### 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

## 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.

## Nalaganje podatkov

Uvozimo podatke iz datoteke v data frame

import numpy as np

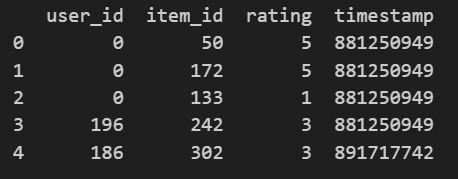
import pandas as pd

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:



Preberemo naslove filmov, in jih dodamo v podatkovni objekt

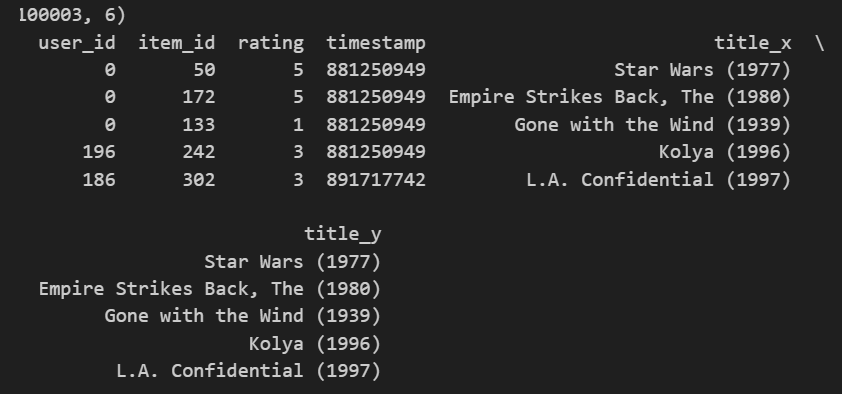
movie\_titles = pd.read\_csv("Movie\_Id\_Titles")

movie\_titles.head()

df = pd.merge(df,movie\_titles,on='item\_id')

print(df.shape)

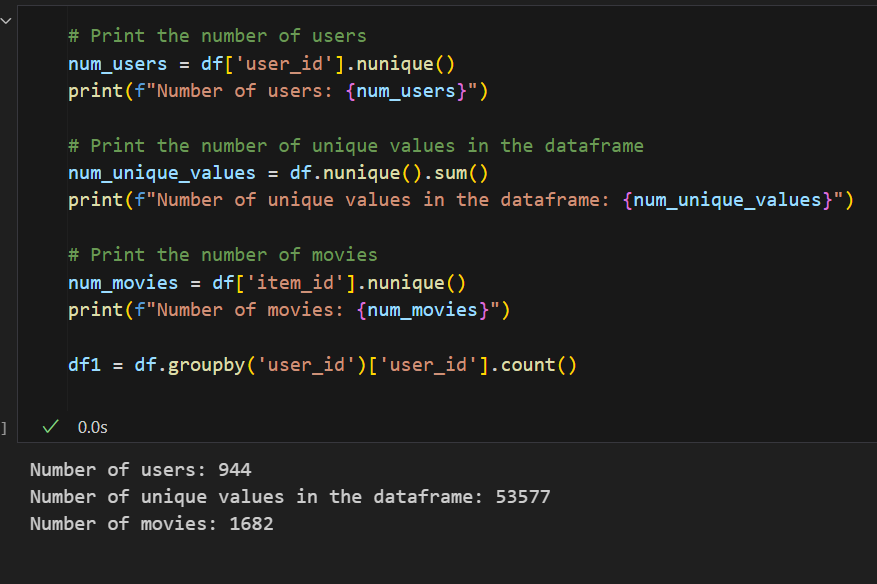
Izpiši nekaj vrstic podatkov:



## Vprašanje 1

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

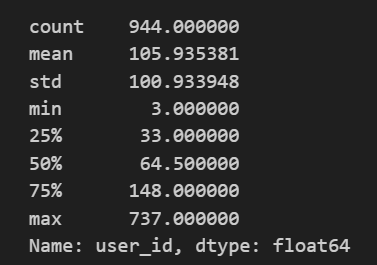
Koda in rezultat:

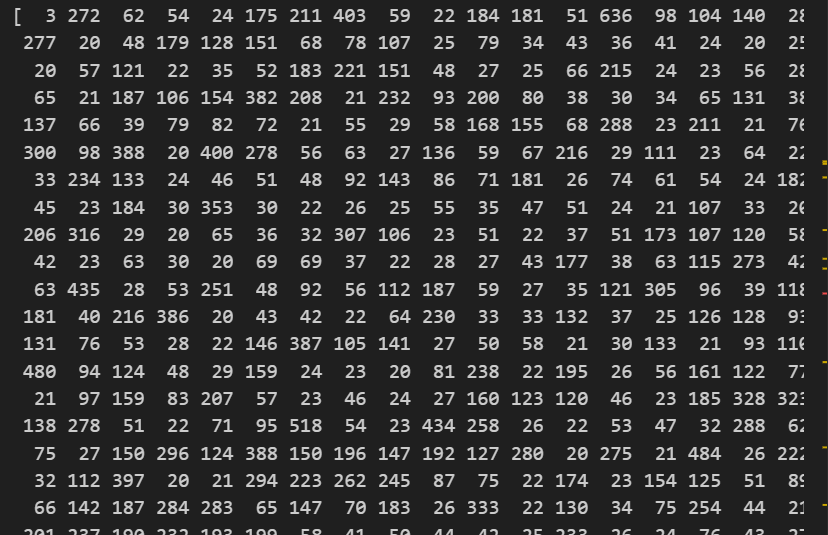


## Podaki po uporabnikih

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?





## Sortiramo uporabnike

series\_user\_nratings = df.groupby('user\_id')['user\_id'].count()

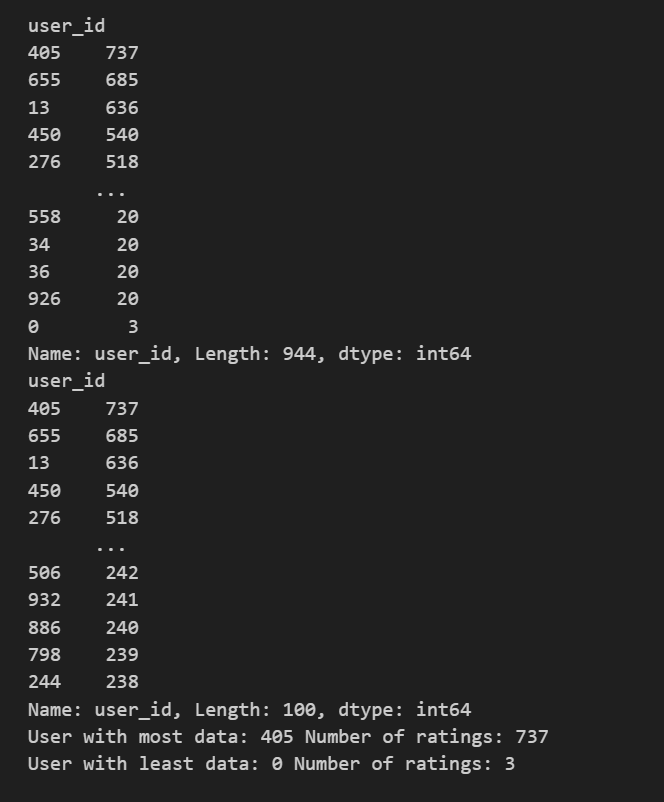
ids\_nratings\_desc = series\_user\_nratings.sort\_values(ascending=False)

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.



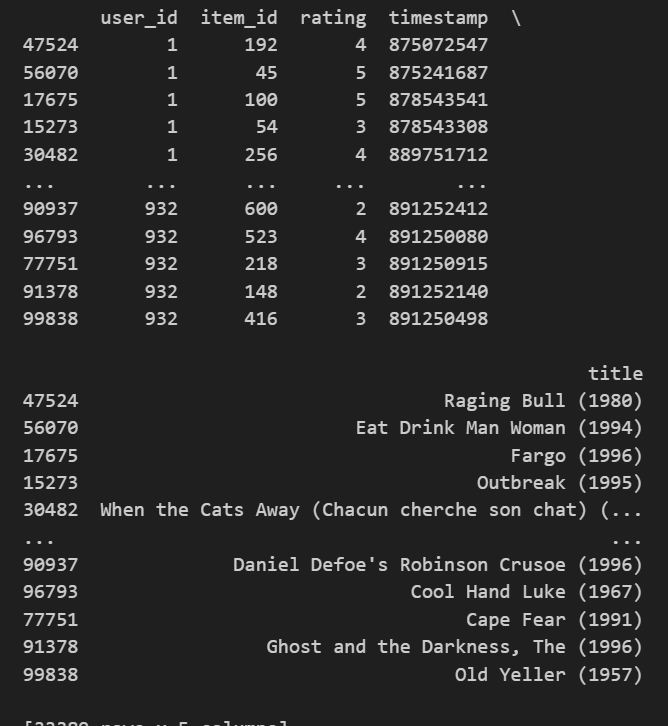
indeksi\_df = df['user\_id'].isin(ids\_100\_nratings.index)

# Nova serija podatkov

df5 = df.loc[indeksi\_df].sort\_values('user\_id')

Izpiši df5. Koliko je podatkov, kaj prestavljajo ?

33k, predstavljajo filme



## Priprava podatkov

# Podatki so naši originalni podatki

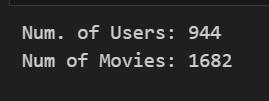
train\_data = df

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))



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,:]

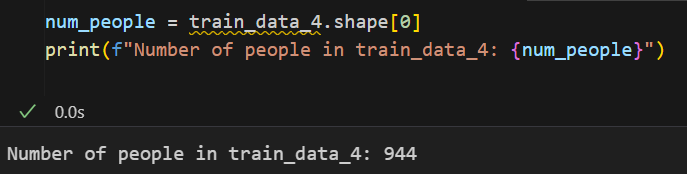
print(train\_data\_4.shape)

print(train\_data\_4)

np.nansum(train\_data\_4, axis=1)

Koliko oseb vsebujejo podatki v train\_data\_4 ?

100

**

# Pearsonova korelacija

Definicija:



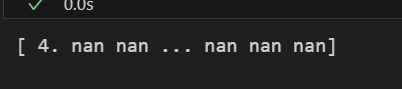
<https://en.wikipedia.org/wiki/Pearson_correlation_coefficient>

## Izberi podatke dveh izbranih uporabnikov

ur\_1 = train\_data\_4[1,:]

ur\_2 = train\_data\_4[50,:]

print(ur\_1)



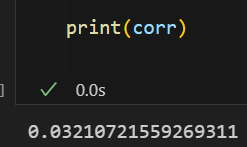
## Izračunaj korelacijski koeficient

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.

## 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

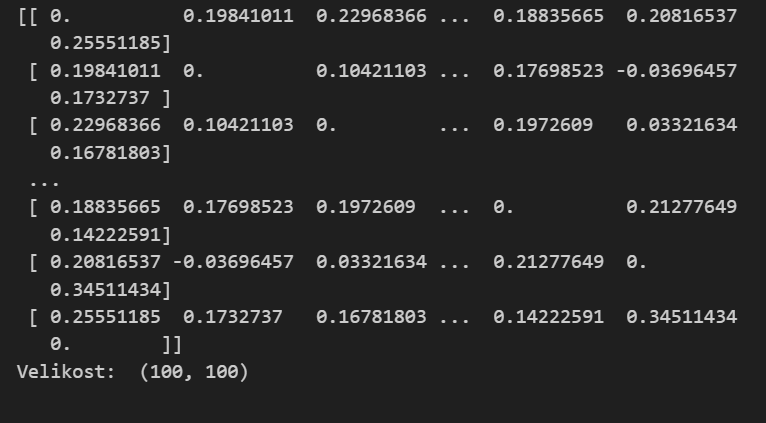
Izračunaj matriko Pearsonovih korelacij. Preglej vsebino matrike.

# Izracunaj pears\_cor

pears\_cor = calcPearson(train\_data\_4)

print(pears\_cor)

print("Velikost: ",pears\_cor.shape)



# Izračun napovedi ocene filma za uporabnika

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



# 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

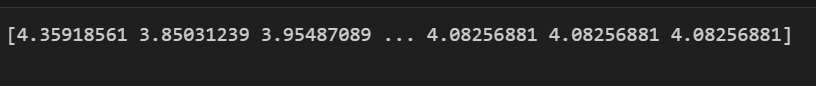
c4 = c2 / np.array([np.abs(usr1).sum()]).T

# Napoved ocene filmov : pristejemo povprecno oceno uporabnika

povpr\_ocena = mean\_user\_rating[user\_id]

napoved\_ocen = povpr\_ocena + c4

Izpiši napoved\_ocen



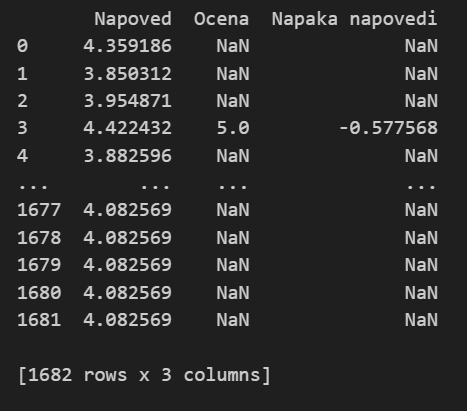
# naredi Dataframe in vstavi dejansko oceno in njeno napoved

rezultat = pd.DataFrame(napoved\_ocen, columns=['Napoved'])

rezultat['Ocena'] = train\_data\_4[user\_id,:]

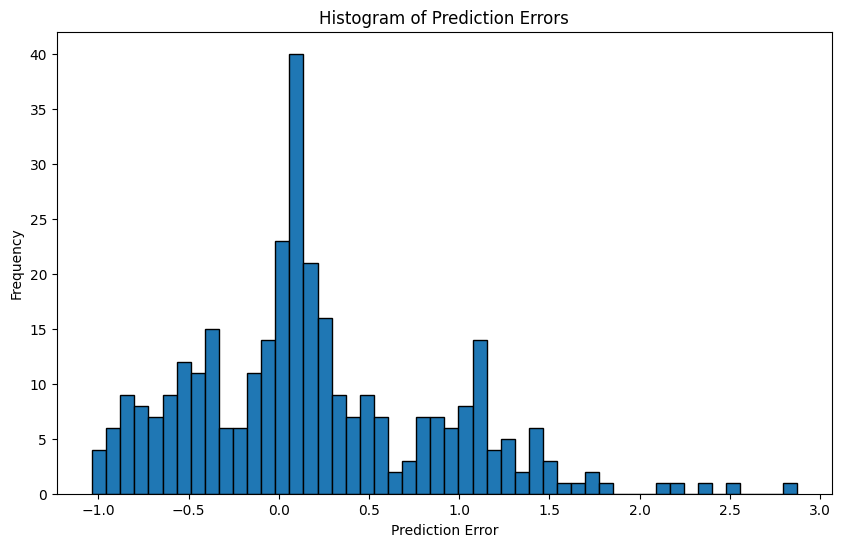
rezultat['Napaka napovedi'] = rezultat['Napoved']- rezultat['Ocena']

print(rezultat)

Kaj smo dobili v rezultatu ? 

## Ocena napake

Izriši histogram napak v rezultatu. Vstavi:



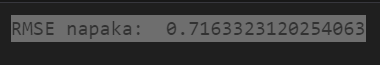
Izračunaj povprečno kvadratično napako (RMSE) :

povpr = np.nanmean(((rezultat['Napaka napovedi'].to\_numpy()) \*\*2 ))

rmse = np.sqrt(povpr)

print('RMSE napaka: ', rmse)

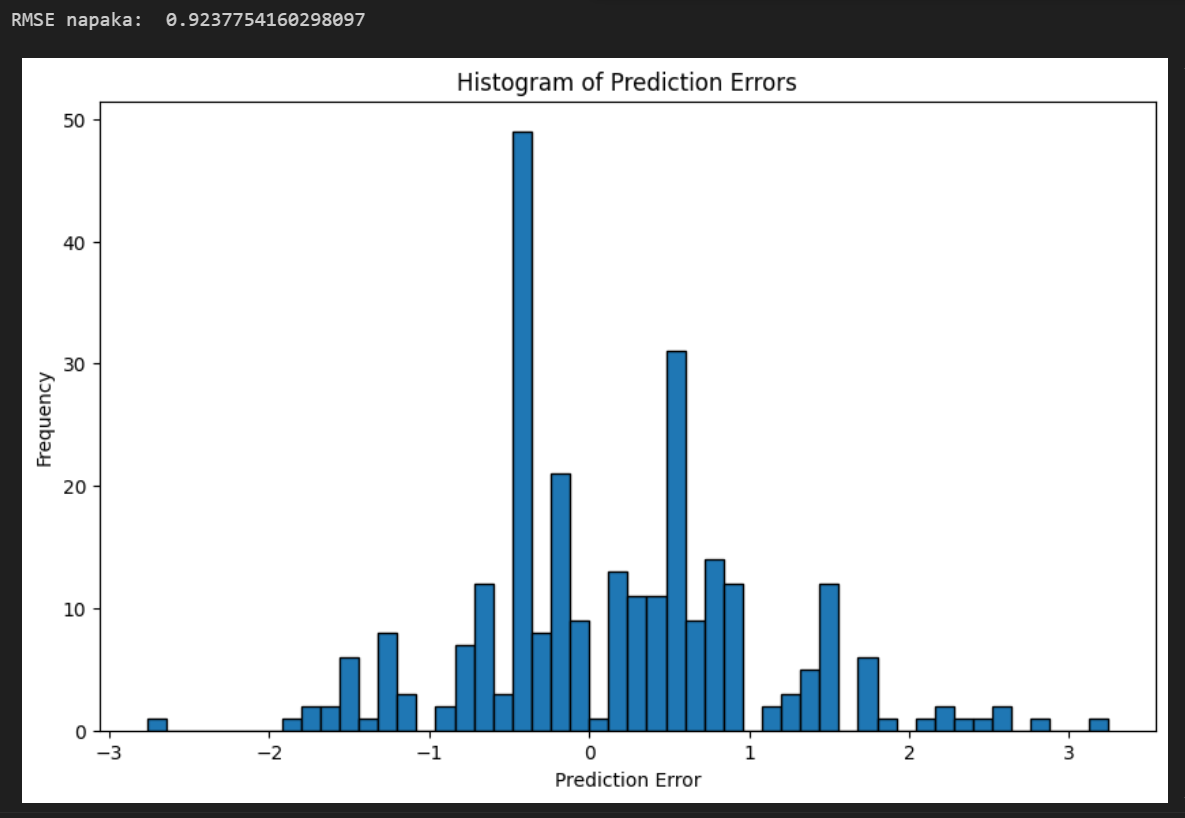
Koliko je bila povprečna napaka napovedi, in koliko napovedi je bilo generiranih ?



## Izračun napovedi za drugega uporabnika

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

Uporabnik 69



# 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)

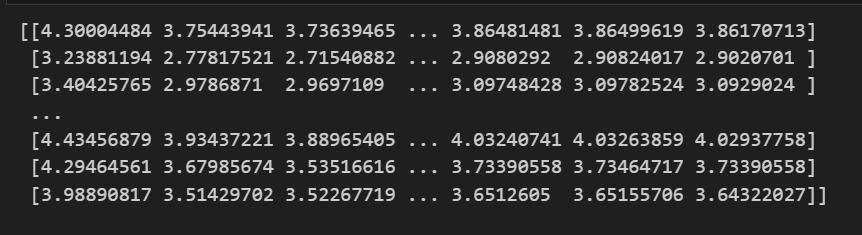
# ulomek

calc2 = calc1 / c5

# napoved ocene : matrika za vse uporabnike in vse filme

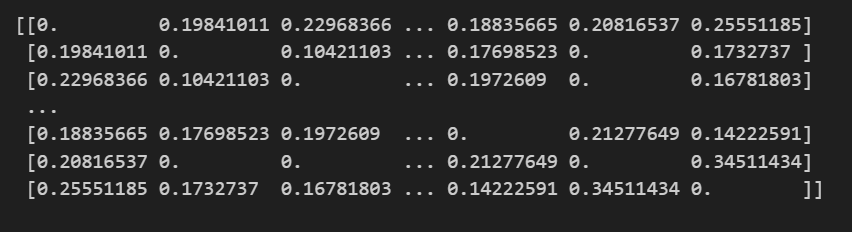
calc3 = calc2 + c3

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



Kaj vsebuje matrika user\_similarity?

Podobnosti uporabnikov.



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.

## 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)

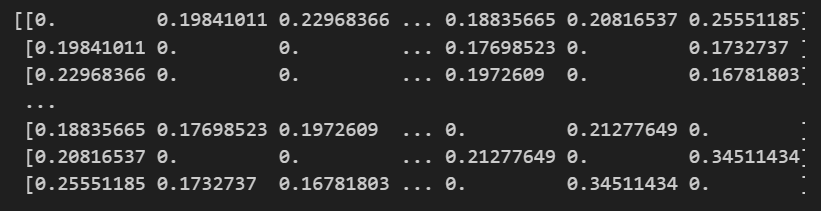
Koliko je bila RMSE napaka napovedanih ocen ?

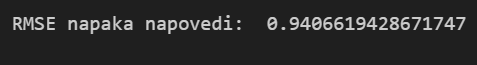


## Naloga 2: napaka napovedi

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

Iz podobnost 0.1 na 0.15. Error se je zmanjšal





## Dodatna naloga 3:

Izdelaj metodo za izračun napovedanih ocen, in jo preskusi:

# 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