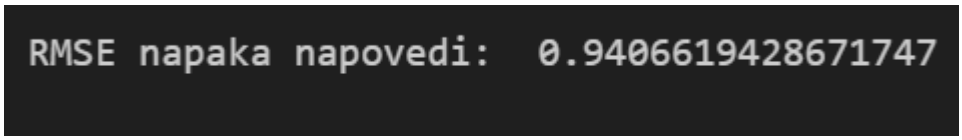
4.2 Naloga 2: napaka napovedi

Spreminjaj število podobnih uporabnikov pri izračunu napovedi ocen, in izmeri povprečno napako.

Iz podobnost 0.1 na 0.15. Error se je zmanjšal



```
[[0.         0.19841011 0.22968366 ... 0.18835665 0.20816537 0.25551185]
 [0.19841011 0.         0.         ... 0.17698523 0.         0.1732737 ]
 [0.22968366 0.         0.         ... 0.1972609  0.         0.16781803]
 ...
 [0.18835665 0.17698523 0.1972609  ... 0.         0.21277649 0.        ]
 [0.20816537 0.         0.         ... 0.21277649 0.         0.34511434]
 [0.25551185 0.1732737  0.16781803 ... 0.         0.34511434 0.        ]]
```



```
RMSE napaka napovedi:  0.9406619428671747
```

4.3 Dodatna naloga 3:

Izdelaj metodo za izračun napovedanih ocen, in jo preskusi:

```python
# Metoda naj vrne matriko napovedanih ocen.

def predictRating (rating_data, user_similarity):
    """ Izracun napovedi ocen
        Metoda naj vrne matriko napovedanih ocen.

    Args:
        rating_data (_type_): _description_
        user_similarity (_type_): _description_

    Returns:
        _type_: _description_
    """
    pred_rating = 0;
    # Povprecni rating uporabnika
    # Povpr rating kot stolpcni vektor
    # Razlika rating - povprecje, vsebuje nan
    # Relativni ratnig, vsebuje 0 kjer ni ocene

    return pred_rating
```